Boosting with Structural Sparsity

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

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

Abstract

Despite popular belief, boosting algorithms and related coordinate descent methods are prone to overfitting. We derive modifications to AdaBoost and related gradient-based coordinate descent methods that incorporate, along their entire run, sparsity-promoting penalties for the norm of the predictor that is being learned. The end result is a family of coordinate descent algorithms that integrates forward feature induction and back-pruning through sparsity promoting regularization along with an automatic stopping criterion for feature induction. We study penalties based on the ℓ_1, ℓ_2, and ℓ_∞ norm of the learned predictor and also introduce mixed-norm penalties that build upon the initial norm-based penalties. The mixed-norm penalties facilitate structural sparsity of the parameters of the predictor, which is a useful property in multiclass prediction and other related learning tasks. We report empirical results that demonstrate the power of our approach in building accurate and structurally sparse models from high dimensional data.

Keywords: Boosting, coordinate descent, group sparsity, feature induction and pruning.

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. Rather than give a detailed overview of boosting, we refer the reader to Meir and Rätsch (2003) or Schapire (2003) and the numerous references therein. While the analysis of boosting algorithms suggests that boosting attempts to find large ℓ_1 margin predictors subject to their weight vectors belonging to the simplex (Schapire et al., 1998), AdaBoost and boosting algorithms in general do not directly use the ℓ_1-norm of their weight vectors. Many boosting algorithms can also be viewed as forward-greedy feature induction procedures. In this view, the weak-learner provides new predictors which seem to perform well either in terms of their error-rate with respect to the distribution that boosting maintains or in terms of their potential of reducing the empirical loss (see, e.g. Schapire and Singer (1999)). Thus, 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 original AdaBoost algorithm (Freund and Schapire, 1997) and its confidence-rated counterpart (Schapire and Singer, 1999) are notable examples of this forward-greedy feature induction and weight assignment procedure, where the difference between the two variants of AdaBoost boils down mostly to the weak-learner's feature scoring and selection scheme.

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 and Yu, 2005). The empirical success of Boosting algorithms helped popularize the view that boosting algorithms are relatively resilient to overfitting. However, several researchers have noted the deficiency of the forward-greedy boosting algorithm and suggested alternative coordinate descent algorithms, such as totally-corrective boosting (Warmuth et al., 2006) and a combination of forward induction with backward pruning (Zhang, 2008). The algorithms that we present in this paper build on existing boosting and other coordinate descent procedures while incorporating, throughout their run, regularization on the weights of the selected features. The added regularization terms influence both the induction and selection of new features and the weight assignment steps. Moreover, as we discuss below, the regularization term 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. Furthermore, the explicit incorporation of regularization also enables us to group features and impose structural sparsity on the learned weights, which is the focus and one of the main contributions of the paper.

Our starting point is a simple yet effective modification to AdaBoost that incorporates, along the entire run, an ℓ_1 penalty for the norm of the weight vector it constructs. The update we devise can be used both for weight optimization and induction of new accurate hypotheses while taking the resulting ℓ_1-norm into account, and it also gives a criterion for terminating the boosting process. A closely related approach was suggested by Dudík et al. (2007) in the context of maximum entropy. We provide new analysis for the classification case that is based on an abstract fixed-point theorem. This rather general theorem is also applicable to other norms, in particular the ℓ_∞ norm which serves as a building block for imposing structural sparsity.

We now describe more formally our problem setting. As mentioned above, our presentation and analysis is for classification settings, though our derivation can be easily extended to and used in regression and other prediction problems (as demonstrated in the experimental section and the appendix). For simplicity of exposition, 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 $\boldsymbol{x} \in \mathbb{R}^n$. We discuss in Sec. 8 the use of our framework in the presence of countably infinite features, also known as the task of feature induction. In the binary case, 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. We build on the generalization of AdaBoost from Collins et al. (2002), which analyzes the the exponential-loss and the log-loss, as the means to obtain this type of high confidence linear prediction. Formally, given a sample $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$, the algorithm focuses on finding $\boldsymbol{w}$ for which one of the following empirical losses is small:

$$\sum_{i=1}^{m} \log\left(1 + \exp(-y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i)\right) \ \text{(LogLoss)} \qquad \sum_{i=1}^{m} \exp\left(-y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i)\right) \ \text{(ExpLoss)} \ . \tag{1}$$

We give our derivation and analysis for the log-loss as it also encapsulates the derivation for the exp-loss. We first derive an adaptation that incorporates the ℓ_1-norm of the weight vector into the empirical loss,

$$Q(\boldsymbol{w}) = \sum_{i=1}^{m} \log\left(1 + \exp(-y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i))\right) + \lambda \|\boldsymbol{w}\|_1 . \tag{2}$$

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 (see for instance Koh et al. (2007) and the references therein). This regularization has many advantages, including its ability to yield sparse weight vectors $\boldsymbol{w}$ and, under certain conditions, to recover the true sparsity of $\boldsymbol{w}$ (Meinshausen and Bühlmann, 2006; Zhao and Yu, 2006). Our derivation shares similarity with the analysis in Dudík et al. (2007) for maximum-entropy models, however, we focus on boosting and more general regularization strategies. We provide an explicit derivation and proof for ℓ_1 regularization in order to make the presentation more accessible and to motivate the more complex derivation presented in the sequel.

Next, we replace the ℓ_1-norm penalty with an ℓ_∞-norm penalty, proving an abstract primal-dual fixed point theorem on sums of convex losses with an ℓ_∞ regularizer that we use throughout the paper. While this penalty cannot serve as a regularization term in isolation, as it is oblivious to the value of most of the weights of the vector, it serves as an important building block for achieving structural sparsity by using mixed-norm regularization, denoted ℓ_1/ℓ_p. Mixed-norm regularization is used when there is a partition of or structure over the weights that separates them into disjoint groups of parameters. Each group is tied together through an ℓ_p-norm regularizer. 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 multitask learning we have a set of tasks $\{1, \ldots, k\}$ and a weight vector $\boldsymbol{w}_r$ for each task. Without loss of generality, we assume that all the tasks share training examples (we could easily sum only over examples specific to a task and normalize appropriately). Our goal is to learn a matrix W that minimizes

$$Q(W) = L(W) + \lambda R(W) = \sum_{i=1}^{m} \sum_{r=1}^{k} \log\left(1 + \exp(-y_{i,r}(\boldsymbol{w}_r \cdot \boldsymbol{x}_i))\right) + \lambda \sum_{j=1}^{n} \left\|\overline{\boldsymbol{w}}_j\right\|_p \quad . \tag{3}$$

The other generalization we describe is the multiclass logistic loss. In this setting, we assume again there are k weight vectors $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_k$ 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 loss of $W = [\boldsymbol{w}_1 \cdots \boldsymbol{w}_k]$ is

$$Q(W) = L(W) + \lambda R(W) = \sum_{i=1}^{m} \log\left(1 + \sum_{r \neq y_i} \exp(\boldsymbol{w}_r \cdot \boldsymbol{x}_i - \boldsymbol{w}_{y_i} \cdot \boldsymbol{x}_i)\right) + \lambda \sum_{j=1}^{n} \left\|\overline{\boldsymbol{w}}_j\right\|_p \quad . \tag{4}$$

In addition to the incorporation of ℓ_1/ℓ_∞ regularization we also derive a completely new upper bound for the multiclass loss. Since previous multiclass constructions for boosting assume that the each base hypothesis provides a different prediction per class, they are not directly applicable to the more common multiclass setting discussed in this paper, which allocates a dedicated predictor per class. The end result is an efficient boosting-based update procedures for both multiclass and multitask logistic regression with ℓ_1/ℓ_∞ regularization. Our derivation still follows the skeleton of templated boosting updates from Collins et al. (2002) in which multiple weights can be updated on each round.

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 bound above the log-loss by a quadratic function. We term the resulting update GradBoost as it updates one or more coordinates in a fashion that follows gradient-based updates. Similar to the generalization of AdaBoost, we study ℓ_1 and ℓ_∞ penalties by *reusing* the fixed-point theorem. We also derive an update with ℓ_2 regularization. Finally, we derive GradBoost updates with both ℓ_1/ℓ_∞ and ℓ_1/ℓ_2 mixed-norm regularizations.

The end result of our derivations is a portfolio of coordinate descent-like algorithms for updating one or more blocks of parameters (weights) for logistic-based problems. These types of problems are often solved by techniques based on Newton's method (Koh et al., 2007; Lee et al., 2006), which can become inefficient when the number of parameters n is large since they have complexity at least quadratic (and usually cubic) in the number of features. In addition to new descent procedures, the bounds on the loss-decrease made by each update can serve as the means for selecting (inducing) new features. By re-updating previously selected weights we are able to prune-back existing features. Moreover, we can alternate between pure weight updates (restricting ourselves to the current set of hypotheses) and pure induction of new hypotheses (keeping the weight of existing hypotheses intact). Further, our algorithms provide sound criteria for terminating boosting. As demonstrated empirically in our experiments, the end result of boosting with the structural sparsity based on ℓ_1, ℓ_1/ℓ_2, or mixed-norm ℓ_1/ℓ_∞ regularizations is a compact and accurate model. The structural sparsity is especially useful in complex prediction problems, such as multitask and multiclass learning, when features are expensive to compute. The mixed-norm regularization avoids the computation of features at test time since entire rows of W may set to be zero.

The paper is organized as follows. In the reminder of this section we give a brief overview of related work. In Sec. 2 we describe the modification to AdaBoost which incorporates ℓ_1 regularization. We then switch to ℓ_∞ norm regularization in Sec. 3 and provide a general primal-dual theorem which also serves us in later sections. We use the two norms in Sec. 4, where describe our first structural ℓ_1/ℓ_∞ regularization. We next turn the focus to gradient-based coordinate descent and describe in Sec. 5 the GradBoost update with ℓ_1 regularization. In Sec. 6 we describe the GradBoost version of the structural ℓ_1/ℓ_∞ regularization and in Sec. 7 a GradBoost update with ℓ_1/ℓ_2 regularization. In Sec. 8 we discuss the implications of the various updates for learning sparse models with forward feature induction and backward pruning, showing how to use the updates during boosting and for termination of the boosting process. In Sec. 9 we briefly discuss the convergence properties of the various updates. Finally, in Sec. 10 we describe the results of experiments with binary classification, regression, multiclass, and multitask problems.

Related work: Our work intersects with several popular settings, builds upon existing work, and is related to numerous boosting and coordinate descent algorithms. It is clearly impossible to cover in depth the related work. We recap here the connections made thus far, and also give a short overview in attempt to distill the various contributions of this work. Coordinate descent algorithms are well studied in the optimization literature. An effective use of coordinate descent algorithms for machine learning tasks was given for instance in Zhang and Oles (2001), which was later adapted by Madigan and colleagues for text classification (2005). Our derivation follows the structure of template-based algorithm from Collins et al. (2002) while incorporating regularization and scoring the regularized base-hypotheses in a way analogous to the maximum-entropy framework of Dudík et al. (2007). The base GradBoost algorithm we derive shares similarity with LogitBoost (Friedman et al., 2000), while similar bounding techniques to ours were first suggested by Dekel et al. (2005).

Learning sparse models for the logistic loss and other convex loss functions with ℓ_1 regularization is the focus of a voluminous amount of work in different research areas, from statistics to information theory. For instance, see Duchi et al. (2008) or Zhang (2008) and the numerous references therein for two recent machine learning papers that focus on ℓ_1 domain constraints and forward-backward greedy algorithms in order to obtain sparse models with small ℓ_1 norm. The alternating induction with weight-updates of Zhang (2008) is also advocated in the boosting literature by Warmuth and colleagues, who term the approach "totally corrective" (2006). In our setting, the backward-pruning is not performed in a designated step or in a greedy manner but is rather a different facet of the weight update. Multiple authors have also studied the setting of mixed-norm regularization. Negahban and Wainwright (2008) recently analyzed the structural sparsity characteristic of the ℓ_1/ℓ_∞ mixed-norm, and ℓ_1/ℓ_2-regularization was analyzed by Obozinski et al. (2008). Group Lasso and tied models through absolute penalties are of great interest in the statistical estimation literature. See for instance Meinshausen and Bühlmann (2006); Zhao and Yu (2006); Zhao et al. (2006); Zhang et al. (2008), where the focus is on consistency and recovery of the true non-zero parameter rather on efficient learning algorithms for large scale problems. The problem of simultaneously learning multiple tasks is also the focus of many studies, see Evgeniou et al. (2005); Rakotomamonjy et al. (2008); Jacob et al. (2008) for recent examples and the references therein. In this paper we focus on a specific aspect of multiple task learning through structured regularization, which can potentially be used in other multitask problems such as multiple kernel learning.

```
INPUT:    Training set $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$ ;
          Update templates $\mathcal{A} \subseteq \mathbb{R}_+^n$ s.t. $\forall \boldsymbol{a} \in \mathcal{A} \ \max_i \sum_{j=1}^n a_j |x_{i,j}| \le 1$
          regularization $\lambda$ ; number of rounds $T$
FOR $t = 1$ to $T$
    // Compute importance weights
    FOR $i = 1$ to $m$
        SET $q^t(i) = \frac{1}{1+\exp(y_i(\boldsymbol{w}^t \cdot \boldsymbol{x}_i))}$
    CHOOSE $\boldsymbol{a} \in \mathcal{A}$
    // Compute feature correlations
    FOR $j$ s.t. $a_j \neq 0$
        $\mu_j^+ = \sum_{i:y_i x_{i,j} > 0} q^t(i)|x_{i,j}|$ ,   $\mu_j^- = \sum_{i:y_i x_{i,j} < 0} q^t(i)|x_{i,j}|$
        // Compute change in weights  ($\delta_j^t = 0$ for all $j$ s.t. $a_j = 0$)
        $\delta_j^t = \begin{cases} -w_j^t & \text{if } \left|\mu_j^+ e^{w_j^t/a_j} - \mu_j^- e^{-w_j^t/a_j}\right| \le \lambda \\ a_j \log \frac{-\lambda + \sqrt{\lambda^2 + 4\mu_j^+ \mu_j^-}}{2\mu_j^-} & \text{if } \mu_j^+ e^{w_j^t/a_j} > \mu_j^- e^{-w_j^t/a_j} + \lambda \\ a_j \log \frac{\lambda + \sqrt{\lambda^2 + 4\mu_j^+ \mu_j^-}}{2\mu_j^-} & \text{if } \mu_j^+ e^{w_j^t/a_j} < \mu_j^- e^{-w_j^t/a_j} - \lambda \end{cases}$
    $\boldsymbol{w}^{t+1} = \boldsymbol{w}^t + \boldsymbol{\delta}^t$
```

Figure 1: AdaBoost for ℓ_1-regularized log-loss.

2. AdaBoost with ℓ_1 regularization

In this section we describe our ℓ_1 infused modification to AdaBoost using the general framework developed in Collins et al. (2002). In this framework, the weight update taken on each round of boosting is based on a template that selects and amortizes the update over (possibly) multiple features. The pseudocode for the algorithm is given in Fig. 1. On each round t of boosting an importance weight q^t is calculated for each example. These weights are simply the probability the current weight vector $\boldsymbol{w}^t$ assigns to the incorrect label for example i, and they are identical to the distribution defined over examples in the standard AdaBoost algorithm for the log-loss.

Let us defer the discussion on the use of templates to the end of Sec. 8 and assume that the template $\boldsymbol{a}$ simply selects a single feature to examine, i.e. $a_j > 0$ for some index j and $a_k = 0$ for all $k \neq j$. Once a feature is selected we compute its correlation and inverse correlation with the label according to the distrubution q^t, denoted by the variables μ_j^+ and μ_j^-. These correlations are also calculated by AdaBoost. The major difference is the computation of the update to the weight j, denoted δ_j^t. The value of δ_j^t of standard confidence-rated AdaBoost is $\frac{1}{2}\log(\mu_j^+/\mu_j^-)$, while our update incorporates ℓ_1-regularization with a multiplier λ. However, if we set $\lambda = 0$, we obtain the weight update of AdaBoost. We describe a derivation of the updates of for AdaBoost with ℓ_1-regularization that constitutes the algorithm described in Fig. 1. While we present a complete derivation in this section, the algorithm can be obtained as a special case of the analysis presented in Sec. 3. The later analysis is rather lengthy and detailed, however, and we thus provide a concrete and simple analysis for the important case of ℓ_1 regularization.

We begin by building on existing analyses of AdaBoost and show that each round of boosting is guaranteed to decrease the penalized loss. In the generalized version of boosting, originally described by 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 features, or base hypotheses, whose weight we update. Moreover, the template vector can may specify a different budget for each feature update so long as the vector $\boldsymbol{a}$ satisfies the condition $\sum_j a_j |x_{i,j}| \le 1$. Classical boosting sets a single coordinate in the vector $\boldsymbol{a}$ to a non-zero value, while the simultaneous update described in Collins et al. (2002) sets all coordinates of $\boldsymbol{a}$ to be the same. We start by recalling the progress bound for AdaBoost with the log-loss when using a template vector.

Lemma 1 (Boosting progress bound (Collins et al., 2002)) *Define importance weights $q^t(i) = 1/(1+\exp(y_i \boldsymbol{w}^t \cdot \boldsymbol{x}_i))$ and correlations*

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

Let $\boldsymbol{w}^{t+1} = \boldsymbol{w}^t + \boldsymbol{\delta}^t$ such that $\delta_j^t = a_j d_j^t$ and the vector $\boldsymbol{a}$ satisfies $\sum_j a_j |x_{i,j}| \le 1$. 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

$$\Delta_t \ge \sum_{j=1}^n a_j \left(\mu_j^+ \left(1 - e^{-d_j^t}\right) + \mu_j^- \left(1 - e^{d_j^t}\right)\right) = \sum_{j:a_j>0} a_j \left(\mu_j^+ \left(1 - e^{-\delta_j^t/a_j}\right) + \mu_j^- \left(1 - e^{\delta_j^t/a_j}\right)\right) \ .$$

For convenience and completeness, we provide a derivation of the above lemma using the notation established in this paper in section A of the appendix. 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

$$Q(\boldsymbol{w}^t) - Q(\boldsymbol{w}^{t+1}) \ge \sum_{j=1}^n a_j \left(\mu_j^+ \left(1 - e^{-\delta_j^t/a_j}\right) + \mu_j^- \left(1 - e^{\delta_j^t/a_j}\right)\right) - \lambda\|\boldsymbol{\delta}^t + \boldsymbol{w}^t\|_1 + \lambda\|\boldsymbol{w}^t\|_1 \ . \quad (5)$$

By construction, Eq. (5) is additive in j, so long as $a_j > 0$. We can thus maximize the progress individually for each such index j,

$$a_j\mu_j^+ \left(1 - e^{-\delta_j^t/a_j}\right) + a_j\mu_j^- \left(1 - e^{\delta_j^t/a_j}\right) - \lambda\left|\delta_j^t + w_j^t\right| + \lambda\left|w_j^t\right| \ . \quad (6)$$

Omitting the index j and eliminating constants, we are left with the following minimization problem in δ:

$$\underset{\delta}{\text{minimize}} \ a\mu^+ e^{-\delta/a} + a\mu^- e^{\delta/a} + \lambda\left|\delta + w\right| \ . \quad (7)$$

We now give two lemmas that aid us in finding the $\delta^\star$ minimizing Eq. (7).

Lemma 2 *If $\mu^+ e^{w/a} - \mu^- e^{-w/a} > 0$, then the minimizing $\delta^\star$ of Eq. (7) satisfies $\delta^\star + w \ge 0$. Likewise, if $\mu^+ e^{w/a} - \mu^- e^{-w/a} < 0$, then $\delta^\star + w \le 0$.*

Proof Without loss of generality, we focus on the case when $\mu^+ e^{w/a} > \mu^- e^{-w/a}$. Assume for the sake of contradiction that $\delta^\star + w < 0$. We now take the derivative of Eq. (7) with respect to δ, bearing in mind that the derivative of $|\delta + w|$ is -1. Equating the result to zero, we have

$$-\mu^+ e^{-\delta^\star/a} + \mu^- e^{\delta^\star/a} - \lambda = 0 \quad \Rightarrow \quad \mu^+ e^{-\delta^\star/a} \le \mu^- e^{\delta^\star/a}.$$

We assumed, however, that $\delta^\star + w < 0$, so that $e^{(\delta^\star+w)/a} < 1$ and $e^{-(\delta^\star+w)/a} > 1$. Thus, multiplying the left side of the above inequality by $e^{(\delta^\star+w)/a} < 1$ and the right by $e^{-(\delta^\star+w)/a} > 1$, we have that

$$\mu^+ e^{-\delta^\star/a} e^{\delta^\star/a + w/a} < \mu^- e^{\delta/a} e^{-\delta/a - w/a} \quad \Rightarrow \quad \mu^+ e^{w/a} < \mu^- e^{-w/a}.$$

This is a contradiction, so we must have that $\delta^\star + w \ge 0$. The proof for the symmetric case follows similarly. ■

In light of Lemma 2, we can eliminate the absolute value from the term $|\delta + w|$ in Eq. (7) and force $\delta + w$ to take a certain sign. This property helps us in the proof of our second lemma.

Lemma 3 *The optimal solution of equation Eq. (7) with respect to δ is $\delta^\star = -w$ if and only if $\left|\mu^+ e^{w/a} - \mu^- e^{-w/a}\right| \le \lambda$.*

Proof Again, let $\delta^\star$ denote the optimal solution of Eq. (7). Based on Lemma 2, we focus without loss of generality on the case where $\mu^+ e^{w/a} > \mu^- e^{-w/a}$, which allows us to remove the absolute value from Eq. (7). We replace it with $\delta + w$ and add the constraint that $\delta + w \ge 0$, yielding the following scalar optimization problem:

$$\underset{\delta}{\text{minimize}}\ a\mu^+ e^{-\delta/a} + a\mu^- e^{\delta/a} + \lambda(\delta + w) \quad \text{s.t.} \quad \delta + w \ge 0 \ \ .$$

The Lagrangian of the above problem is $\mathcal{L}(\delta, \beta) = a\mu^+ e^{-\delta/a} + a\mu^- e^{\delta/a} + \lambda(\delta + w) - \beta(\delta + w)$. To find the Lagrangian's saddle point for δ we take its derivative and obtain $\frac{\partial}{\partial \delta}\mathcal{L}(\delta, \beta) = -\mu^+ e^{-\delta/a} + \mu^- e^{\delta/a} + \lambda - \beta$. Let us first suppose that $\mu^+ e^{w/a} - \mu^- e^{-w/a} \le \lambda$. If $\delta^\star + w > 0$ (so that $\delta^\star > -w$), then by the complementarity conditions for optimality (Boyd and Vandenberghe, 2004), $\beta = 0$, hence,

$$0 = -\mu^+ e^{-\delta^\star/a} + \mu^- e^{\delta^\star/a} + \lambda > -\mu^+ e^{w/a} + \mu^- e^{-w/a} + \lambda \ \ .$$

This implies that $\mu^+ e^{w/a} - \mu^- e^{-w/a} > \lambda$, a contradiction. Thus, we must have that $\delta^\star + w = 0$. To prove the other direction we assume the that $\delta^\star = -w$. Then, since $\beta \ge 0$ we get $-\mu^+ e^{w/a} + \mu^- e^{-w/a} + \lambda \ge 0$, which implies that $\lambda \ge \mu^+ e^{w/a} - \mu^- e^{-w/a} \ge 0$, as needed. The proof for the symmetric case is analogous. ■

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 $\delta_j^\star \ne -w_j$ and $\delta_j^\star + w_j > 0$. We need to solve the following equation:

$$-\mu_j^+ e^{-\delta_j/a_j} + \mu_j^- e^{\delta_j/a_j} + \lambda = 0 \quad \text{or} \quad \mu_j^- \beta^2 + \lambda\beta - \mu_j^+ = 0 \ \ ,$$

where $\beta = e^{\delta_j/a_j}$. Since β is strictly positive, it is equal to the positive root of the above quadratic, yielding

$$\beta = \frac{-\lambda + \sqrt{\lambda^2 + 4\mu_j^+\mu_j^-}}{2\mu_j^-} \quad \Rightarrow \quad \delta_j^\star = a_j \log \frac{-\lambda + \sqrt{\lambda^2 + 4\mu_j^+\mu_j^-}}{2\mu_j^-}. \tag{8}$$

In the symmetric case, when $\delta_j^\star + w_j^t < 0$, we get that $\delta_j^\star = a_j \log \frac{\lambda + \sqrt{\lambda^2 + 4\mu_j^+\mu_j^-}}{2\mu_j^-}$. Finally, when the absolute value of the difference between $\mu_j^+ \exp(w_j^t/a_j)$ and $\mu_j^- \exp(-w_j^t/a_j)$ is less than or equal to λ, Lemma 3 implies that $\delta_j^\star = -w_j^t$. When one of μ_j^+ or μ_j^- is zero but $|\mu_j^+ e^{w_j^t/a_j}| > \lambda$ of $|\mu_j^- e^{-w_j^t/a_j}| > \lambda$, the solution is simpler. If $\mu_j^- = 0$, then $\delta_j^\star = a_j \log(\mu_j^+/\lambda)$, and when $\mu_j^+ = 0$, then $\delta_j^\star = a_j \log(\lambda/\mu_j^-)$. Omitting for clarity the cases when $\mu_j^\pm = 0$, the different cases constitute the core of the update given in Fig. 1.

3. Incorporating ℓ_∞ regularization

Here we begin to lay the framework for multitask and multiclass boosting with mixed norm regularizers. In particular, we derive boosting-style updates for the multitask and multiclass losses of equations (3) and (4). The derivation in fact encompasses the derivation of the updates for ℓ_1-regularized AdaBoost, which we also discuss below.

Before we can derive updates for boosting, we make a digression to consider a more general framework of minimizing a separable function with ℓ_∞ regularization. In particular, the problem that we solve assumes that we have a sum of one-dimensional, convex, bounded below and differentiable functions $f_j(d)$ (where we assume that if $f_j'(d) = 0$, then d is uniquely determined) plus an ℓ_∞-regularizer. That is, we want to solve

$$\underset{\boldsymbol{d}}{\text{minimize}} \sum_{j=1}^{k} f_j(d_j) + \lambda \|\boldsymbol{d}\|_\infty \quad . \tag{9}$$

The following theorem characterizes the solution $\boldsymbol{d}^\star$ of Eq. (9), and it also allows us to develop efficient algorithms for solving particular instances of Eq. (9). In the theorem, we allow the unconstrained minimizer of $f_j(d_j)$ to be infinite, and we use the shorthand $[k]$ to mean $\{1, 2, \ldots, k\}$ and $[z]_+ = \max\{z, 0\}$. The intuition behind the theorem and its proof is that we can move the values of $\boldsymbol{d}$ in their negative gradient directions together in a block, freezing entries of $\boldsymbol{d}$ that satisfy $f_j'(d_j) = 0$, until the objective in Eq. (9) begins to increase.

Theorem 4 *Let $\tilde{d}_j$ satisfy $f_j'(\tilde{d}_j) = 0$. The optimal solution $\boldsymbol{d}^\star$ of Eq. (9) satisfies the following properties:*

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

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

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

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

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

(c) When $\boldsymbol{d}^\star \ne \boldsymbol{0}$, $\sum_{j=1}^k |f_j'(d_j^\star)| = \sum_{j \in B} |f_j'(d_j^\star)| = \lambda$.

Proof Before proving the particular statements (i), (ii), and (iii) above, we perform a few preliminary calculations with the Lagrangian of Eq. (9) that will simplify the proofs of the various parts. Minimizing $\sum_j f_j(d_j) + \lambda \|\boldsymbol{d}\|_\infty$ is equivalent to solving the following problem:

$$\begin{array}{rl} \text{minimize} & \sum_{j=1}^k f_j(d_j) + \lambda \xi \\ \text{s.t.} & -\xi \le d_j \le \xi \;\; \forall j, \text{ and } \xi \ge 0 \quad . \end{array} \tag{10}$$

Although the positivity constraint on ξ is redundant, it simplifies the proof. To solve Eq. (10), we introduce Lagrange multiplier vectors $\boldsymbol{\alpha} \succeq 0$ and $\boldsymbol{\beta} \succeq 0$ for the constraints that $-\xi \le d_j \le \xi$ and the multiplier $\gamma \ge 0$ for the non-negativity of ξ. This gives the Lagrangian

$$\mathcal{L}(\boldsymbol{d}, \xi, \boldsymbol{\alpha}, \boldsymbol{\beta}, \gamma) = \sum_{j=1}^k f_j(d_j) + \lambda \xi + \sum_{j=1}^k \alpha_j (d_j - \xi) + \sum_{j=1}^k \beta_j (-d_j - \xi) - \gamma \xi \quad . \tag{11}$$

To find the saddle point of Eq. (11), we take the infimum of $\mathcal{L}$ with respect to ξ, which is $-\infty$ unless

$$\lambda - \sum_{j=1}^k \alpha_j - \sum_{j=1}^k \beta_j - \gamma = 0 \quad . \tag{12}$$

The non-negativity of γ implies that $\sum_{j=1}^k \alpha_j + \beta_j \le \lambda$. Complimentary slackness (Boyd and Vandenberghe, 2004) then shows that if $\xi > 0$ at optimum, $\gamma = 0$ so that $\sum_{j=1}^k \beta_j + \alpha_j = \lambda$.

We are now ready to start proving assertion (i) of the theorem. By taking derivatives of $\mathcal{L}$ to find the saddle point of the primal-dual problem, we know that at the optimal point $\boldsymbol{d}^\star$ the following equality holds:

$$f_j'(d_j^\star) - \beta_j + \alpha_j = 0 \quad . \tag{13}$$

Suppose that $\boldsymbol{d}^\star = \mathbf{0}$ so that the optimal $\xi = 0$. By Eq. (12), Eq. (13), and complimentary slackness on $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ we have

$$\sum_{j=1}^{k} |f_j'(0)| = \sum_{j=1}^{k} |\alpha_j - \beta_j| \le \sum_{j=1}^{k} \alpha_j + \beta_j \le \lambda \quad .$$

This completes the proof of the first direction of (i). We now prove the converse. Suppose that $\sum_{j=1}^{k} |f_j'(0)| \le \lambda$. In this case, if we set $\alpha_j = \left[-f_j'(0)\right]_+$ and $\beta_j = \left[f_j'(0)\right]_+$, we have $|f_j'(0)| = \beta_j + \alpha_j$ and $\sum_{j=1}^{k} \alpha_j + \beta_j \le \lambda$. Simply setting $\xi = 0$ and letting $\gamma = \lambda - \sum_{j=1}^{k}(\alpha_j + \beta_j) \ge 0$, we have $\sum_{j=1}^{k} \alpha_j + \beta_j + \gamma = \lambda$. The KKT conditions for the problem are thus satisfied and $\mathbf{0}$ is the optimal solution. This proves part (i) of the theorem.

We next prove statement (ii) of the theorem. We begin by proving that if $f_j'(0) \le 0$, then $d_j^\star \ge 0$ and similarly, that if $f_j'(0) \ge 0$, $d_j^\star \le 0$. If $f_j'(0) = 0$, we could choose $d_j^\star = 0$ without incurring any penalty in the ℓ_∞ norm. Thus, suppose that $f_j'(0) < 0$. Then, we have $f_j(-\delta) > f_j(0)$ for all $\delta > 0$, since the derivative of a convex function is non-decreasing. As such, the optimal setting of d_j must be non-negative. The argument for the symmetric case is similar, and we have shown that $f_j'(0) d_j^\star \le 0$.

The second half of (ii) is derived similarly. We know from Eq. (13) that $f_j'(d_j^\star) = \beta_j - \alpha_j$. If $f_j'(0) < 0$, the complimentary slackness condition implies that $\beta_j = 0$ and $\alpha_j \ge 0$. These two properties together imply that $f_j'(d_j^\star) \le 0$ whenever $f_j'(0) < 0$. An analogous argument is true when $f_j'(0) > 0$, which implies that $f_j'(d_j^\star) \ge 0$. Combining these properties with the statements of previous paragraph, we have that $f_j'(d_j^\star) d_j^\star \le 0$. This completes the proof of part (ii) of the theorem.

Let us now consider part (iii) of the theorem. If $\boldsymbol{d}^\star = \mathbf{0}$, then the set U is empty, thus (a), (b), and (c) are trivially satisfied. For the remainder of the proof, we assume that $\boldsymbol{d}^\star \ne \mathbf{0}$. In this case we must have $\xi > 0$, and complimentary slackness guarantees that $\gamma = 0$ and that at most one of α_j and β_j are nonzero. Consider the indices in U, that is, j such that $-\xi < d_j^\star < \xi$. For any such index j, both α_j and β_j must be zero by complimentary slackness. Therefore, from Eq. (13) we know $f_j'(d_j^\star) = 0$. The assumption that $\tilde{d}_j$ is the unique minimizer of f_j means that $d_j^\star = \tilde{d}_j$, i.e. coordinates not at the bound ξ simply take their unconstrained solutions. This proves point (a) of part (iii).

We now consider point (b) of statement (iii) from the theorem. For $j \in B$, we have $|\tilde{d}_j| \ge \xi$. Otherwise we could take $d_j^\star = \tilde{d}_j$ and have $|\tilde{d}_j| < \xi$. This clearly decreases the objective for f_j because $f_j(\tilde{d}_j) < f_j(d_j^\star)$. Further, $|\tilde{d}_j| < \xi$ so the constraints on ξ would remain intact, and we would have $j \notin B$. We therefore must have $|\tilde{d}_j| \ge \xi = |d_j^\star|$, which finishes the proof of part (b).

Finally, we arrive at part (c) of statement (iii). Applying Eq. (12) with $\gamma = 0$, we have $\lambda = \sum_{j=1}^{k} \alpha_j + \beta_j$, and applying Eq. (13) gives

$$\sum_{j=1}^{k} |f_j'(d_j^\star)| = \sum_{j=1}^{k} |\beta_j - \alpha_j| = \sum_{j=1}^{k} \alpha_j + \beta_j = \lambda \quad . \tag{14}$$

This proves point (c) for the first of the two sums. Next, we divide the solutions $d_j^\star$ into the two sets B (for bounded) and U (for unbounded), where

$$B = \{j : |d_j^\star| = \xi = \|\boldsymbol{d}^\star\|_\infty\} \quad \text{and} \quad U = \{j : |d_j^\star| < \xi\}.$$

Clearly, B is non-empty, since otherwise we could decrease ξ in the objective from Eq. (10). Recalling that for $j \in U$, $f_j'(d_j^\star) = 0$, Eq. (14) implies that the second equality of point (c) holds, namely

$$\sum_{j=1}^{k} |f_j'(d_j^\star)| = \sum_{j \in B} |f_j'(d_j^\star)| + \sum_{j \in U} \underbrace{|f_j'(d_j^\star)|}_{=0} = \lambda \quad .$$

```
INPUT: Convex functions $\{f_r\}_{r=1}^k$, regularization $\lambda$
IF $\sum_{r=1}^k |f_r'(0)| \le \lambda$
    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 $\tilde{d}_{(1)} \ge \tilde{d}_{(2)} \ge \ldots \ge \tilde{d}_{(k)}$
SORT $\{\tilde{d}_r\}$ (descending) into $\{\tilde{d}_{(r)}\}$; $\tilde{d}_{(k+1)} = 0$
FOR $l = 1$ to $k$
    SOLVE for $\xi$ such that $\sum_{i=1}^l f_{(i)}'(s_i\xi) = -\lambda$
    IF $\xi \ge \tilde{d}_{(l+1)}$
      BREAK
RETURN $\boldsymbol{d}^\star$ such that $d_r^\star = s_r \min\{\tilde{d}_r, \xi\}$
```

Figure 2: Algorithm for minimizing $\sum_r f_r(d_r) + \lambda \|\boldsymbol{d}\|_\infty$.

■

In Fig. 2, we present a general algorithm that builds directly on the implications of Thm. 4 for finding the minimum of $\sum_j f_j(d_j)+\lambda \|\boldsymbol{d}\|_\infty$. The algorithm begins by flipping signs so that all $d_j \ge 0$ (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) of the theorem). When the algorithm finds a set B and bound $\xi = \|\boldsymbol{d}\|_\infty$ so that part (iii) of the theorem is satisfied (which is guaranteed by (iii.b), since all indices in the bounded set B must have unregularized solutions greater than the bound $\|\boldsymbol{d}^\star\|_\infty$), it terminates. Note that the algorithm naïvely has runtime complexity of $O(k^2)$. In the following sections, we show that this complexity can be brought down to $O(k \log k)$ by exploiting the structure of the functions f_j that we consider (this could be further decreased to $O(k)$ using generalized median searches, however, this detracts from the main focus of this paper).

Revisiting AdaBoost with ℓ_1-regularization We can now straightforwardly derive lemmas 2 and 3 as special cases of Theorem 4. Recall the ℓ_1-regularized minimization problem we faced for the exponential loss in Eq. (7): 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

$$a\mu^+ e^{w/a} e^{-\theta/a} + a\mu^- e^{-w/a} e^{\theta/a} + \lambda|\theta|.$$

The above problem amounts to a simple one-dimensional version of the type of problem considered in Thm. 4. We take the derivative of the first two terms with respect to θ, which at $\theta = 0$ is $-\mu^+e^{w/a} + \mu^-e^{-w/a}$. If this term is positive, then $\theta^\star \le 0$, otherwise, $\theta^\star \ge 0$. This result is equivalent to the conditions for $\delta^\star + w \le 0$ and $\delta^\star + w \ge 0$ in Lemma 2. Lemma 3 can also be obtained as an immediate corollary of Thm. 4 since $\theta^\star = 0$ if and only if $|\mu^+e^{w/a} - \mu^-e^{-w/a}| \le \lambda$, which implies that $w + \delta^\star = 0$. The solution in Eq. (8) amounts to finding the θ such that $|\mu^+e^{w/a}e^{-\theta/a} - \mu^-e^{-w/a}e^{\theta/a}| = \lambda$.

4. AdaBoost with ℓ_1/ℓ_∞ mixed-norm regularization

In this section we build on the results presented in the previous sections and present extensions of the AdaBoost algorithm to multitask and multiclass problems with mixed-norm regularization given in Eq. (3) and Eq. (4).

We start with a bound on the logistic loss to derive boosting-style updates for the mixed-norm *multitask* loss of Eq. (3) with $p = \infty$. To remind the reader, the loss we consider is

$$Q(W) = L(W) + \lambda R(W) = \sum_{i=1}^{m} \sum_{r=1}^{k} \log\left(1 + \exp(-y_{i,r}(\boldsymbol{w}_r \cdot \boldsymbol{x}_i))\right) + \lambda \sum_{j=1}^{n} \|\overline{\boldsymbol{w}}_j\|_\infty \quad .$$

In the above, $\boldsymbol{w}_r$ represents the r^{th} column of W while $\overline{\boldsymbol{w}}_j$ represents the j^{th} row. In order to extend the boosting algorithm from Fig. 1, we first need to extend the progress bounds for binary logistic regression to the multitask objective.

The multitask loss is decomposable into sums of losses, one per task. Hence, for each separate task we obtain the same bound as that of Lemma 1. However, we now must 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 $q^t(i,r) = 1/(1 + \exp(y_{i,r}\boldsymbol{w}_r \cdot \boldsymbol{x}_i))$ and the correlations $\mu^{\pm}_{r,j}$ for each task r through a simple generalization of the correlations in Lemma 1 as

$$\mu^+_{r,j} = \sum_{i:y_{i,r}x_{i,j}>0} q^t(i,r)|x_{i,j}| \quad \text{and} \quad \mu^-_{r,j} = \sum_{i:y_{i,r}x_{i,j}<0} q^t(i,r)|x_{i,j}| \quad .$$

Defining $\boldsymbol{\delta}_j = [\delta_{j,1} \cdots \delta_{j,k}]$ and applying lemma 1 yields that when we update $W^{t+1} = W^t + [\boldsymbol{\delta}^t_1 \cdots \boldsymbol{\delta}^t_n]^\top$, we can lower bound the change in the loss $\Delta_t = L(W^{t+1}) - L(W^t)$ between iterations t and $t+1$ by

$$\Delta_t \geq \sum_{j=1}^{n} a_j \sum_{r=1}^{k} \left(\mu^+_{r,j}\left(1 - e^{-\delta^t_{j,r}/a_j}\right) + \mu^-_{r,j}\left(1 - e^{\delta^t_{j,r}/a_j}\right)\right) \quad . \tag{15}$$

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

Before proceeding to the specifics of the algorithm, we revisit the multiclass objective from Eq. (4) and bound it as well. Again, we have

$$Q(W) = L(W) + \lambda R(W) = \sum_{i=1}^{m} \log\left(1 + \sum_{r \neq y_i} \exp(\boldsymbol{w}_r \cdot \boldsymbol{x}_i - \boldsymbol{w}_{y_i} \cdot \boldsymbol{x}_i)\right) + \lambda \sum_{j=1}^{n} \|\overline{\boldsymbol{w}}_j\|_\infty \quad .$$

A change in the definition of the importance weights to be the probability that the weight matrix W assigns to a particular class, along with an updated definition of the correlations $\mu^{\pm}_{r,j}$, gives us a bound on the change in the multiclass loss. In particular, the following lemma modifies and extends the multiclass boosting bounds of Collins et al. (2002).

Lemma 5 (Multiclass boosting progress bound) *Let $q^t(i,r)$ denote importance weights for each example index i and class index $r \in \{1, \ldots, k\}$, where*

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

Define the importance-weighted correlations as

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

Let the update to the j^{th} row of the matrix W^t is $\overline{\boldsymbol{w}}^{t+1}_j = \overline{\boldsymbol{w}}^t_j + \boldsymbol{\delta}^t_j$ and that the vector $\boldsymbol{a}$ satisfies $\max_i 2\sum_j a_j|x_{i,j}| \leq 1$. *Then the change in the multiclass logistic loss, $\Delta_t = L(W^{t+1}) - L(W^t)$, is* *lower bounded by*

$$\Delta_t \geq \sum_{j=1}^{n} a_j \sum_{r=1}^{k} \left(\mu_{r,j}^{+} \left(1 - e^{-\delta_{j,r}^t / a_j}\right) + \mu_{r,j}^{-} \left(1 - e^{\delta_{j,r}^t / a_j}\right) \right) \quad . \tag{16}$$

The proof of the lemma is provided in Appendix A. The standard boosting literature usually assumes that instead of having multiple class vectors $\boldsymbol{w}_r$, there is a single vector $\boldsymbol{w}$ and that features have different outputs per-class, denoted $\boldsymbol{h}(\boldsymbol{x}_i, r)$. Our setting instead allows single view of $\boldsymbol{x}_i$ with multiple prediction vectors $\boldsymbol{w}_r$, arguably a more natural (and common) parameterization. With the lemma in place, we note that the boosting bounds we have achieved for the multitask loss in Eq. (15) and the multiclass loss in Eq. (16) are syntactically identical. They differ only in their computation of the importance weights and weighted correlations. These similarities allow us to attack the boosting update for both together, deriving one efficient algorithm for ℓ_1/ℓ_∞-regularized boosting based on a corollary to Theorem 4 and the algorithm of Fig. 2.

Now we proceed with the derivation of AdaBoost with structural regularization. Adding ℓ_∞-regularization terms to Eq. (15) and Eq. (16), we have

$$\begin{aligned} Q(W^t) - Q(W^{t+1}) \quad \geq \quad & \sum_{j=1}^{n} a_j \sum_{r=1}^{k} \left(\mu_{r,j}^{+} \left(1 - e^{-\delta_{j,r} / a_j}\right) + \mu_{r,j}^{-} \left(1 - e^{\delta_{j,r} / a_j}\right) \right) \\ & - \lambda \sum_{j=1}^{n} \left\| \overline{\boldsymbol{w}}_j^t + \boldsymbol{\delta}_j \right\|_\infty + \lambda \sum_{j=1}^{n} \left\| \overline{\boldsymbol{w}}_j^t \right\|_\infty \quad . \end{aligned} \tag{17}$$

For simplicity of our derivation, we focus on updating a single row j in W, and we temporarily assume that $\overline{\boldsymbol{w}}_j^t = \boldsymbol{0}$. We make the substitution $a_j d_r = \delta_{j,r}$. Using these assumptions and a few simple algebraic manipulations, we find that we need to solve the following minimization problem:

$$\underset{\boldsymbol{d}}{\text{minimize}} \sum_{r=1}^{k} \mu_r^{+} e^{-d_r} + \mu_r^{-} e^{d_r} + \lambda \left\| \boldsymbol{d} \right\|_\infty \quad . \tag{18}$$

First, we note that the objective function of Eq. (18) is separable in $\boldsymbol{d}$ with an ℓ_∞-regularization term. Second, the derivative of each of the terms, dropping the regularization term, is $-\mu_r^{+} e^{-d_r} + \mu_r^{-} e^{d_r}$. Third, the un-regularized 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 Theorem 4:

Corollary 6 *The optimal solution $\boldsymbol{d}^\star$ of Eq. (18) is $\boldsymbol{d}^\star = \boldsymbol{0}$ if and only if $\sum_{r=1}^{k} |\mu_r^{+} - \mu_r^{-}| \leq \lambda$. Assume without loss of generality that $\mu_r^{+} \geq \mu_r^{-}$. When $\boldsymbol{d}^\star \neq \boldsymbol{0}$, 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 above, 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. 2) to find $\tilde{d}_{(\rho)}$, where $\tilde{d}_{(\rho)}$ is the ρ^{th} largest unregularized solution. The algorithm then solves the following equation in d for each $\rho \in [k]$,

$$e^{-d} \sum_{r : \tilde{d}_r \geq \tilde{d}_{(\rho)}} \mu_r^{+} \; - \; e^{d} \sum_{r : \tilde{d}_r \geq \tilde{d}_{(\rho)}} \mu_r^{-} - \lambda = 0 \quad . \tag{19}$$

This process continues until we find an index ρ such that the solution $d^\star$ for Eq. (19) satisfies $d^\star \geq \tilde{d}_{(\rho+1)}$. To develop an efficient algorithm, we define

$$M_\rho^\pm = \sum_{r:\tilde{d}_r \geq \tilde{d}_{(\rho)}} \mu_r^\pm \ .$$

In order to solve Eq. (19) for each d, we can apply the reasoning for Eq. (8) and find

$$d^\star = \log \frac{-\lambda + \sqrt{\lambda^2 + 4M_\rho^+ M_\rho^-}}{2M_\rho^-} \ . \tag{20}$$

When $M_\rho^- = 0$, we get $d^\star = \log(\lambda/M_\rho^+)$. We can use Eq. (20) successively in the algorithm of Fig. 2 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. 2 to solve the multiclass and multitask extensions of AdaBoost with ℓ_1/ℓ_∞-regularization.

It remains to show how to solve the more general update from Eq. (17). In particular, we would like to find the minimum of

$$a\sum_{r=1}^{k}\left(\mu_r^+ e^{-d_r} + \mu_r^- e^{d_r}\right) + \lambda \left\|\boldsymbol{w} + a\boldsymbol{d}\right\|_\infty \ . \tag{21}$$

We can make the transformation $\gamma_r = w_r/a + d_r$, which reduces our problem to 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 as in the prequel, and we then recover $\boldsymbol{d}^\star = \boldsymbol{\gamma}^\star - \boldsymbol{w}/a$.

Combining our reasoning for the multiclass and multitask losses, we obtain an algorithm that solves both problems by appropriately setting $\mu_{r,j}^\pm$ while using the algorithm of Fig. 2. The combined algorithm for both problems is presented in Fig. 3.

5. Gradient Boosting with ℓ_1 regularization

We have presented thus far algorithms and analyses that are based on the original AdaBoost algorithm. In this section we shift our attention to a lesser known approach and derive additive updates for the logistic-loss based on quadratic upper bounds. The bounding techniques we use are based on those described by Dekel et al. (2005). We follow popular naming conventions and term these methods GradBoost for their use of gradients and bounds on the Hessian. We would like to note that the resulting algorithms do not necessarily entertain the weak-to-strong learnability properties of AdaBoost. Generally speaking, our methods can be view as guarded (block) coordinate descent methods for binary and multivariate logistic losses. GradBoost gives a different style of updates, which in some cases may be faster than the AdaBoost updates, while still enjoying the same feature induction benefits and convergence guarantees as the previous algorithms (see the later sections 8 and 9). Furthermore, the use of the quadratic bounds on the logistic functions allows us to perform boosting-style steps with ℓ_2 and ℓ_2^2 regularization in addition to the ℓ_1 and ℓ_∞ regularization studied above. 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 is chosen to upper bound $\nabla^2 L(\boldsymbol{w})$. In the binary case, for example, $D = \mathrm{diag}(1/4\sum_{i=1}^m x_{i,j}^2)$.

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

Figure 3: Boosting algorithm for solving ℓ_1/ℓ_∞-regularized multitask or multiclass logistic regression using Alg. 2

We first focus on the binary logistic loss of Eq. (2). GradBoost, similar to AdaBoost, uses a template vector $\boldsymbol{a}$ to parameterize updates to the weight vectors. We can thus perform updates ranging from a single feature, analogous to the AdaBoost algorithm, to all the features, in a manner similar to the parallel version of AdaBoost from Collins et al. (2002). The following lemma provides a quantitative statement on the amount of progress that GradBoost can make. We provide a proof of the lemma in Appendix B.

Lemma 7 (GradBoost Progress Bound) *Denote by $\boldsymbol{g}$ the gradient of the logistic loss so that $g_j = -\mu_j^+ + \mu_j^-$, where μ_j^+, μ_j^- are defined as in lemma 1. Assume that we are provided an update template $\boldsymbol{a} \in \mathbb{R}_+^n$ such that $\sum_{i,j} a_j x_{i,j}^2 \le 1$. Let $\boldsymbol{w}^{t+1} = \boldsymbol{w}^t + \boldsymbol{\delta}^t$ and $\delta_j^t = a_j d_j^t$. Then the change, $\Delta_t = L(\boldsymbol{w}^t) - L(\boldsymbol{w}^{t+1})$, in the logistic-loss between two consecutive iterations of GradBoost is lower bounded by*

$$\Delta_t \ge -\sum_{j=1}^n a_j \left(g_j d_j^t + \frac{(d_j^t)^2}{8}\right) = -\sum_{j: a_j > 0} \left(g_j \delta_j^t + \frac{(\delta_j^t)^2}{8a_j}\right) .$$

The assumption on $\boldsymbol{a}$ at a first glance seems strong, but is in practice no stronger than the assumption on the template vectors for AdaBoost in Lemma 1. AdaBoost requires that $\max_i \sum_{j=1}^n a_j |x_{i,j}| \le 1$, while GradBoost requires a bound on the average for each template vector. Indeed, had we used the average logistic loss, dividing $L(\boldsymbol{w})$ by m, and assumed that $\max_j |x_{i,j}| = 1$, AdaBoost and GradBoost would have shared the same set of possible update templates. Note that the minus sign in front of the bound also seems a bit odd upon a first look. However, we choose $\boldsymbol{\delta}^t$ in the *opposite* direction of $\boldsymbol{g}$ to decrease the logistic-loss substantially, as we show in the sequel.

To derive a usable bound for GradBoost with ℓ_1-regularization, we replace the progress bound in lemma 7 with a bound dependent on $\boldsymbol{w}^{t+1}$ and $\boldsymbol{w}^t$. Formally, we rewrite the bound by substituting $\boldsymbol{w}^{t+1} - \boldsymbol{w}^t$ for $\boldsymbol{\delta}^t$. Recalling Eq. (2) to incorporate ℓ_1-regularization, we get

$$Q(\boldsymbol{w}^{t+1}) - Q(\boldsymbol{w}^t) \leq \sum_{j:a_j>0} g_j(w_j^{t+1} - w_j^t) + \frac{1}{8}\sum_{j:a_j>0} \frac{1}{a_j}(w_j^{t+1} - w_j^t)^2 + \lambda \left\|\boldsymbol{w}^{t+1}\right\|_1 - \lambda \left\|\boldsymbol{w}^t\right\|_1 \quad . \tag{22}$$

As the bound in Eq. (22) is separable in each w_j, we can decompose it and minimize with respect to the individual entries of $\boldsymbol{w}$. Performing a few algebraic manipulations and letting w be short for w_j^{t+1}, we want to minimize

$$\left(g_j - \frac{w_j^t}{4a_j}\right) w + \frac{1}{8a_j} w^2 + \lambda |w| \quad . \tag{23}$$

The minimizer of Eq. (23) can be derived using Lagrange multipliers or subgradient calculus. Instead, we describe a solution that straightforwardly builds on Thm. 4.

First, note that when $a_j = 0$, we simply set $w_j^{t+1} = w_j^t$. We can thus focus on an index j such that $a_j > 0$. Multiplying Eq. (23) by $4a_j$, the quadratic form we need to minimize is

$$\frac{1}{2}w^2 + (4a_j g_j - w_j^t)w + 4a_j\lambda|w| \quad . \tag{24}$$

To apply Thm. 4, we take derivatives of the terms in Eq. (24) not involving $|w|$, which results in the term $w + 4a_j g_j - w_j^t$. At $w = 0$, this expression becomes $4a_j g_j - w_j^t$, thus Thm. 4 implies that $w^\star = -\operatorname{sign}(4a_j g_j - w_j^t)\alpha$ for some $\alpha \geq 0$. If $4a_j g_j - w_j^t \leq -4a_j\lambda$, then we solve the equation $w + 4a_j g_j - w_j^t + 4a_j\lambda = 0$ for $w^\star$, which gives $w^\star = w_j^t - 4a_j g_j - 4a_j\lambda$. Note that $w_j^t - 4a_j g_j = |w_j^t - 4a_j g_j|$. In the case $4a_j g_j - w_j^t \geq 4a_j\lambda$, the theorem indicates that $w^\star \leq 0$. Thus we solve $w + 4a_j g_j - w_j^t - 4a_j\lambda = 0$ for $w^\star = w_j^t - 4a_j g_j + 4a_j\lambda$. In this case, $|w_j^t - 4a_j g_j| = 4a_j g_j - w_j^t$, while $\operatorname{sign}(w_j^t - 4a_j g_j) \leq 0$. Combining the above reasoning, we obtain that the solution is simply a thresholded minimizer of the quadratic without ℓ_1-regularization,

$$w^\star = \operatorname{sign}(w_j^t - 4a_j g_j)\left[|w_j^t - 4a_j g_j| - 4a_j\lambda\right]_+ \quad . \tag{25}$$

We now can use Eq. (25) to derive a convergent GradBoost algorithm for the ℓ_1-regularized logistic loss. The algorithm is presented in Fig. 4. On each round of the algorithm a vector of importance weights, $q^t(i)$, is computed, a template vector $\boldsymbol{a}$ is chosen, and finally the update implied by Eq. (25) is performed.

6. GradBoost with ℓ_1/ℓ_∞ mixed-norm regularization

We now transition to the problem of minimizing the non-binary losses considered in Sec. 4. Specifically, in this section we describe ℓ_1/ℓ_∞ mixed-norm regularization for multitask and multiclass logistic losses. We begin by examining the types of steps GradBoost needs to take in order to minimize the losses. Our starting point is the quadratic loss with ℓ_∞-regularization. Specifically, we would like to minimize

$$\sum_{r=1}^{k} \frac{1}{2}a_r w_r^2 + b_r w_r + \lambda \left\|\boldsymbol{w}\right\|_\infty . \tag{26}$$

Omitting the regularization term, the minimizer of Eq. (26) is $\tilde{w}_r = \frac{-b_r}{a_r}$. Equipped with Eq. (26) and the form of its unregularized minimizer, we obtain the following corollary of Thm. 4.

```
INPUT:   Training set $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$;
         Update templates $\mathcal{A} \subseteq \mathbb{R}_+^n$ s.t. $\forall \boldsymbol{a} \in \mathcal{A}$, $\sum_{i,j} a_j x_{i,j}^2 \le 1$
         Regularization $\lambda$; number of rounds $T$
FOR $t = 1$ to $T$
    FOR $i = 1$ to $m$
        // Compute importance weights
        SET $q^t(i) = 1/(1 + \exp(y_i(\boldsymbol{x}_i \cdot \boldsymbol{w}^t)))$
    CHOOSE update template $\boldsymbol{a} \in \mathcal{A}$
        FOR $j$ s.t. $a_j \neq 0$
            // Compute gradient term
            SET $g_j = -\sum_{i=1}^m q^t(i) x_{i,j} y_i$
            // Compute new weight for parameter $j$
            $w_j^{t+1} = \mathrm{sign}(w_j^t - 4a_j g_j) \left[|w_j^t - 4a_j g_j| - 4a_j\lambda\right]_+$
```

Figure 4: GradBoost for the ℓ_1-regularized log-loss.

Corollary 8 *The optimal solution $\boldsymbol{w}^\star$ of Eq. (26) is $\boldsymbol{w}^\star = \boldsymbol{0}$ if and only if $\sum_{r=1}^k |b_r| \le \lambda$. Assume without loss of generality that $b_r \le 0$, so that $w_r^\star \ge 0$. Let $B = \{r : |w_r^\star| = \|\boldsymbol{w}^\star\|_\infty\}$ and $U = [k] \setminus B$, then*

(a) For all $r \in U$, $a_r w_r^\star + b_r = 0$, i.e. $w_r^\star = \tilde{w}_r = -b_r/a_r$

(b) For all $r \in B$, $\tilde{w}_r \ge w_r^\star = \|\boldsymbol{w}^\star\|_\infty$

(c) $\sum_{r \in B} a_r w_r^\star + b_r + \lambda = 0$.

Similar to our derivation of AdaBoost with ℓ_∞-regularization, we now describe an efficient procedure for finding the minimizer of the quadratic loss with ℓ_∞ regularization. First, we replace each b_r with its negative absolute value while recording the original sign. This change guarantees the non-negativity of the components in the solution vector. The procedure sorts the indices in $[k]$ by the magnitude of the unregularized solutions $\tilde{w}_r$. It then iteratively solves for w satisfying the following equality,

$$w \sum_{r:\tilde{w}_r \ge \tilde{w}_{(\rho)}} a_r - \sum_{r:\tilde{w}_r \ge \tilde{w}_{(\rho)}} b_r + \lambda = 0 \quad . \tag{27}$$

As in the AdaBoost case, we solve Eq. (27) for growing values of ρ until we find an index ρ for which the solution $w^\star$ satisfies the condition $w^\star \ge \tilde{w}_{(\rho+1)}$, where $\tilde{w}_{(\rho)}$ is the ρ^{th} largest unregularized solution. Analogous to the AdaBoost case, we define

$$A_\rho = \sum_{r:\tilde{w}_r \ge \tilde{w}_{(\rho)}} a_r \quad \text{and} \quad B_\rho = \sum_{r:\tilde{w}_r \ge \tilde{w}_{(\rho)}} b_r \quad .$$

These variables allow us to efficiently update the sums in Eq. (27), making possible to compute its solution in constant time. We can now plug the updates into the algorithm of Fig. 2 and get an efficient procedure for minimizing Eq. (26). We would like to note that we can further improve the run time of the search procedure and derive an algorithm has $O(k)$ expected runtime, rather than $O(k \log k)$ time. The linear algorithm is obtained by mimicking a randomized median search with side information. We omit the details as this improvement is a digression from the focus of the paper and is fairly straightforward to derive from the $O(k \log k)$ procedure.

We are left with the task of describing upper bounds for the multitask and multiclass losses of Eq. (3) and Eq. (4) that result in quadratic terms of the form given by Eq. (26). We start with the multitask loss.

Multitask GradBoost with regularization The multitask loss is decomposable into sums of losses, one per task. Thus, Lemma 7 provides the same bound for each task as that for the binary logistic loss. However, analogous to the bound derived for AdaBoost in Sec. 4, we need to redefine our importance weights and gradient terms as,

$$q^t(i,r) = \frac{1}{1+\exp(y_{i,r}(\boldsymbol{w}_r \cdot \boldsymbol{x}_i))} \quad \text{and} \quad g_{r,j} = -\sum_{i=1}^{m} q^t(i,r) y_{i,r} x_{i,j} \ .$$

Assuming as in lemma 7 that $\sum_{i,j} a_j x_{i,j}^2 \le 1$, we reapply the lemma and the bound used in Eq. (22) to get,

$$Q(W^{t+1}) - Q(W^t) \le \sum_{j:a_j>0} \left[\sum_{r=1}^{k} g_{r,j}(w_{j,r}^{t+1} - w_{j,r}^t) + \sum_{r=1}^{k} \frac{(w_{j,r}^{t+1} - w_{j,r}^t)^2}{8a_j} + \lambda \left\| \overline{\boldsymbol{w}}_j^{t+1} \right\|_\infty - \lambda \left\| \overline{\boldsymbol{w}}_j^t \right\|_\infty \right] \ . \tag{28}$$

The upper bound on $Q(W^{t+1})$ from Eq. (28) is a separable quadratic function with an ℓ_∞ regularization term. We therefore can use verbatim the procedure for solving Eq. (26).

Multiclass GradBoost with regularization We next consider the multiclass objective of Eq. (4). To derive a quadratic upper bound on the objective, we need to define per-class importance weights and gradient terms. As for multiclass AdaBoost, the importance weight $q^t(i,r)$ (for $r \in \{1,\ldots,k\}$) for a given example and class r is the probability that the current weight matrix W^t assigns to label r. The gradient as well is slightly different. Formally, we have gradient terms $g_{r,j} = \frac{\partial}{\partial w_{j,r}} L(W)$ defined by

$$q^t(i,r) = \frac{\exp(\boldsymbol{w}_r^t \cdot \boldsymbol{x}_i)}{\sum_s \exp(\boldsymbol{w}_s \cdot \boldsymbol{x}_i)} \quad \text{and} \quad g_{r,j} = \sum_{i=1}^{m} \left(q^t(i,r) - \mathbf{1}\{r = y_i\} \right) x_{i,j} \ . \tag{29}$$

Similar to the previous losses, we can derive a quadratic upper bound on the multiclass logistic loss. To our knowledge, as for multiclass AdaBoost, this also is a new upper bound on the multiclass loss.

Lemma 9 (Multiclass Gradboost Progress) *Define $g_{r,j}$ as in Eq. (29). Assume that we are provided an update template $\boldsymbol{a} \in \mathbb{R}_+^n$ such that $\sum_{i,j} a_j x_{i,j}^2 \le 1$. Let the update to row j of the matrix W^t be $\overline{\boldsymbol{w}}_j^{t+1} = \overline{\boldsymbol{w}}_j^t + \boldsymbol{\delta}_j^t$. Then the change in the multiclass logistic loss from Eq. (4) is lower bounded by*

$$L(W^t) - L(W^{t+1}) \ge -\sum_{j:a_j>0} \left(\sum_{r=1}^{k} g_{r,j} \delta_{j,r}^t + \frac{1}{4} \sum_{r=1}^{k} \frac{(\delta_{j,r}^t)^2}{a_j} \right) \ .$$

We prove the lemma in Appendix B. As in the case for the binary logistic loss, we typically set $\boldsymbol{\delta}_j$ to be in the opposite direction of the gradient $g_{r,j}$. We can replace $\delta_{j,r}^t$ from the lemma with $w_{j,r}^{t+1} - w_{j,r}^t$, which gives us the following upper bound:

$$\begin{aligned} L(W^{t+1}) \quad \le \quad & L(W^t) + \sum_{j:a_j>0} \sum_{r=1}^{k} \left(g_{r,j} - \frac{w_{j,r}^t}{2a_j} \right) w_{j,r}^{t+1} \\ & + \frac{1}{4} \sum_{j:a_j>0} \sum_{r=1}^{k} \frac{(w_{j,r}^{t+1})^2}{a_j} + \frac{1}{4} \sum_{j:a_j>0} \sum_{r=1}^{k} \frac{(w_{j,r}^t)^2}{a_j} - \sum_{j:a_j>0} \sum_{r=1}^{k} g_{r,j} w_{j,r}^t \ . \end{aligned} \tag{30}$$

```
INPUT:  Training set $S = \{(\boldsymbol{x}_i, \langle \boldsymbol{y}_i \rangle)\}_{i=1}^m$
        Regularization $\lambda$, number of rounds $T$;
        Update templates $\mathcal{A}$ s.t. $\forall \boldsymbol{a} \in \mathcal{A}$, $\sum_{i,j} a_j x_{i,j}^2 \le 1$
FOR $t = 1$ to $T$
    CHOOSE update template $\boldsymbol{a} \in \mathcal{A}$
    FOR $i = 1$ to $m$ and $r = 1$ to $k$
        // Compute importance weights for each task/class
        $q^t(i,r) = \begin{cases} \frac{\exp(\boldsymbol{w}_r \cdot \boldsymbol{x}_i)}{\sum_l \exp(\boldsymbol{w}_r \cdot \boldsymbol{x}_i)} & [\text{MC}] \\ \frac{1}{1+\exp(y_{i,r}(\boldsymbol{w}_r^t \boldsymbol{x}_i))} & [\text{MT}] \end{cases}$
    // Loop over rows of the matrix W
    FOR $j$ s.t. $a_j > 0$
        FOR $r = 1$ to $k$
            // Compute gradient and scaling terms
            $g_{r,j} = \begin{cases} \sum_{i=1}^m (q^t(i,r) - \mathbf{1}\{r = y_i\}) x_{i,j} & [\text{MC}] \\ -\sum_{i=1}^m q^t(i,r) x_{i,j} y_{i,r} & [\text{MT}] \end{cases}$
            $a = \begin{cases} a_j & [\text{MC}] \\ 2a_j & [\text{MT}] \end{cases}$
        // Compute new weights for the row $\overline{\boldsymbol{w}}_j^{t+1}$ using Alg. 2
        $\overline{\boldsymbol{w}}_j^{t+1} = \text{argmin}_{\boldsymbol{w}} \Big\{$
                $\frac{1}{4a} \sum_{r=1}^k w_r^2 + \sum_{r=1}^k \left(g_{r,j} - \frac{w_{j,r}^t}{2a}\right) w_r + \lambda \|\boldsymbol{w}\|_\infty \Big\}$
```

Figure 5: GradBoost for ℓ_1/ℓ_∞-regularized multitask and multiclass boosting

Adding ℓ_∞-regularization terms to Eq. (30), we can upper bound $Q(W^{t+1}) - Q(W^t)$ by

$$\begin{aligned} Q(W^{t+1}) - Q(W^t) \;\; \le & \sum_{j:a_j>0} \left[\sum_{r=1}^k \left(g_{r,j} - \frac{w_{j,r}^t}{2a_j} \right) w_{j,r}^{t+1} + \frac{1}{4} \sum_{r=1}^k \frac{(w_{j,r}^{t+1})^2}{a_j} + \lambda \left\| \overline{\boldsymbol{w}}_j^{t+1} \right\|_\infty \right. \\ & \left. - \sum_{r=1}^k g_{r,j} w_{j,r}^t + \frac{1}{4} \sum_{r=1}^k \frac{(w_{j,r}^t)^2}{a_j} - \lambda \left\| \overline{\boldsymbol{w}}_j^t \right\|_\infty \right] . \end{aligned} \tag{31}$$

As was the upper bound for the multitask loss, Eq. (31) is clearly a separable convex quadratic function with ℓ_∞ regularization for each row $\overline{\boldsymbol{w}}_j$ of W.

We conclude the section with the pseudocode of the unified GradBoost algorithm for the multitask and the multiclass losses of equations (3) and (4). Note that the upper bounds of Eq. (28) and Eq. (31) for both losses are almost identical. The sole difference between the two losses distills to the definition of the gradient $g_{r,j}$ terms and that in Eq. (31) the constant on a_j is half of that in Eq. (28). The algorithm is simple: it iteratively calculates the gradient terms $g_{r,j}$, then employs an update template $\boldsymbol{a} \in \mathcal{A}$ and calls the algorithm of Fig. 2 to minimize Eq. (31). The pseudocode of the algorithm is given in Fig. 5.

7. GradBoost with ℓ_1/ℓ_2 Regularization

One form of regularization that has rarely 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 decrease in the loss. Concretely, the coupling of the exponential terms with $\|\boldsymbol{w}\|^2$ or $\|\boldsymbol{w}\|$ leads to non-trivial minimization problems. GradBoost, however, can straightforwardly incorporate ℓ_2-based penalties, since it uses linear and quadratic bounds on the decrease in the loss rather than the exponential bounds of AdaBoost. In this section we focus on multiclass GradBoost. The modification for multitask or standard boosting is straightforward and follows the lines of derivation discussed 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 from Eq. (4),

$$Q(W) = \sum_{i=1}^{m} \log\left(1 + \sum_{r \neq y_i} \exp(\boldsymbol{w}_r \cdot \boldsymbol{x}_i - \boldsymbol{w}_{y_i} \cdot \boldsymbol{x}_i)\right) + \lambda \sum_{j=1}^{n} \|\bar{\boldsymbol{w}}_j\|_2 \quad . \tag{32}$$

Using the bounds from lemma 9 and Eq. (30) and the assumption that $\sum_{i,j} a_j x_{i,j}^2 \leq 1$ as before, we upper bound $Q(W^{t+1}) - Q(W^t)$ by

$$\sum_{j:a_j>0} \left[\sum_{r=1}^{k} \left(g_{r,j} - \frac{w_{j,r}^t}{2a_j}\right) w_{j,r}^{t+1} + \frac{1}{4}\sum_{r=1}^{k} \frac{(w_{j,r}^{t+1})^2}{a_j} + \lambda \left\|\bar{\boldsymbol{w}}_j^{t+1}\right\|_2 + \sum_{r=1}^{k}\left(\frac{(w_{j,r}^t)^2}{4a_j} - g_{r,j} w_{j,r}^t\right) - \lambda \left\|\bar{\boldsymbol{w}}_j^t\right\|_2\right] \tag{33}$$

The above bound is evidently a separable quadratic function with ℓ_2-regularization. We would like to use Eq. (33) to perform block coordinate descent on the ℓ_1/ℓ_2-regularized loss Q from Eq. (32). Thus, to minimize the upper bound with respect to $w_{j,r}^{t+1}$, we would like to minimize a function of the form

$$\frac{1}{2} a \sum_{r=1}^{k} w_r^2 + \sum_{r=1}^{k} b_r w_r + \lambda \|\boldsymbol{w}\|_2 \quad . \tag{34}$$

The following lemma gives a closed form solution for the minimizer of the above.

Lemma 10 *The minimizing $\boldsymbol{w}^\star$ of Eq. (34) is*

$$\boldsymbol{w}^\star = -\frac{1}{a}\left[1 - \frac{\lambda}{\|\boldsymbol{b}\|_2}\right]_+ \boldsymbol{b} \quad .$$

Proof We first give conditions under which the solution $\boldsymbol{w}^\star$ is $\mathbf{0}$. Characterizing the $\mathbf{0}$ solution can be done is several ways. We give here an argument based on the calculus of subgradients (Bertsekas, 1999). The subgradient set of $\lambda\|\boldsymbol{w}\|$ at $\boldsymbol{w} = \mathbf{0}$ is the set of vectors $\{\boldsymbol{z} : \|\boldsymbol{z}\| \leq \lambda\}$. Thus, the subgradient set of Eq. (34) evaluated at $\boldsymbol{w} = \mathbf{0}$ is $\boldsymbol{b} + \{\boldsymbol{z} : \|\boldsymbol{z}\| \leq \lambda\}$, which includes $\mathbf{0}$ if and only if $\|\boldsymbol{b}\|_2 \leq \lambda$. When $\|\boldsymbol{b}\|_2 \leq \lambda$, we clearly have $1 - \lambda/\|\boldsymbol{b}\|_2 \leq 0$ which immediately implies that $[1 - \lambda/\|\boldsymbol{b}\|_2]_+ = 0$ and therefore $\boldsymbol{w}^\star = \mathbf{0}$ in the statement of the lemma. Next, consider the case when $\|\boldsymbol{b}\|_2 > \lambda$, so that $\boldsymbol{w}^\star \neq \mathbf{0}$ and $\partial\|\boldsymbol{w}\|_2 = \boldsymbol{w}/\|\boldsymbol{w}\|_2$ is well defined. Computing the gradient of Eq. (34) with respect to $\boldsymbol{w}$, we obtain the optimality condition

$$a\boldsymbol{w} + \boldsymbol{b} + \frac{\lambda}{\|\boldsymbol{w}\|_2}\boldsymbol{w} = \mathbf{0} \qquad \Rightarrow \qquad \left(1 + \frac{\lambda}{a\|\boldsymbol{w}\|_2}\right)\boldsymbol{w} = -\frac{1}{a}\boldsymbol{b} \quad ,$$

which implies that $\boldsymbol{w} = s\boldsymbol{b}$ for some $s \leq 0$. We next replace the original objective of Eq. (34) with

$$\underset{s}{\text{minimize}} \ \frac{1}{2}s^2\left(a\sum_{r=1}^{k} b_r^2\right) + s\sum_{r=1}^{k} b_r^2 - \lambda s\|\boldsymbol{b}\|_2 \quad .$$

Taking the derivative of the objective with respect to s yields,

$$s\, a\,\|\boldsymbol{b}\|_2^2 + \|\boldsymbol{b}\|_2^2 - \lambda\|\boldsymbol{b}\|_2 = 0 \qquad \Rightarrow \qquad s = \frac{\lambda\|\boldsymbol{b}\|_2 - \|\boldsymbol{b}\|_2^2}{a\|\boldsymbol{b}\|_2^2} = \frac{1}{a}\left(\frac{\lambda}{\|\boldsymbol{b}\|_2} - 1\right) \quad .$$

```
INPUT: Training set $S = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^m$;
          Regularization $\lambda$; number of rounds $T$;
          Update templates $\mathcal{A}$ s.t. $\forall \boldsymbol{a} \in \mathcal{A}$, $\sum_{i,j} a_j x_{ij}^2 \le 1$
FOR $t = 1$ to $T$
    CHOOSE $\boldsymbol{a} \in \mathcal{A}$
    FOR $j$ s.t. $a_j > 0$
        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 6: GradBoost for ℓ_1/ℓ_2-regularized multiclass boosting.

Combining the above result with the case when $\|\boldsymbol{b}\|_2 \le \lambda$ (which yields that $\boldsymbol{w}^\star = \mathbf{0}$) while noticing that $\lambda / \|\boldsymbol{b}\|_2 - 1 \le 0$ when $\|\boldsymbol{b}\|_2 \ge \lambda$ gives the lemma's statement. ■

Returning to Eq. (33), we derive the update to the j^{th} row of W. Defining the gradient vector $\boldsymbol{g}_j = [g_{1,j} \cdots g_{r,j}]^\top$ and performing a few algebraic manipulations to Lemma 10, we obtain that the update that is performed in order to minimize Eq. (33) with respect to row $\overline{\boldsymbol{w}}_j$ of W is

$$\overline{\boldsymbol{w}}_j^{t+1} = \left(\overline{\boldsymbol{w}}_j - 2a_j \boldsymbol{g}_j\right) \left[1 - \frac{2a_j\lambda}{\left\|\overline{\boldsymbol{w}}_j^t - 2a_j\boldsymbol{g}_j\right\|_2}\right]_+ . \tag{35}$$

To recap, we obtain an algorithm for minimizing the ℓ_1/ℓ_2-regularized multiclass loss by iteratively choosing update templates $\boldsymbol{a}$ and then applying the update provided in Eq. (35) to each index j for which $a_j > 0$. The pseudocode of the algorithm is given in Fig. 6.

8. Learning sparse models by feature induction and pruning

Boosting naturally connotes induction of base hypotheses, or features. Our infusion of regularization into boosting and coordinate descent algorithms also facilitates the ability to prune back selected features. The end result is an algorithmic infrastructure that facilitates forward induction of new features and backward pruning of existing features in the context of an expanded model while optimizing weights. In this section we discuss the merits of our paradigm as the means for learning sparse models by introducing and scoring features that have not yet entered a model and pruning back features that are no longer predictive. To do so, we consider the progress bounds we derived for AdaBoost and GradBoost. We show how the bounds can provide the means for scoring and selecting new features. We also show that, in addition to scoring features, each of our algorithms provides a stopping criterion for inducing new features, indicating feature sets beyond which the introduction of new hypotheses *cannot* further decrease the loss. We begin by considering AdaBoost and then revisit the rest of the algorithms and regularizers, finishing the section with a discussion of the boosting termination and hypothesis pruning benefits of our algorithms.

Scoring candidate hypotheses for ℓ_1-regularized AdaBoost The analysis in Sec. 2 also facilitates assessment of the quality of a candidate weak hypothesis (newly examined feature) during the boosting process. To obtain a bound on the contribution of a new weak hypothesis we plug the form for δ_j^t from Eq. (8) into the progress bound of Eq. (5). The progress bound can be written as a sum over the progress made by each hypothesis that the chosen template $\boldsymbol{a} \in \mathcal{A}$ activates. For concreteness and overall utility, we focus on the case where we add a single weak hypothesis j. Since j is as yet un-added, we assume that $w_j = 0$, and as in standard analysis of boosting we assume that $|x_{i,j}| \leq 1$ for all i. Furthermore, for simplicity we assume that $a_j = 1$ and $a_k = 0$ for $k \neq j$. That is, we revert to the standard boosting process, also known as sequential boosting, in which we add a new hypothesis on each round. We overload our notation and denote by Δ_j the decrease in the ℓ_1-regularized log-loss due to the addition of hypothesis j, and if we define

$$\nu_j^- = \frac{-\lambda + \sqrt{\lambda^2 + 4\mu_j^+\mu_j^-}}{2\mu_j^-} \quad \text{and} \quad \nu_j^+ = \frac{\lambda + \sqrt{\lambda^2 + 4\mu_j^+\mu_j^-}}{2\mu_j^-} \quad ,$$

routine calculations allow us to score a new hypothesis as

$$\Delta_j = \begin{cases} \mu_j^+ + \mu_j^- - \frac{\mu_j^+}{\nu_j^-} - \mu_j^-\nu_j^- - \lambda\left|\log \nu_j^-\right| & \text{if } \mu_j^+ > \mu_j^- + \lambda \\ \mu_j^+ + \mu_j^- - \frac{\mu_j^+}{\nu_j^+} - \mu_j^-\nu_j^+ - \lambda\left|\log \nu_j^+\right| & \text{if } \mu_j^- > \mu_j^+ + \lambda \\ 0 & \text{if } |\mu_j^+ - \mu_j^-| \leq \lambda \ . \end{cases} \tag{36}$$

We can thus score candidate weak hypotheses and choose the one with the highest potential to decrease the loss. Indeed, if we set $\lambda = 0$ the decrease in the loss becomes $\left(\sqrt{\mu_j^+} - \sqrt{\mu_j^-}\right)^2$, which is the standard boosting progress bound (Collins et al., 2002).

Scoring candidate hypotheses for ℓ_1/ℓ_∞-regularized AdaBoost Similar arguments to those for ℓ_1-regularized AdaBoost show that we can assess the quality of a new hypothesis we consider during the boosting process for mixed-norm AdaBoost. We do not have a closed form for the update $\boldsymbol{\delta}$ that maximizes the bound in the change in the loss from Eq. (17). However, we can solve for the optimal update for the weights associated with hypothesis j as in Eq. (21) using the algorithm of Fig. 2 and the analysis in Sec. 4. We can plug the resulting updates into the progress bound of Eq. (17) to score candidate hypotheses. The process of scoring hypotheses depends only on the variables $\mu_{r,j}^{\pm}$, which are readily available during the weight update procedure. Thus, the scoring process introduces only minor computational burden over the weight update. We also would like to note that the scoring of hypotheses takes the same form for multiclass and multitask boosting provided that $\mu_{r,j}^{\pm}$ have been computed.

Scoring candidate hypotheses for ℓ_1-regularized GradBoost It is also possible to derive a scoring mechanism for the induction of new hypotheses in GradBoost by using the lower bound used to derive the GradBoost update. The process of selecting base hypotheses is analogous to the selection process to get new weak hypotheses for AdaBoost with ℓ_1-regularization, which we considered above. Recall the quadratic upper bounds on the loss $Q(\boldsymbol{w})$ from Sec. 5. Since we consider the addition of new hypotheses, we can assume that $w_j^t = 0$ for all candidate hypotheses. We can thus combine the bound of Eq. (22) and the update of Eq. (25) to obtain

$$Q(\boldsymbol{w}^t) - Q(\boldsymbol{w}^{t+1}) \geq \begin{cases} 2a_j\left(|g_j| - \lambda\right)^2 & |g_j| > \lambda \\ 0 & \text{otherwise} \ . \end{cases} \tag{37}$$

If we introduce a single new hypothesis at a time, that is, we use an update template $\boldsymbol{a}$ such that $a_j \neq 0$ for a single index j, we can score features individually. To satisfy the constraint that $\sum_i a_j x_{i,j}^2 \leq 1$, we simply let $a_j = 1/\sum_i x_{i,j}^2$. In this case, Eq. (37) becomes

$$Q(\boldsymbol{w}^t) - Q(\boldsymbol{w}^{t+1}) \geq \begin{cases} \frac{2(|g_j| - \lambda)^2}{\sum_{i=1}^m x_{i,j}^2} & |g_j| > \lambda \\ 0 & \text{otherwise} \ . \end{cases}$$

Note that the above progress bound incorporates a natural trade-off between the coverage of a feature, as expressed by the term $\sum_i x_{i,j}^2$, and its correlation with the label, expressed through the difference $|g_j| = |\mu_j^+ - \mu_j^-|$. The larger the difference between μ_j^+ and μ_j^-, the higher the potential of the feature. However, this difference is scaled back by the sum $\sum_i x_{i,j}^2$, which is proportional to the coverage of the feature. A similar though more tacit tension between coverage and correlation is also exhibited in the score for AdaBoost as defined by Eq. (36).

Scoring candidate hypotheses for mixed-norm regularized GradBoost The scoring of new hypotheses for multiclass and multitask GradBoost is similar to that for mixed-norm AdaBoost. The approach for ℓ_1/ℓ_∞ regularization is analogous to the scoring procedure for AdaBoost. When a hypothesis or feature with index j is being considered for addition on round t, we know that $\overline{\boldsymbol{w}}_j^t = \mathbf{0}$. We plug the optimal solution of Eq. (28) (or equivalently Eq. (31)), into the progress bound and obtain the potential loss decrease due to the introduction of a new feature. While we cannot provide a closed form expression for the potential progress, the complexity of the scoring procedure requires the same time as a single feature weight update.

The potential progress does take a closed-form solution when scoring features using ℓ_1/ℓ_2 mixed-norm regularized GradBoost. We simply plug the update of Eq. (35) into the bound of Eq. (33) while recalling the definition of the gradient terms $\boldsymbol{g}_j = [g_{1,j} \cdots g_{k,j}]^\top$ for multiclass or multitask boosting. Since, again, $\overline{\boldsymbol{w}}_j^t = 0$ for a new hypothesis j, a few algebraic manipulations yield that the progress bound when adding a single new hypothesis j is

$$Q(W^{t+1}) - Q(W^t) \geq \frac{\left(\left[\left\|\boldsymbol{g}_j\right\|_2 - \lambda\right]_+\right)^2}{\sum_{i=1}^m x_{i,j}^2} \quad . \tag{38}$$

Termination Conditions The progress bounds for each of the regularizers also provide us with principled conditions for terminating the induction of new hypotheses. We have three different conditions for termination, depending on whether we use ℓ_1, ℓ_1/ℓ_∞, or ℓ_1/ℓ_2-regularization. Going through each in turn, we begin with ℓ_1. For AdaBoost, Eq. (36) indicates that when $|\mu_j^+ - \mu_j^-| \leq \lambda$, our algorithm assigns the hypothesis zero weight and no progress can be made. For GradBoost, $g_j = \mu_j^- - \mu_j^+$, so the termination conditions are identical. In the case of ℓ_1/ℓ_∞-regularization, the termination conditions for AdaBoost and Gradboost are likewise identical. For AdaBoost, Corollary 6 indicates that when $\sum_{r=1}^k |\mu_{r,j}^+ - \mu_{r,j}^-| \leq \lambda$ the addition of hypothesis j cannot decrease. Analogously for GradBoost, Corollary 8 shows that when $\sum_{r=1}^k |g_{r,j}| \leq \lambda$ then $\overline{\boldsymbol{w}}_j^{t+1} = \mathbf{0}$, which is identical since $g_{r,j} = \mu_{r,j}^- - \mu_{r,j}^+$. For ℓ_1/ℓ_2-regularized GradBoost, examination of Eq. (38) indicates that if $\left\|\boldsymbol{g}_j\right\|_2 \leq \lambda$, then adding hypothesis j does not decrease the loss.

As we discuss in the next section, each of our algorithms converges to the optimum of its respective loss. Therefore, assume we have learned a model with a set of features such that the feature weights are at the optimum for the regularized loss (using only the current features in the model) we are minimizing. The convergence properties indicate that if our algorithm cannot make progress using the j^{th} hypothesis, then truly no algorithm that uses the j^{th} hypothesis in conjunction with the current model can make progress on the objective. The same property holds even in the case of an infinite hypothesis space. We thus see that each algorithm gives a condition for terminating boosting. Specifically, we know when we have exhausted the space of base hypotheses that can contribute to a reduction in the *regularized* loss.

Backpruning In addition to facilitating simple scoring mechanisms, the updates presented for AdaBoost and GradBoost in the various settings also enable resetting the weight of a hypothesis if in retrospect (after further boosting rounds) its predictive power decreases. Take for example the ℓ_1-penalized boosting steps of AdaBoost. When the weight-adjusted difference between μ_j^+ and μ_j^- in the AdaBoost algorithm in Fig. 1 falls below λ, that is, $\left|\mu_j^+ e^{w_j^t/a_j} - \mu_j^- e^{-w_j^t/a_j}\right| \leq \lambda$, we set $\delta_j^t = -w_j^t$ and we zero out the j^{th} weight. Thus, the ℓ_1-penalized boosting steps enable *both* induction of new hypotheses along with backward pruning of previously selected hypotheses. Similar statements apply to all of our algorithms, namely, when weight-adjusted correlations or gradients fall below the regularization, the algorithms can zero weights or rows of weights.

As mentioned in the introduction, we can also alternate between pure weight updates (restricting ourselves to the current set of hypotheses) and pure induction of new hypotheses (keeping the weight of existing hypotheses intact). As demonstrated empirically in our experiments, the end result of boosting with the sparsity promoting ℓ_1, ℓ_1/ℓ_2, or ℓ_1/ℓ_∞-regularizers is a compact and accurate model. This approach of alternating weight optimization and hypothesis induction is also reminiscent of the recently suggested forward-backward greedy process for learning sparse representations in least-squares models (Zhang, 2008). However, in our setting the backward pruning is not performed in a greedy manner but is rather driven by the non-differentiable convex regularization penalties.

Template Selection Finally, the templating of our algorithms allows us to make the templates $\boldsymbol{a}$ application and data dependent. If the computing environment consists of a few uncoupled processors and the features are implicit (e.g. boosted decision trees), then the most appropriate set of templates is the set of singleton vectors (we get the best progress guarantee from the highest scoring singletons). When the features are explicit and the data is rather sparse, i.e. $\sum_j |x_{i,j}| \ll n$ where $x_{i,j}$ is the prediction of the j^{th} base hypothesis on the i^{th} example, we can do better by letting the templates be dense. For example, for ℓ_1-regularized AdaBoost, we can use a single template vector $\boldsymbol{a}$ with $a_j = 1/\max_i \sum_j |x_{i,j}|$ for all j. For GradBoost, we can use the single template $\boldsymbol{a}$ with that $a_j = 1/\left(n \sum_{i=1}^m x_{i,j}^2\right)$. Dense templates are particularly effective in parallel computing environments where we can efficiently compute importance-weighted correlations μ_j^+ and μ_j^- or gradients $g_{r,j}$ for all possible features. We can then make progress simultaneously for multiple features and use the full parallelism available.

9. Convergence properties

We presented several variants for boosting-based stepping with different regularized logistic losses throughout this paper. To conclude the formal part of the paper, we discuss the convergence properties of these algorithms. Each of our algorithms is guaranteed to converge to an optimum of its respective loss, provided that the weight of each feature is examined and potentially updated sufficiently often. We thus discuss jointly the convergence of all the algorithms. Concretely, when the number of hypotheses is finite, the boosting updates for both the AdaBoost and GradBoost algorithms are guaranteed to converge under realistic conditions. The conditions require the templates $\boldsymbol{a} \in \mathcal{A}$ to span the entire space, assume that there are a finite number of hypotheses, and each hypothesis is updated infinitely often. The optimal weights of the (finitely many) induced features can then be found using a set of templates that touch each of the features. In particular, a single template that updates the weight of all of the features simultaneously can be used, as could a set of templates that iteratively selects one feature at a time. The following theorem summarizes the convergence properties. Due to its technical nature we provide its proof in Appendix C. The proof relies on the fact that the regularization term forces the set of possible solutions of any of the regularized losses we discuss to be compact. In addition, each of the updates guarantees some decrease in its respective regularized loss. Roughly, each sequence of weights obtained by any of the algorithms therefore converges to a stationary point guaranteed to be the unique optimum by convexity. We are as yet unable to derive rates of convergence for our algorithms, but we hope that recent views of boosting processes as primal-dual games (Schapire, 1999; Shalev-Shwartz and Singer, 2008) or analysis of randomized coordinate descent algorithms (Shalev-Shwartz and Tewari, 2009) can help in deriving more refined results.

Theorem 11 *Assume that the number of hypotheses is finite and each hypothesis participates in an update based on either AdaBoost or GradBoost infinitely often. Then all the variants of regularized boosting algorithms converge to the optimum of their respective objectives. Specifically,*

i. AdaBoost with ℓ_1-regularization converges to the optimum of Eq. (2).

ii. Multitask and Multiclass AdaBoost with ℓ_1/ℓ_∞-regularization converge to the optimum of Eq. (3) and Eq. (4), respectively.

iii. GradBoost with ℓ_1-regularization convergences to the optimum of Eq. (2).

iv. Multitask and Multiclass GradBoost with ℓ_1/ℓ_∞-regularization converge to the optimum of Eq. (3) and Eq. (4), respectively.

v. GradBoost with ℓ_1/ℓ_2-regularization converges to the optimum of Eq. (32).

10. Experiments

In this section we focus on empirical evaluations of our algorithms. We compare the algorithms' performance to a few other state-of-the-art learning algorithms for the problems we investigate and discuss their relative performance. We emphasize in particular the ability to achieve structural sparsity in multiclass problems.

Boosting Experiments For our first series of experiments, we focus on boosting and feature induction, investigating the effect of ℓ_1-regularization and its early stopping. We perform both classification and regression. For the classification task, we used the Reuters RCV1 Corpus (Lewis et al., 2004), which consists of 804,414 news articles; after stemming and stopwording, there are around 100,000 unigram features in the corpus with a high degree of sparsity. Each article is labeled as one or more of MCAT, CCAT, ECAT, or GCAT (medical, corporate/industrial, economics, or government), and we trained boosted classifiers for each class separately (one vs. all classification). We present average classification error rates and logistic loss rates over a series of tests using 30,000 (randomly chosen) articles with a 70/30 training/test split in Fig. 7. The top row of the figure shows misclassification rates and the bottom log-loss rates for the four different classes.

As a baseline comparison, we used AdaBoost regularized with a smooth-ℓ_1 penalty (Dekel et al., 2005), which is an approximation to the ℓ_1-norm that behaves similarly to ℓ_2 when w is close to 0. Specifically, in the smooth-ℓ_1 version the weights are penalized as follows,

$$\lambda\left[\log(1+e^{w})+\log(1+e^{-w})\right] \quad .$$

We also compared our base ℓ_1-regularized versions to an ℓ_2-regularized logistic regression with intelligently chosen features. The description of the feature selection process for ℓ_2 logistic regression is provided in Appendix D. For the ℓ_1-regularized boosting, we chose a penalty of $\lambda = 4$ using cross-validation. For the smooth-ℓ_1 boosting we chose $\lambda = 2$, which gave the best performance of the penalties we considered on test set error rates, and for the ℓ_2-regularized a small penalty of $\lambda = 10^{-2}$ gave the best performance on validation and test data. For both boosting algorithms, we ran "totally corrective" boosting (Warmuth et al., 2006) in which the weights of the selected features are optimized after each induction step. We added the 30 top-scoring features at every iteration for the ℓ_1 booster and the single top-scoring feature for the smooth-ℓ_1 regularized booster.

The graphs in Fig. 7 underscore an interesting phenomenon. In all of the graphs, the ℓ_1-regularized boosting algorithm ceases adding features at around iteration 30 (about 700 features after the backward pruning steps). Hence, the error and loss lines for ℓ_1-boosting terminate early, while the smooth-ℓ_1 variant starts over-fitting the training on set as early as iteration 200. In general, the ℓ_2-regularized logistic regression had test loss and error rates slightly better than smooth-ℓ_1 boosted logistic regression, but worse than the ℓ_1-regularized booster.

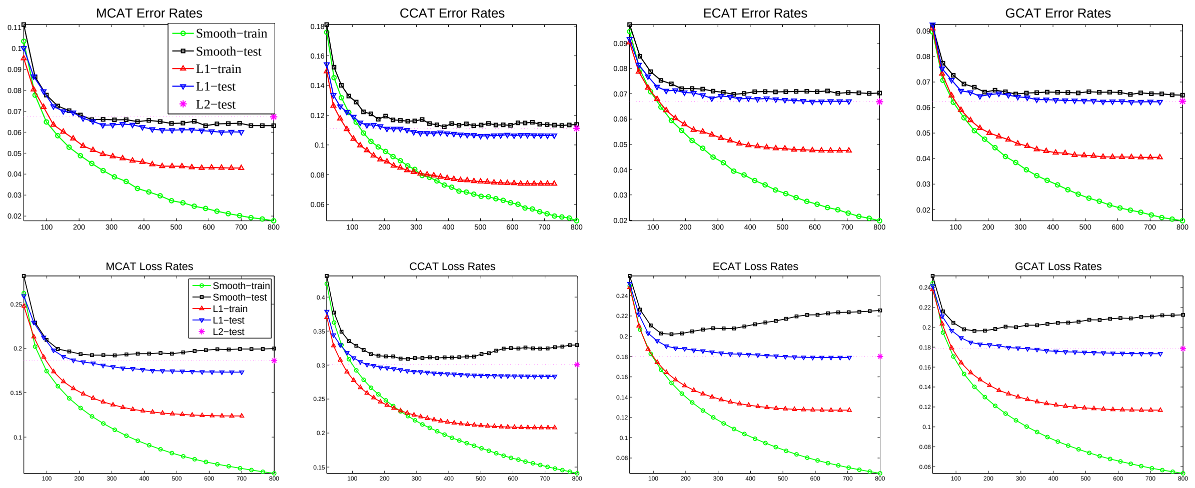


Figure 7: Error rates and losses on the Reutes corpus for various boosting-based algorithms. The legend is the same for all graphs.

We also conducted various regression experiments. We describe here the results obtained for the Boston housing data set from the UCI repository (Asuncion and Newman, 2007) and a dataset on controlling an F16 aircraft, where the goal is to predict a control action on the ailerons of the aircraft given its state (Camacho, 1997). We standardized both datasets so that their variables are all in the range $[0, 1]$ (for housing) or $[-1, 1]$ (for ailerons). We then used boosted ε-insensitive regression (Dekel et al., 2005) to learn a predictor $\boldsymbol{w}$. In this case, our objective is

$$\sum_{i=1}^{m} \log\left(1 + e^{\boldsymbol{w}\cdot\boldsymbol{x}_i - y_i - \varepsilon}\right) + \log\left(1 + e^{y_i - \boldsymbol{w}\cdot\boldsymbol{x}_i - \varepsilon}\right) \quad , \tag{39}$$

which approximates ε-insensitive hinge regression $[\boldsymbol{w}\cdot\boldsymbol{x}_i - y_i - \varepsilon]_+ + [y_i - \boldsymbol{w}\cdot\boldsymbol{x}_i - \varepsilon]_+$ where $[x]_+ = \max\{x, 0\}$. For ε-insensitive regression, an analysis similar to that for standard boosting can be performed to compute μ^+ and μ^- for every feature (see appendix section A), which allows us to perform scoring during feature induction and to take update steps identical to those already described.

For these tests, we compared the unregularized "classical" sequential AdaBoost, ℓ_1-regularized totally corrective boosting with induction of eight top-scoring features at the end of each optimization step, 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 products of features. All algorithms were started with a single bias feature. Thus, the algorithms could construct arbitrarily many products of raw features as base (weak) hypotheses and explore complex correlations between the features. For ℓ_1-regularized least squares, we simply trained on the base regressors, and for ℓ_2-regularized hinge loss we trained on the base regressors using projected subgradient methods described in Shalev-Shwartz et al. (2007).

Fig. 8 illustrates the results of these experiments. The housing results are the left pair of graphs while aileron results are the right. The left plot of each pair provides the root-mean-square error on the test set, and the right of the pair the average absolute error on the test sets. In all experiments, we set $\lambda = .02$ for the ℓ_1-penalty, and the ℓ_1-regularized booster stopped after inducing an average of under 35 features. The last iteration of the ℓ_1-regularized version is marked with a star in the graphs, after which a dotted line indicates the resulting performance. We see that even when the sequential (denoted Classical) boosting is allowed to run for 1000 iterations, its performance on test does not meet the performance obtained by the 35 feature ℓ_1-regularized model. As a further experiment, we allowed the sequential boosting process to run for 3000 iterations, yet still its performance did

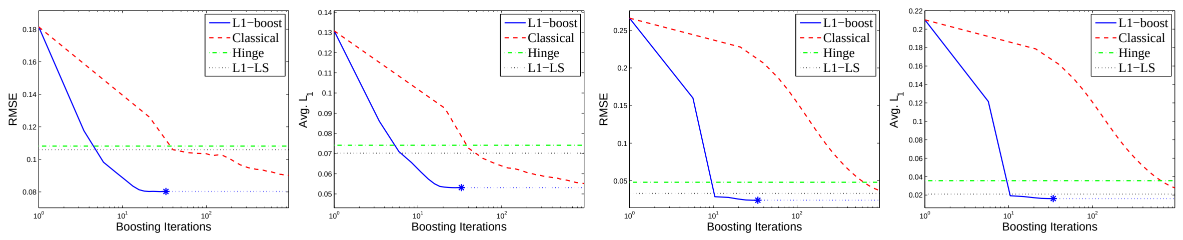


Figure 8: Regression losses (mean squared error and absolute error) for Boston housing (left two) and Aileron.

not match the 35-feature model built by the ℓ_1-penalized version. Furthermore, the latter trains at least order of magnitude faster than the sequentially learned regressor and results in a significantly simpler model. The 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.

Multiclass Experiments In this set of experiments, we compare the different structured regularizers with multiclass logistic losses to one another and to unstructured ℓ_1 and ℓ_2^2 regularization, providing examples in which the structured regularization can help improve performance. For our multiclass experiments, we focus on two error metrics. The first is a simple misclassification error, the proportion of examples classified incorrectly. The second is *coverage*. The idea behind coverage is to measure how wrong our classifier is. Given k weight vectors $\boldsymbol{w}_r$ and an example $\boldsymbol{x}_i$, the coverage is the position in the sorted list of inner products $\boldsymbol{w}_r \cdot \boldsymbol{x}_i$ the correct weight vector is. 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. So coverage is an upper bound on the misclassification rate that also gives an indication of how wrong the classifier is in predicting the rank of the correct class.

We used five datasets for our multiclass experiments. The first two were the StatLog Landsat Satellite dataset (Spiegelhalter and Taylor, 1994) and the MNIST handwritten digits database. We also experimented with three datasets from the UCI machine learning repository (Asuncion and Newman, 2007): the Pendigit dataset, a vowel recognition dataset, and the noisy waveform database. The purpose of the experiments is not to claim that we get better classification results across many datasets than previous approaches, but rather to demonstrate the performance of our algorithms, showing some of the trade-offs of different regularization and boosting strategies.

We begin with the StatLog Landsat Satellite dataset, which consists of spectral values of pixels in 3×3 neighborhoods in a satellite image. We expanded the features by taking products of all possible features, giving 1296 features for each example. The goal is to classify a pixel (a piece of ground) as one of six ground types (e.g. red soil, crops, damp soil). We separated the data into a training set of 3104 examples, 1331 validation examples, and 2000 test samples. In Fig. 9, we plot coverage and log-loss on the test set as a function of sparsity and as a function of the number of features actually used. The classifiers were all trained with a random training set using 240 examples per class (results when training with fewer or more examples were similar). The plots on the left show the test set coverage as a function of the proportion of zeros in the learned weight matrix $W^\star$. The far left plot shows 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 *zero* rows in W. The middle left plot shows test set coverage simply as a function of overall sparsity in W, and thus does not reflect the number of features that we must compute. The plots on the right similarly show test set loss as a function either of row sparsity in W or overall sparsity. We see from the plots that for a given performance level, the ℓ_1-regularized solution is sparser in terms of the absolute number of zeros in $W^\star$. However, the ℓ_1-regularized classifier requires at least 50% more features to be computed than does the ℓ_1/ℓ_2-regularized classifier for the same test accuracy. The results for misclassification rates

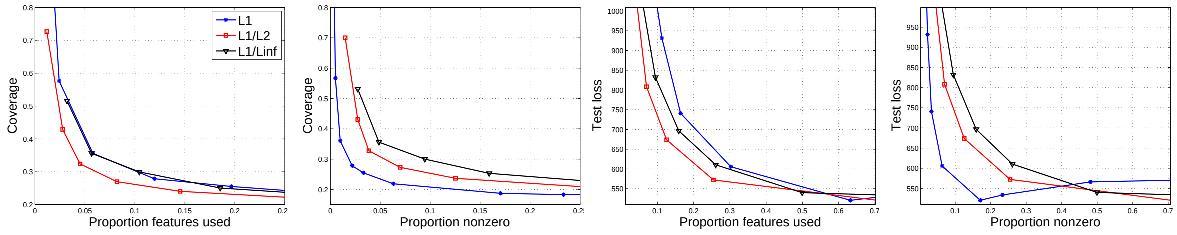


Figure 9: LandSat test coverage and losses. Far left: coverage versus row sparsity. Middle left: coverage versus overall sparsity. Middle right: test loss versus row sparsity. Far right: test loss versus overall sparsity.

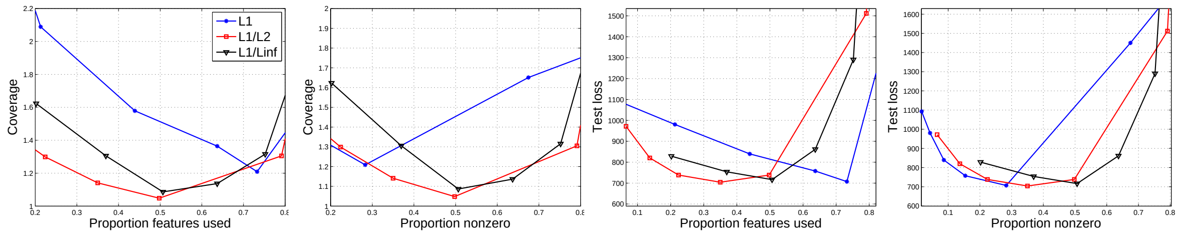


Figure 10: Vowels test coverage and losses. Far left: coverage versus row sparsity. Middle left: coverage versus overall sparsity. Middle right: test loss versus row sparsity. Far right: test loss versus overall sparsity.

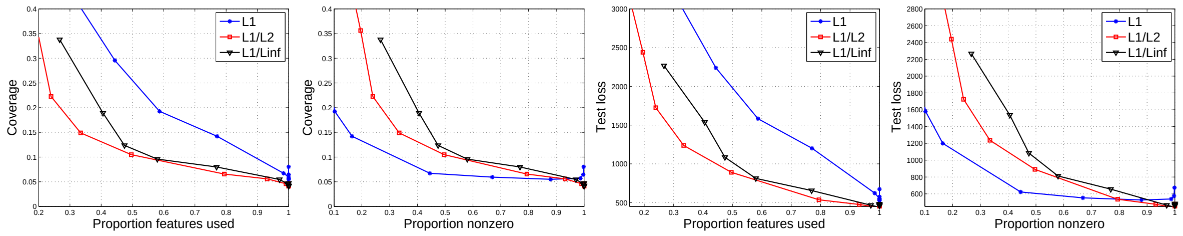


Figure 11: Pendigit handwritten digits test coverage and losses. Far left: coverage versus row sparsity. Middle left: coverage versus overall sparsity. Middle right: test loss versus row sparsity. Far right: test loss versus overall sparsity.

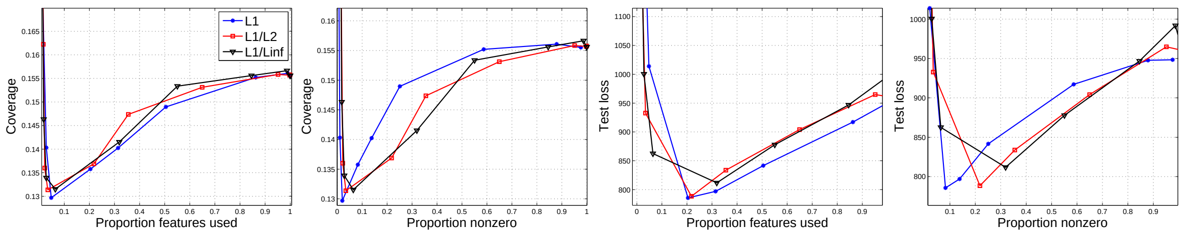


Figure 12: Waveform test coverage and losses. Far left: coverage versus row sparsity. Middle left: coverage versus overall sparsity. Middle right: test loss versus row sparsity. Far right: test loss versus overall sparsity.

are similar. We computed these results over ten randomly chosen subsets of the training data and the variance of each point in the plot is smaller than 10^{-3}.

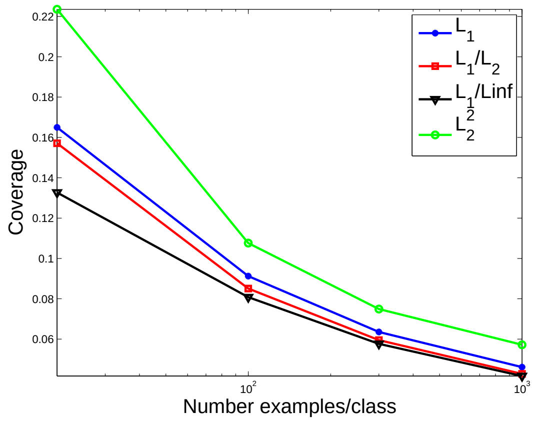
In figures 10, 11, and 12, we plot the results obtained on the remaining UCI datasets. The vowel experiments require classifying a vowel spoken in the middle of a word as one of 11 different vowels. We expanded the original 10 features by taking products of all features. We trained with 20 training examples per class (results are similar using 40 examples per class) over 10 different random training runs. We plot test set coverage and loss as a function of sparsity in Fig. 10. In the Pendigit experiments we classify a handwritten digit, with 16 features recorded from a pressure-sensitive stylus, as one of 10 different digits. Again we expanded into products of features, and in Fig. 11 we plot loss and coverage for this task versus sparsity in the solution matrix W. We trained with 300 examples per class (as with our other experiments, results were similar across training set sizes) and used a test set of 3498 examples. The last UCI dataset we consider was the Waveform dataset, a synthetic 3 class problem where the goal is to classify different randomly generated waves. Originally consisting of 40 attributes, we expanded the feature space using products of feature and created 820 features. We then applied our multiclass boosting algorithms. We used a test set of 2500 examples and training set of 600 examples per class. We plot our test loss and coverage results versus sparsity in W in Fig. 12. We repeated each of our UCI experiments over 10 randomly sub-selected training sets. Each of the plots in figures 10 and 11 exhibits results similar to the results obtained for the Landsat dataset. Examining the performance as a function of the actual number of features that must be computed, ℓ_1/ℓ_2 and ℓ_1/ℓ_∞ regularizations seem to yield better performance, while if we are mainly concerned with overall sparsity of the learned matrix W, ℓ_1-regularization gives better performance. For the Waveform dataset (Fig. 12), it seems that performance was similar across all the regularizers, which may be an artifact of the waveforms being synthetic, with performance improving as the number of features used got very small.

We also conducted experiments using the MNIST handwritten digits database. The MNIST dataset consists of 60,000 training examples and a 10,000 example test set and has 10-classes. Each image is a gray-scale 28×28 matrix, which we represent as vector $\boldsymbol{x}_i \in \mathbb{R}^{784}$. Rather than directly using the input $\boldsymbol{x}_i$, we learned weights $\boldsymbol{w}_j$ for Kernel-based weak-hypotheses

$$h_j(\boldsymbol{x}) = K(\boldsymbol{x}_j, \boldsymbol{x}), \qquad K(\boldsymbol{x}, \boldsymbol{z}) = e^{-\frac{1}{2}\|\boldsymbol{x}-\boldsymbol{z}\|^2} \quad ,$$

for $j \in S$, where S is a 2766 element support set. We generated the support set by running the Perceptron algorithm once through the dataset while keeping examples on which it made classification mistakes. We obtained this way a 27,660 dimensional multiclass problem to which we apply our algorithms. On the left side of Fig. 13, we plot the coverage on the 10,000 example test set of each algorithm versus the number of training examples used per class. We chose regularization values using cross-validation. The performance improves as the number of training examples increases. However, it is clear that the sparsity promoting regularizers, specifically the structural ℓ_1/ℓ_∞ and ℓ_1/ℓ_2 regularizers, yield better performance than the others. The error rate on the test set is roughly half the coverage value and behaves qualitatively similar.

To conclude this section we would like to attempt to provide a few insights into the relative merits of AdaBoost versus GradBoost when both can be applied, namely, for ℓ_1 and ℓ_1/ℓ_∞-regularized problems. On the right side of Fig. 13, we plot the training objective as a function of training time for both AdaBoost and GradBoost on the Landsat and MNIST datasets. On the same time scale, we plot the test error rate and sparsity of the classifiers as a function of training time in Fig. 14 (in the left and right plots, respectively). From Fig. 14, 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 backward-pruning phase in which they remove features that are not predictive enough without increasing the loss 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.



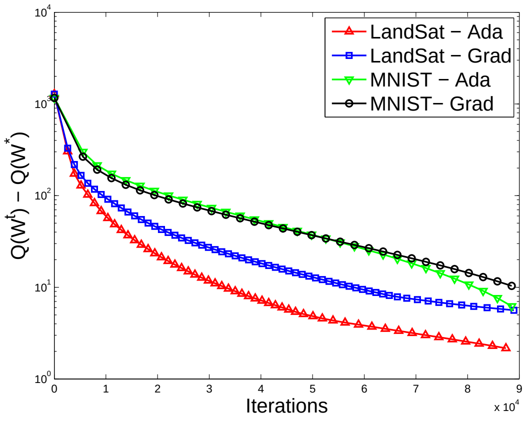
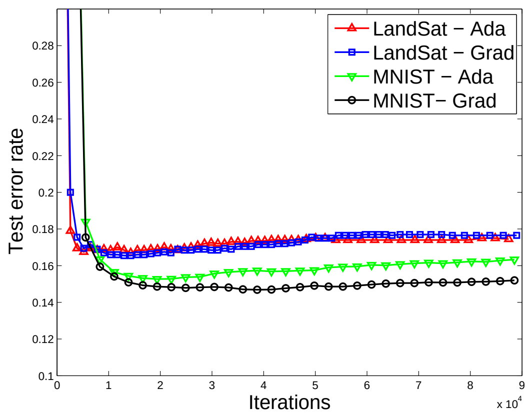


Figure 13: Left: Coverage in MNIST for different regularizers versus number of training examples per class. Right: objective value versus number of iterations for AdaBoost and GradBoost training on MNIST and LandSat.



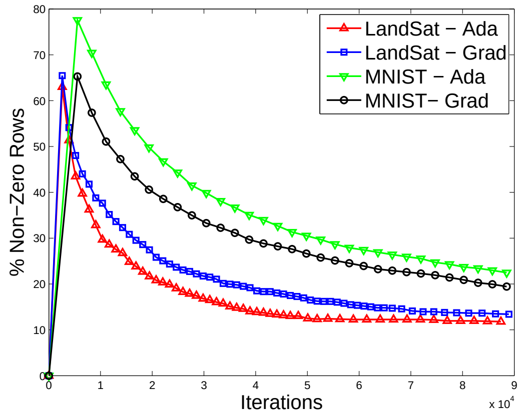


Figure 14: Left: Error rate for MNIST and LandSat versus training time. Right: Percent of non-zero features for MNIST and LandSat versus training time.

11. Conclusion

We proposed and analyzed in this paper several new variants on boosting that allow both induction and scoring of weak hypotheses as well as a new phase of backward-pruning to remove retrospectively uninformative features. Our new boosting algorithms all enjoy the same convergence guarantees, and they provide a simple termination mechanism for boosting. We described experimental results across a range of benchmark datasets, showing that our algorithms improve over previous approaches to boosting with early stopping and smooth regularization. In the experiments, the regularized versions indeed automatically terminate, typically avoid overfitting, and give good performance as a function of the number of features that were selected by the classifiers. We plan to further investigate the convergence rate of our algorithms. It may also be interesting to examine the generalization and consistency properties of the boosting process for structurally-regularized boosters using techniques from Schapire et al. (1998) or Zhang and Yu (2005) and other techniques for analyzing the generalization properties of boosting.

Acknowledgements

A substantial part of the work of J. Duchi was performed at Google. We would like to thank Shai Shalev-Shwartz for useful discussions.

References

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

D.P. Bertsekas. *Nonlinear Programming.* Athena Scientific, 1999.

S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

Rui Camacho. Ailerons dataset. URL `http://www.liaad.up.pt/ ltorgo/Regression/DataSets.html`. Available at http://www.liaad.up.pt/˜ltorgo/Regression/DataSets.html, 1997.

M. Collins, R.E. Schapire, and Y. Singer. Logistic regression, AdaBoost and Bregman distances. *Machine Learning*, 47(2/3):253–285, 2002.

O. Dekel, S. Shalev-Shwartz, and Y. Singer. Smooth epsilon-insensitive regression by loss symmetrization. *Journal of Machine Learning Research*, 6:711–741, May 2005.

J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the ℓ_1-ball for learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

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

T. Evgeniou, C.Micchelli, and M. Pontil. Learning multiple tasks with kernel methods. *Journal of Machine Learning Research*, 6:615–637, 2005.

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

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28(2):337–374, April 2000.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Pathwise coordinate optimization. *Annals of Applied Statistics*, 1(2):302–332, 2007.

Roger A. Horn and Charles R. Johnson. *Matrix Analysis.* Cambridge University Press, 1985.

L. Jacob, F. Bach, and J.-P. Vert. Clustered multi-task learning: A convex formulation. In *Advances in Neural Information Processing Systems 22*, 2008.

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

S. I. Lee, H. Lee, P. Abbeel, and A. Y. Ng. Efficient ℓ_1-regularized logistic regression. In *Proceedings AAAI-06.* American Association for Artificial Intelligence, 2006.

David Lewis, Yiming Yang, Tony Rose, and Fan Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

D. Madigan, A. Genkin, D. D. Lewis, and D. Fradkin. Bayesian multinomial logistic regression for author identification. In *25th International Workshop on Bayesian Inference and Maximum Entropy Methods in Science and Engineering*, 2005.

N. Meinshausen and P. Bühlmann. High dimensional graphs and variable selection with the Lasso. *Annals of Statistics*, 34:1436–1462, 2006.

R. Meir and G. Rätsch. An introduction to boosting and leveraging. In S. Mendelson and A. Smola, editors, *Advanced Lectures on Machine Learning*, pages 119–184. Springer, 2003.

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

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

G. Obozinski, M. Wainwright, and M. Jordan. High-dimensional union support recovery in multivariate regression. In *Advances in Neural Information Processing Systems 22*, 2008.

A. Rakotomamonjy, F. Bach, S. Canu, and Y. Grandvalet. SimpleMKL. *Journal of Machine Learning Research*, 9:2491–2521, 2008.

R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):1–40, 1999.

R.E. Schapire. The boosting approach to machine learning: An overview. In D.D. Denison, M.H. Hansen, C. Holmes, B. Mallick, and B. Yu, editors, *Nonlinear Estimation and Classification*. Springer, 2003.

R.E. Schapire. Drifting games. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, 1999.

R.E. Schapire, Y. Freund, P. Bartlett, and W.S. Lee. Boosting the margin: A new explanation for the effectiveness of voting methods. *The Annals of Statistics*, 26(5):1651–1686, October 1998.

S. Shalev-Shwartz and Y. Singer. On the equivalence of weak learnability and linear separability: new relaxations and efficient algorithms. In *Proceedings of the Twenty First Annual Conference on Computational Learning Theory*, 2008.

S. Shalev-Shwartz and A. Tewari. Stochastic methods for ℓ_1-regularized loss minimization. In *Proceedings of the 26th International Conference on Machine Learning*, 2009.

S. Shalev-Shwartz, Y. Singer, and N. Srebro. Pegasos: Primal estimated sub-gradient solver for SVM. In *Proceedings of the 24th International Conference on Machine Learning*, 2007.

D. Spiegelhalter and C. Taylor. *Machine Learning, Neural and Statistical Classification*. Ellis Horwood, 1994.

M. Warmuth, J. Liao, and G. Ratsch. Totally corrective boosting algorithms that maximize the margin. In *Proceedings of the 23rd international conference on Machine learning*, 2006.

H. Zhang, H. Liu, Y. Wu, and J. Zhu. Variable selection for the multi-category SVM via adaptive sup-norm regularization. *Electronic Journal of Statistics*, 2:1149–1167, 2008.

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

T. Zhang and F. J. Oles. Text categorization based on regularized linear classification methods. *Information Retrieval*, 4:5–31, 2001.

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

P. Zhao and B. Yu. On model selection consistency of Lasso. *Journal of Machine Learning Research*, 7:2541–2567, 2006.

P. Zhao, G. Rocha, and B. Yu. Grouped and hierarchical model selection through composite absolute penalties. Technical Report 703, Statistics Department, University of California Berkeley, 2006.

Appendix A. Progress bounds for AdaBoost

AdaBoost is guaranteed to decrease the log-loss and the exp-loss on each boosting iteration. First, we describe an alternate derivation for the progress of AdaBoost in Lemma 1. Next, we give a new progress bound in Lemma 5 for the multiclass version of AdaBoost that we study in the paper.

Proof of Lemma 1 We begin by lower bounding the change in loss for a single example on iteration t of the algorithm, which we denote by $\Delta_t(i)$. As in the pseudocode given in Fig. 1, we denote by $\boldsymbol{\delta}^t$ the difference between $\boldsymbol{w}^{t+1}$ and $\boldsymbol{w}^t$. Simple algebraic manipulations yield that

$$\begin{aligned}
&\Delta_t(i)\\
&= \log\left(1+e^{-y_i(\boldsymbol{w}^t\cdot\boldsymbol{x}_i)}\right) - \log\left(1+e^{-y_i(\boldsymbol{w}^{t+1}\cdot\boldsymbol{x}_i)}\right) = -\log\left(\frac{1+e^{-y_i(\boldsymbol{w}^{t+1}\cdot\boldsymbol{x}_i)}}{1+e^{-y_i(\boldsymbol{w}^t\cdot\boldsymbol{x}_i)}}\right)\\
&= -\log\left(\frac{1}{1+e^{-y_i(\boldsymbol{w}^t\cdot\boldsymbol{x}_i)}} + \frac{e^{-y_i(\boldsymbol{w}^{t+1}\cdot\boldsymbol{x}_i)}}{1+e^{-y_i(\boldsymbol{w}^t\cdot\boldsymbol{x}_i)}}\right) = -\log\left(1-\frac{1}{1+e^{y_i(\boldsymbol{w}^t\cdot\boldsymbol{x}_i)}} + \frac{e^{-y_i((\boldsymbol{w}^{t+1}-\boldsymbol{w}^t)\cdot\boldsymbol{x}_i)}}{1+e^{y_i(\boldsymbol{w}^t\cdot\boldsymbol{x}_i)}}\right)\\
&= -\log\left(1-q^t(i)+q^t(i)e^{-y_i(\boldsymbol{\delta}^t\cdot\boldsymbol{x}_i)}\right) \qquad (40)
\end{aligned}$$

where in Eq. (40)) we used the fact that $\boldsymbol{\delta}^t = \boldsymbol{w}^{t+1}-\boldsymbol{w}^t$ and we have defined $q^t(i) = \frac{1}{1+e^{y_i(\boldsymbol{w}^t\cdot\boldsymbol{x}_i)}}$. Recalling that $-\log(1-z) \geq z$ for $z<1$ we can bound $\Delta_t(i)$:

$$\Delta_t(i) \geq q^t(i)\left(1-e^{-y_i(\boldsymbol{\delta}^t\cdot\boldsymbol{x}_i)}\right) = q^t(i)\left[1-\exp\left(\sum_j -s_{i,j}\delta_j^t|x_{i,j}|\right)\right]$$

where $s_{i,j} = \text{sign}(y_i x_{i,j})$. Using the assumption that $\sum_j a_j|x_{i,j}| \leq 1$ along with the convexity of the exponential function we upper bound the exponential term via

$$\begin{aligned}
\exp\left(\sum_j -s_{ij}\delta_j^t|x_{i,j}|\right) &= \exp\left(\sum_j -s_{ij}a_j d_j^t|x_{i,j}|\right)\\
&\leq \sum_j a_j|x_{i,j}|\exp(-s_{ij}d_j^t) + \left(1-\sum_j a_j|x_{i,j}|\right) = \sum_j a_j|x_{i,j}|\left(\exp(-s_{ij}d_j^t)-1\right)+1 \ ,
\end{aligned}$$

where $\delta_j^t = a_j d_j^t$. We thus obtain that

$$\Delta_t(i) \geq q^t(i) \left[\sum_j a_j |x_{i,j}| \left(1 - e^{-s_{ij} d_j^t}\right) \right] .$$

Summing over all our training examples, we get

$$\begin{aligned} \Delta_t = \sum_{i=1}^m \Delta_t(i) \quad &\geq \quad \sum_{i=1}^m q^t(i) \left[\sum_j a_j |x_{i,j}| \left(1 - e^{-s_{ij} d_j^t}\right) \right] = \sum_{j=1}^n a_j \sum_{i=1}^m q^t(i) |x_{i,j}| \left(1 - e^{-s_{ij} d_j^t}\right) \\ &= \quad \sum_{j=1}^n a_j \left[\sum_{i: y_i x_{i,j} > 0} q^t(i) |x_{i,j}| \left(1 - e^{-d_j^t}\right) + \sum_{i: y_i x_{i,j} < 0} q^t(i) |x_{i,j}| \left(1 - e^{d_j^t}\right) \right] \\ &= \quad \sum_{j=1}^n a_j \left(\mu_j^+ \left(1 - e^{-d_j^t}\right) + \mu_j^- \left(1 - e^{d_j^t}\right) \right) \quad , \end{aligned} \tag{41}$$

where in Eq. (41) we use the definitions of μ_j^+ and μ_j^-,

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

Finally, replacing d_j^t with δ_j^t / a_j, we get

$$\Delta_t \geq \sum_{j=1}^n a_j \left(\mu_j^+ \left(1 - e^{-\delta_j^t / a_j}\right) + \mu_j^- \left(1 - e^{\delta_j^t / a_j}\right) \right) \quad , \tag{42}$$

which is our desired progress bound. ■

Proof of Lemma 5 For clarity, we restate the multiclass loss function for which we derive a progress bound,

$$L(W) = \sum_{i=1}^m \log \left(1 + \sum_{r \neq y_i} \exp(\boldsymbol{w}_r \cdot \boldsymbol{x}_i - \boldsymbol{w}_{y_i} \cdot \boldsymbol{x}_i) \right) \quad .$$

Our proof builds on the derivation provided by Collins et al. (2002). The setting studied by Collins et. al assumes that have one weight vector $\boldsymbol{w}$ and the hypotheses are different for each class. That is, we have $\boldsymbol{h}(\boldsymbol{x}_i, r)$ for every class r. Thus, instead of the inner-product $\boldsymbol{w}_r \cdot \boldsymbol{x}_i$ we use $\boldsymbol{w} \cdot \boldsymbol{h}(\boldsymbol{x}_i, r)$,

$$L(\boldsymbol{w}) = \sum_{i=1}^m \log \left(1 + \sum_{r \neq y_i} \exp(\boldsymbol{w} \cdot \boldsymbol{h}(\boldsymbol{x}_i, r) - \boldsymbol{w} \cdot \boldsymbol{h}(\boldsymbol{x}_i, y_i)) \right) . \tag{43}$$

We now define a *matrix* of importance weights with an entry for each example-label pair,

$$q^t(i, r) = \frac{\exp(\boldsymbol{w}^t \cdot \boldsymbol{h}(\boldsymbol{x}_i, r))}{\sum_l \exp(\boldsymbol{w}^t \cdot \boldsymbol{h}(\boldsymbol{x}_i, l))} \quad . \tag{44}$$

Under the logistic mode $q^t(i, r)$ can be simply viewed as the probability that the weight vector $\boldsymbol{w}^t$ at iteration t assigns to label r for the i^{th} example. Following analogous steps to those in Eq. (40), we obtain that the change in the loss for a single example is equal to

$$\begin{aligned}
&\Delta_t(i)\\
&= -\log\left(\frac{1+\sum_{r\neq y_i}\exp(\boldsymbol{w}^{t+1}\cdot\boldsymbol{h}(\boldsymbol{x}_i,r)-\boldsymbol{w}^{t+1}\cdot\boldsymbol{h}(\boldsymbol{x}_i,y_i))}{1+\sum_{r\neq y_i}\exp(\boldsymbol{w}^{t}\cdot\boldsymbol{h}(\boldsymbol{x}_i,r)-\boldsymbol{w}^{t}\cdot\boldsymbol{h}(\boldsymbol{x}_i,y_i))}\right)\\
&= -\log\left(\frac{1}{1+\sum_{r\neq y_i}\exp(\boldsymbol{w}^{t}\cdot\boldsymbol{h}(\boldsymbol{x}_i,r)-\boldsymbol{w}^{t}\cdot\boldsymbol{h}(\boldsymbol{x}_i,y_i))}+\right.\\
&\qquad\left.\sum_{r\neq y_i}\frac{\exp(\boldsymbol{w}^{t+1}\cdot\boldsymbol{h}(\boldsymbol{x}_i,r)-\boldsymbol{w}^{t+1}\cdot\boldsymbol{h}(\boldsymbol{x}_i,y_i))}{1+\sum_{l\neq y_i}\exp(\boldsymbol{w}^{t}\cdot\boldsymbol{h}(\boldsymbol{x}_i,l)-\boldsymbol{w}^{t}\cdot\boldsymbol{h}(\boldsymbol{x}_i,y_i))}\right)\\
&= -\log\left(1-\sum_{r\neq y_i}q^t(i,r)+\sum_{r\neq y_i}\frac{\exp(\boldsymbol{w}^{t}\cdot\boldsymbol{h}(\boldsymbol{x}_i,r)-\boldsymbol{w}^{t}\cdot\boldsymbol{h}(\boldsymbol{x}_i,y_i)+\boldsymbol{\delta}^t\cdot\boldsymbol{h}(\boldsymbol{x}_i,r)-\boldsymbol{\delta}^t\cdot\boldsymbol{h}(\boldsymbol{x}_i,y_i))}{1+\sum_{l\neq y_i}\exp(\boldsymbol{w}^{t}\cdot\boldsymbol{h}(\boldsymbol{x}_i,l)-\boldsymbol{w}^{t}\cdot\boldsymbol{h}(\boldsymbol{x}_i,y_i))}\right)\\
&= -\log\left(1-\sum_{r\neq y_i}q^t(i,r)+\sum_{r\neq y_i}q^t(i,r)\exp(\boldsymbol{\delta}^t\cdot(\boldsymbol{h}(\boldsymbol{x}_i,r)-\boldsymbol{h}(\boldsymbol{x}_i,y_i)))\right) \quad ,
\end{aligned}$$

where as before, $\boldsymbol{\delta}^t=\boldsymbol{w}^{t+1}-\boldsymbol{w}^t$. Now define the vector $\boldsymbol{\pi}_{i,r}=\boldsymbol{h}(\boldsymbol{x}_i,y_i)-\boldsymbol{h}(\boldsymbol{x}_i,r)$ and $s_{i,r,j}=\operatorname{sign}(\pi_{i,r,j})$. Using again the inequality $-\log(1-z)\geq z$, along with the newly introduced notation, we have

$$\begin{aligned}
\Delta_t(i) &= -\log\left(1-\sum_{r\neq y_i}q^t(i,r)\left(1-\exp(-\boldsymbol{\delta}^t\cdot\boldsymbol{\pi}_{i,r})\right)\right)\\
&\geq \sum_{r\neq y_i}q^t(i,r)\left(1-\exp(-\boldsymbol{\delta}^t\cdot\pi_{i,r})\right)=\sum_{r\neq y_i}q^t(i,r)\left[1-\exp\left(-\sum_j s_{i,r,j}\delta_j^t|\pi_{i,r,j}|\right)\right] \quad .
\end{aligned}$$

We now incorporate the update templates $\boldsymbol{a}$ while defining $\delta_j^t=a_jd_j^t$ and using the assumption that $\sum_j a_j|\pi_{i,r,j}|\leq 1$ for all examples i and classes r. The convexity of the exponential function, $\exp(\cdot)$, further gives us the bound

$$\Delta_t(i)\geq\sum_{r\neq y_i}q^t(i,r)\sum_j a_j|\pi_{i,r,j}|\left(1-\exp(-s_{i,r,j}d_j^t)\right) \quad .$$

Summing over all the examples, we have

$$\begin{aligned}
\Delta_t=\sum_{i=1}^m\Delta_t(i) &\geq \sum_j a_j\sum_{i=1}^m\sum_{r\neq y_i}q^t(i,r)|\pi_{i,r,j}|\left(1-\exp(-s_{i,r,j}d_j^t)\right)\\
&= \sum_j a_j\left[\sum_{i,r:\pi_{i,r,j}>0}q^t(i,r)\pi_{i,r,j}\left(1-e^{-d_j^t}\right)-\sum_{i,r:\pi_{i,r,j}<0}q^t(i,r)\pi_{i,r,j}\left(1-e^{d_j^t}\right)\right]\\
&= \sum_j a_j\sum_r\left[\mu_{r,j}^+\left(1-e^{-d_j^t}\right)+\mu_{r,j}^-\left(1-e^{d_j^t}\right)\right] \quad , \qquad (45)
\end{aligned}$$

where to obtain Eq. (45) we define

$$\mu_{r,j}^+=\sum_{i:\pi_{i,r,j}>0}q^t(i,r)|\pi_{i,r,j}| \quad \text{and} \quad \mu_{r,j}^-=\sum_{i:\pi_{i,r,j}<0}q^t(i,r)|\pi_{i,r,j}| \quad . \qquad (46)$$

We are now finally ready to tackle the task of bounding Eq. (4). We do so by diverging from the standard derivation of AdaBoost. Our proof exploits exploits the multi-vector construction in which we have weight vectors $\boldsymbol{w}_1, \ldots \boldsymbol{w}_k \in \mathbb{R}^n$, one for each of the k classes. Specifically, we define $\boldsymbol{h}$ by expanding the input instance as follows,

$$\boldsymbol{h}(\boldsymbol{x}_i, r) = \left[\underbrace{\mathbf{0} \, \cdots \, \mathbf{0}}_{r-1} \; \boldsymbol{x}_i^\top \; \underbrace{\mathbf{0} \, \cdots \, \mathbf{0}}_{k-r+1} \right] \quad . \tag{47}$$

This construction yields the following specific form for $\pi_{i,r,j}$:

$$\pi_{i,r,j} = h_j(\boldsymbol{x}_i, y_i) - h_j(\boldsymbol{x}_i, r) = \begin{cases} -x_{i,j} & \text{if } y_i \neq r \\ 0 & \text{if } y_i = r \end{cases} \quad .$$

Let us now examine $\mu^{\pm}_{r,j}$ in the context of this specific construction. For concreteness, we fix an example index i, a label index r, and a feature index j. If $r \neq y_i$, then $\pi_{i,r,j}$ appears once in either the sum constituting $\mu^+_{r,j}$ or $\mu^-_{r,j}$. Since $\pi_{i,r,j} = -x_{i,j}$ we readily obtain that $\pi_{i,r,j}$ contributes the term $q^t(i,r)(-x_{i,j})$ to $\mu^+_{r,j}$ when $x_{i,j} < 0$ and, analogously, the term $q^t(i,r)x_{i,j}$ is part of the summation constituting $\mu^-_{r,j}$ when $x_{i,j} > 0$. When $r = y_i$ the situtation is more involved. In this case, we have $h_j(\boldsymbol{x}_i, y_i) - h_j(\boldsymbol{x}_i, s) = h_j(\boldsymbol{x}_i, r) - h_j(\boldsymbol{x}_i, s) = x_{i,j}$ for $s \neq r$. Thus, for $s \neq r$, $\pi_{i,s,j} > 0$ ($\pi_{i,s,j} < 0$) when $x_{i,j} > 0$ ($x_{i,j} < 0$). For indices such that $r \neq y_i$, we have that $\pi_{i,s,j} = 0$. Combining these observations, we get that the importance-weighted correlations $\mu^+_{r,j}$ and $\mu^-_{r,j}$ for each class r can be written in terms of the examples $\boldsymbol{x}_i$ and the importance weights $q^t(\cdot)$ as follows:

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

Recapping, using the fact that $\delta^t_{r,j} = a_j d^t_{r,j}$ is the value added to the weight $w^t_{r,j}$, the change in the multiclass logistic loss of Eq. (4) can be lower bounded by

$$L(W^t) - L(W^{t+1}) = \Delta_t \geq \sum_{j=1}^n a_j \sum_{r=1}^k \left[\mu^+_{r,j} \left(1 - e^{-d^t_{r,j}}\right) + \mu^-_{r,j} \left(1 - e^{d^t_{r,j}}\right) \right] \quad .$$

Note that our definition of $\boldsymbol{h}$ from Eq. (47) gives that

$$\sum_j |\pi_{i,r,j}| = \|\boldsymbol{\pi}_{i,r}\|_1 = \|\boldsymbol{h}(\boldsymbol{x}_i, y_i) - \boldsymbol{h}(\boldsymbol{x}_i, r)\|_1 = \|\boldsymbol{x}_i\|_1 + \|-\boldsymbol{x}_i\|_1 = 2 \sum_{j=1}^n |x_{i,j}|$$

unless $r = y_i$, in which case the sum is 0. Thus, the allowable template vectors $\boldsymbol{a}$ in the progress bound are a (not necessarily strict) superset of the $\boldsymbol{a} \in \mathbb{R}^n_+$ such that $2 \sum_j a_j |x_{i,j}| \leq 1$ for all i. ■

Appendix B. Progress bounds for GradBoost

In this section we derive the progress bound for the gradient-based boosting algorithms. Our first bound for the binary logistic regression is a variation on a proof technique first given in Dekel et al. (2005). The second bound, for multiclass logistic regression, is new and requires a detailed examination of the Hessian of the logistic function in multiclass settings.

Proof of Lemma 7 To remind the reader, the update we perform takes the form, $\boldsymbol{w}^{t+1} = \boldsymbol{w}^t + \boldsymbol{\delta}^t(\boldsymbol{a})$ where $\delta_j^t = a_j d_j$ and $\boldsymbol{a}$ is set such that $\sum_{i,j} a_j x_{i,j}^2 \le 1$. Let us denote by $\boldsymbol{g}$ the gradient of the logistic function evaluated at $\boldsymbol{w}^t$, thus, $g_j = -\sum_{i=1}^m q^t(i) y_i x_{i,j}$. For clarity of the derivation below we omit the dependency on $\boldsymbol{a}$ and the iteration index t and denote by $\boldsymbol{\delta}$ the update vector. We now use the mean-value theorem and upper bound $L(\boldsymbol{w}^{t+1})$ using its second order Taylor expansion as follows,

$$L(\boldsymbol{w}^{t+1}) = L(\boldsymbol{w}^t + \boldsymbol{\delta}) = L(\boldsymbol{w}^t) + \nabla L(\boldsymbol{w}^t) \cdot \boldsymbol{\delta} + \frac{1}{2}\,\boldsymbol{\delta}^\top \left(\sum_{i=1}^m \tilde{q}^t(i)(1-\tilde{q}^t(i)) \boldsymbol{x}_i \boldsymbol{x}_i^\top \right) \boldsymbol{\delta} \quad , \tag{48}$$

where $\tilde{q}^t(i) = 1/(1+\exp(y_i \boldsymbol{x}_i \cdot (\boldsymbol{w}^t + \alpha\boldsymbol{\delta})))$ for some (unknown) $\alpha \in [0,1]$. By construction, $0 \le \tilde{q}^t(i) \le 1$, it is clear that $\tilde{q}^t(i)(1-\tilde{q}^t(i)) \le 1/4$. Thus, from Eq. (48) we get

$$\begin{aligned} L(\boldsymbol{w}^t + \boldsymbol{\delta}) &\le L(\boldsymbol{w}^t) + \nabla L(\boldsymbol{w}^t) \cdot \boldsymbol{\delta} + \frac{1}{8}\,\boldsymbol{\delta}^\top \left(\sum_{i=1}^m \boldsymbol{x}_i \boldsymbol{x}_i^\top \right) \boldsymbol{\delta} \\ &= L(\boldsymbol{w}^t) + \nabla L(\boldsymbol{w}^t) \cdot \boldsymbol{\delta} + \frac{1}{8} \sum_{i=1}^m (\boldsymbol{\delta} \cdot \boldsymbol{x}_i)^2 \quad . \end{aligned}$$

Recall that $\boldsymbol{\delta} = \boldsymbol{a} \otimes \boldsymbol{d}$. We now rewrite the inner-product $\boldsymbol{\delta} \cdot \boldsymbol{x}_i$ as $(\sqrt{\boldsymbol{a}} \otimes \boldsymbol{d}) \cdot (\sqrt{\boldsymbol{a}} \otimes \boldsymbol{x}_i)$ and use Cauchy-Schwarz inequality to further bound inner product,

$$(\boldsymbol{\delta} \cdot \boldsymbol{x}_i)^2 = \left((\sqrt{\boldsymbol{a}} \otimes \boldsymbol{d}) \cdot (\sqrt{\boldsymbol{a}} \otimes \boldsymbol{x}_i) \right)^2 \le \|\sqrt{\boldsymbol{a}} \otimes \boldsymbol{d}\|^2 \|\sqrt{\boldsymbol{a}} \otimes \boldsymbol{x}_i\|^2 \quad .$$

We can thus further bound $L(\boldsymbol{w}^{t+1})$ as follows,

$$\begin{aligned} L(\boldsymbol{w}^{t+1}) &\le L(\boldsymbol{w}^t) + \sum_{j=1}^n a_j d_j g_j + \frac{1}{8} \left(\sum_{j=1}^n a_j d_j^2 \right) \left(\sum_{i=1}^m \sum_{j=1}^n a_j x_{i,j}^2 \right) \\ &\le L(\boldsymbol{w}^t) + \sum_{j=1}^n a_j d_j g_j + \frac{1}{8} \sum_{j=1}^n a_j d_j^2 \quad , \end{aligned}$$

where we used the requirement on $\boldsymbol{a}$, namely, $\sum_{i,j} a_j x_{i,j}^2 \le 1$ to obtain the final bound. ■

Proof of Lemma 9 The derivation of the progress bound for the multiclass logistic loss of Eq. (4) is somewhat more complex than that for the binary logistic loss, and we introduce some new notation. For simplicity and cleanness, we prove the bound for the case when we update a single row from the matrix W. The update for the general case is a straightforward albeit tedious generalization which we discuss at the end of the proof. Formally, we update $w_{j,r}$ for each $r \in \{1, \ldots, k\}$. Therefore, $w_{j',s}^{t+1} = w_{j',s}^t$ for $j' \neq j$ and $\overline{\boldsymbol{w}}_j^{t+1} = \overline{\boldsymbol{w}}_j^t + \boldsymbol{\delta}$.

We start by examining the gradient for the multiclass logistic loss. First, let us rewrite the logistic loss as the negative log-likelihood for a multinomial logistic,

$$L(W) = -\sum_{i=1}^m \log\left(\frac{\exp(\boldsymbol{w}_{y_i} \cdot \boldsymbol{x}_i)}{\sum_r \exp(\boldsymbol{w}_r \cdot \boldsymbol{x}_i)} \right) = \sum_{i=1}^m \log\left(\sum_r \exp(\boldsymbol{w}_r \cdot \boldsymbol{x}_i) \right) - \boldsymbol{w}_{y_i} \cdot \boldsymbol{x}_i \quad .$$

We omit the iteration index t and define $q(i,r)$ as in Eq. (44). Denoting by ∇_r the partial derivative of L with respect to $w_{r,j}$, we get that

$$\nabla_r = \sum_{i=1}^m q(i,r)\, x_{i,j} - \sum_{i:y_i=r} x_{i,j} \quad .$$

The diagonal elements and off-diagonal elements of the Hessian, denoted $\nabla_{r,s}$ and $\nabla_{r,r}$ respectively, are likewise simple to derive,

$$\nabla_{r,s} = -\sum_{i=1}^{m} q(i,r)q(i,s)x_{i,j}^2 \quad \text{and} \quad \nabla_{r,r} = \sum_{i=1}^{m} q(i,r)x_{i,j}^2 - q^2(i,r)x_{i,j}^2 \ .$$

We now use again the mean value theorem with a second order Taylor expansion of the loss to obtain that,

$$\begin{aligned} L(W^{t+1}) &= L(W^t) + \sum_{r=1}^{k} \delta_r \sum_{i=1}^{m} (q(i,r) - \mathbf{1}\{r = y_i\})x_{i,j} \\ &\quad + \frac{1}{2}\left(\sum_{i=1}^{m} x_{i,j}^2 \sum_{r=1}^{k} \tilde{q}(i,r)\delta_r^2 - \sum_{i=1}^{m} x_{i,j}^2 \sum_{r=1}^{k}\sum_{s=1}^{k} \tilde{q}(i,r)\tilde{q}(i,s)\delta_r\delta_s \right) \ , \end{aligned} \tag{49}$$

where

$$\tilde{q}(i,r) = \frac{\exp((\boldsymbol{w}_r + \alpha\delta_r\boldsymbol{e}_j)\cdot\boldsymbol{x}_i)}{\sum_s \exp((\boldsymbol{w}_s + \alpha\delta_s\boldsymbol{e}_j)\cdot\boldsymbol{x}_i)} \ ,$$

for some $\alpha \in [0,1]$. Here $\boldsymbol{e}_j$ denotes the all zero vector except for a 1 in the j^{th} component. Examining the Hessian term from Eq. (49) allows us to make a few further useful simplifications to these bounds. First, we know that $\sum_r q(i,r) = 1$ and $q(i,r) \geq 0$ for all r. Second, we know that

$$\sum_{r=1}^{k}\sum_{s=1}^{k} q(i,r)q(i,s)\delta_r\delta_s = \sum_{r=1}^{k} q(i,r)\delta_r \sum_{s=1}^{k} q(i,s)\delta_s(i) = \left(\sum_{r=1}^{k} q(i,r)\delta_r\right)^2 \geq 0 \ .$$

Denote by $\boldsymbol{g}$ the vector whose rth component is ∇_r, that is, $g_r = \sum_{i=1}^{m}(q(i,r) - \mathbf{1}\{r = y_i\})x_{i,j}$. We readily can upper bound $L(W^{t+1})$ using Eq. (49) as follows,

$$L(W^{t+1}) \leq L(W^t) + \sum_{r=1}^{k} \delta_r g_r + \frac{1}{2}\sum_{r=1}^{k} \delta_r^2 \sum_{i=1}^{m} x_{i,j}^2 \ .$$

We can get a tighter bound, however, by examining more closely the last term from Eq. (49). To do so, we note that it is actually a sum of terms of the form $\boldsymbol{\delta}^\top \operatorname{diag}(\boldsymbol{v})\boldsymbol{\delta} - \boldsymbol{\delta}^\top \boldsymbol{v}\boldsymbol{v}^\top\boldsymbol{\delta}$, where $\boldsymbol{v} \in \Delta_k$, the k-dimensional probability simplex. Thus, if we upper bound the maximum eigenvalue of $\operatorname{diag}(\boldsymbol{v}) - \boldsymbol{v}\boldsymbol{v}^\top$ for any $\boldsymbol{v} \in \Delta_k$, we get a tighter bound on $L(W^{t+1})$. We use the Geršgorin circle theorem (Horn and Johnson, 1985, Theorem 6.1.1), which says that the eigenvalues of a matrix A are all within disks defined by $A_{ii} \pm \sum_{j\neq i} |A_{ji}|$. Examining $\operatorname{diag}(\boldsymbol{v}) - \boldsymbol{v}\boldsymbol{v}^\top$, we have off-diagonal elements of the form $v_i v_j$ and on-diagonal elements $v_i(1-v_i)$. Thus, all the eigenvalues are bounded above by

$$\max_{i\in[n]} \left\{ v_i(1-v_i) + v_i \sum_{j\neq i} v_j \right\} = \max_{i\in[n]} \{v_i(1-v_i) + v_i(1-v_i)\} \leq \sup_{v\in[0,1]} 2v(1-v) = \frac{1}{2} \ .$$

The inequality above is due to the fact that $v_i \in [0,1]$ for all i. Moreover, this bound is tight since the vector $\boldsymbol{v} = [\frac{1}{2}\ \frac{1}{2}\ 0\ \cdots\ 0]^\top$ is associated with an eigenvalue of one half. We therefore obtain the following bound on the loss,

$$L(W^{t+1}) \leq L(W^t) + \sum_{r=1}^{k} \delta_r g_{r,j} + \frac{1}{4}\sum_{r=1}^{k} \delta_r^2 \sum_{i=1}^{m} x_{i,j}^2 \ . \tag{50}$$

Finally, in the general case where we associate a different update vector $\boldsymbol{\delta}_j$ with each row of W, we can define a template vector $\boldsymbol{a} \in \mathbb{R}^n$ such that $\sum_{i,j} a_j x_{i,j}^2 \leq 1$ and generalize the above derivation (using convexity and the Cauchy-Schwarz inequality as in the proof of lemma 7) for the case where multiple rows are being updated simultaneously, which gives

$$L(W^{t+1}) \leq L(W^t) + \sum_{j:a_j>0} \sum_{r=1}^{k} \delta_{j,r} g_{r,j} + \frac{1}{4} \sum_{j:a_j>0} \sum_{r=1}^{k} \frac{\delta_{j,r}^2}{a_j} \quad .$$

■

Appendix C. Convergence proofs

Before focusing on the proof of the main convergence theorem, we prove a technical lemma on the continuity of solutions to the different boosting updates that simplifies our proofs. We then use Lemma 12 to guarantee that the change in the loss Δ_t for any of the boosting updates is a continuous function of the parameters $\boldsymbol{w}$. This property allows us to lower bound the change in the loss over some compact subset of the parameters in which the optimal point for any of the logistic losses does not lie, guaranteeing progress of the boosting algorithms.

Lemma 12 *Let M and X be compact spaces. Let $f : M \times X \to \mathbb{R}$ be continuous function in $\boldsymbol{\mu}$ and strictly convex in $\boldsymbol{x}$. Let $\boldsymbol{\mu}_j \in M$ for $j \in \{1, 2\}$ and define $\boldsymbol{x}_j^\star = \operatorname{argmin}_{\boldsymbol{x} \in X} f(\boldsymbol{\mu}_j, \boldsymbol{x})$. Given $\varepsilon > 0$, there exists a δ such that if $\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2\| \leq \delta$ then $\|\boldsymbol{x}_1^\star - \boldsymbol{x}_2^\star\| \leq \varepsilon$.*

Proof of Lemma 12 From the strict convexity of f in $\boldsymbol{x}$, we know that if $\|\boldsymbol{x}_1^\star - \boldsymbol{x}_2^\star\| > \varepsilon$, then there is a δ such that

$$f(\boldsymbol{\mu}_1, \boldsymbol{x}_1^\star) < f(\boldsymbol{\mu}_1, \boldsymbol{x}_2^\star) - \delta \quad \text{and} \quad f(\boldsymbol{\mu}_2, \boldsymbol{x}_2^\star) < f(\boldsymbol{\mu}_2, \boldsymbol{x}_1^\star) - \delta.$$

Now, if $f(\boldsymbol{\mu}_1, \boldsymbol{x}_1^\star) \leq f(\boldsymbol{\mu}_2, \boldsymbol{x}_2^\star)$, then

$$f(\boldsymbol{\mu}_1, \boldsymbol{x}_1^\star) - f(\boldsymbol{\mu}_2, \boldsymbol{x}_1^\star) < f(\boldsymbol{\mu}_1, \boldsymbol{x}_1^\star) - f(\boldsymbol{\mu}_2, \boldsymbol{x}_2^\star) - \delta \leq f(\boldsymbol{\mu}_2, \boldsymbol{x}_2^\star) - f(\boldsymbol{\mu}_2, \boldsymbol{x}_2^\star) - \delta = -\delta$$

or $\delta < f(\boldsymbol{\mu}_2, \boldsymbol{x}_1^\star) - f(\boldsymbol{\mu}_1, \boldsymbol{x}_1^\star)$. Likewise, if $f(\boldsymbol{\mu}_2, \boldsymbol{x}_2^\star) \leq f(\boldsymbol{\mu}_1, \boldsymbol{x}_1^\star)$,

$$f(\boldsymbol{\mu}_2, \boldsymbol{x}_2^\star) - f(\boldsymbol{\mu}_1, \boldsymbol{x}_2^\star) < f(\boldsymbol{\mu}_2, \boldsymbol{x}_2^\star) - f(\boldsymbol{\mu}_1, \boldsymbol{x}_1^\star) - \delta \leq f(\boldsymbol{\mu}_1, \boldsymbol{x}_1^\star) - f(\boldsymbol{\mu}_1, \boldsymbol{x}_1^\star) - \delta = -\delta$$

so that $\delta < f(\boldsymbol{\mu}_2, \boldsymbol{x}_2^\star) - f(\boldsymbol{\mu}_1, \boldsymbol{x}_2^\star)$. In either case, we have an $\boldsymbol{x}$ such that $|f(\boldsymbol{\mu}_1, \boldsymbol{x}) - f(\boldsymbol{\mu}_2, \boldsymbol{x})| > \delta$. The contrapositive of what has been shown is that if $|f(\boldsymbol{\mu}_1, \boldsymbol{x}) - f(\boldsymbol{\mu}_2, \boldsymbol{x})| \leq \delta$ for all $\boldsymbol{x} \in X$, then $\|\boldsymbol{x}_1^\star - \boldsymbol{x}_2^\star| \leq \varepsilon$.

■

Put informally, the function $\operatorname{argmin}_{\boldsymbol{x}} f(\boldsymbol{\mu}, \boldsymbol{x})$ is continuous in $\boldsymbol{\mu}$. In the lemma, $\boldsymbol{\mu}$ stands for the correlations variable $\mu_j^\pm$, which are continuous functions of the current weights $\boldsymbol{w}^t$, and $\boldsymbol{x}$ represents $\boldsymbol{\delta}^t$.

Lemma 13 *(a) The iterates $\boldsymbol{w}^t$ generated by the algorithms lie in compact spaces. (b) The progress bounds for AdaBoost or GradBoost presented in this paper satisfy lemma 12.*

Proof We prove the lemma for ℓ_1-regularized AdaBoost, noting that the proofs for ℓ_1/ℓ_∞-regularized multiclass and multitask AdaBoost, ℓ_1-regularized GradBoost, ℓ_1/ℓ_∞-regularized multiclass and multitask GradBoost, and ℓ_1/ℓ_2-regularized GradBoost are essentially identical. First, we show that each of the algorithms' iterates lie in compact spaces by showing that the loss for each is monotonically non-increasing. Hence, a norm of the weights for each of the losses is bounded.

For ℓ_1-regularized AdaBoost, we examine Eq. (42) and Eq. (6). We have $Q(\boldsymbol{w}^t) - Q(\boldsymbol{w}^{t+1}) \geq \sum_j \Delta_t(j) \geq 0$, since

$$\begin{aligned}
\Delta_t(j) &\geq \sup_{\delta_j} \left[a_j \mu_j^+ \left(1 - e^{-\delta_j / a_j}\right) + a_j \mu_j^- \left(1 - e^{\delta_j / a_j}\right) - \lambda \left|\delta_j + w_j^t\right| + \lambda \left|w_j^t\right| \right] \\
&\geq a_j \mu_j^+ (1 - 1) + a_j \mu_j^- (1 - 1) - \lambda \left|w_j^t\right| + \lambda \left|w_j^t\right| = 0 \ .
\end{aligned}$$

As at every step we choose the δ_j that achieves the supremum, we also have that $Q(\mathbf{0}) = m \log 2 \geq Q(\boldsymbol{w}^t) \geq \lambda \left\|\boldsymbol{w}^t\right\|_1$. In summary, for all t, we bound the ℓ_1-norm of $\boldsymbol{w}^t$ by $m \log 2 / \lambda$, which guarantees that $\boldsymbol{w}^t$ are in a compact space. The arguments for all the other algorithms are similar.

We now prove assertion (b). Each bound on the change in one of the losses is a function of the importance weights $q(i)$ (through the gradient or the $\mu_j^\pm$ terms). The importance weights are continuous functions of $\boldsymbol{w}$ and the iterates for each algorithm lie in a compact space. Thus, each progress bound is a continuous function of $\boldsymbol{w}$. Note also that the gradient terms and the $\mu_j^\pm$ terms also lie in compact spaces since they are the continuous image of a compact set. Furthermore, each algorithm's update of $\boldsymbol{w}^t$ satisfies $\lambda \|\boldsymbol{w}^{t+1}\|_q \leq Q(\mathbf{0})$. Concretely, it means that $\lambda \|\boldsymbol{w}^t + \boldsymbol{\delta}^t\|_q \leq Q(\mathbf{0})$ and therefore

$$\lambda \|\boldsymbol{\delta}^t\|_q \leq \lambda \|\boldsymbol{\delta}^t + \boldsymbol{w}^t\|_q + \lambda \|\boldsymbol{w}^t\|_q \leq 2Q(\mathbf{0}) \ .$$

Lastly, by inspection it is clear that each of the progress bounds is strictly convex. ■

Lemma 14 (General Convergence) *Let $Q : \mathbb{R}^n \to \mathbb{R}_+$ be a continuous convex function and $A : \mathbb{R}^n \to \mathbb{R}$ be a continuous function. Let $\boldsymbol{w}^1, \ldots, \boldsymbol{w}^t, \ldots$ be a sequence that satisfies the following conditions:*

(a) The sequence lies in a compact space Ω

(b) $Q(\boldsymbol{w}^{t+1}) - Q(\boldsymbol{w}^t) \leq A(\boldsymbol{w}^t) \leq 0$

(c) If $A(\boldsymbol{w}) = 0$ then $\boldsymbol{w}$ is a fixed point and hence a minimum for the loss Q.

Then $\boldsymbol{w}^t \to \boldsymbol{w}^\star$, where $\boldsymbol{w}^\star = \arg\min_{\boldsymbol{w}} Q(\boldsymbol{w})$.

Proof Let $\Omega^\star$ denote the set of optimal points for Q, and assume for the sake of contradiction that the sequence $\boldsymbol{w}^1, \boldsymbol{w}^2, \ldots$ never enters the open ball $B(\Omega^\star, \gamma)$ of radius $\gamma > 0$ around $\Omega^\star$. The set $\Omega \setminus B(\Omega^\star, \gamma)$ is compact, so $-A$ must attain a minimum $\theta > 0$ on it. This is a contradiction, since assumption (b) would imply that $Q(\boldsymbol{w}^{t+1}) \leq Q(\boldsymbol{w}^t) - \theta$ for all t.

■

Proof of Theorem 11 We show that each variant of regularized AdaBoost and GradBoost presented in this paper satisfies Lemma 14 and has an auxiliary function A. The existence of A is actually a consequence of lemma 13, which bounds the change in the loss of each algorithm. For each variant of GradBoost and AdaBoost, we set $A(\boldsymbol{w}^t)$ to be the value of the supremum of the bound on the change in the loss for each algorithm at the point $\boldsymbol{w}^t$ (see for instance Eq. (33)). The arguments in Lemma 13 imply that we have $Q(\boldsymbol{w}^t) - Q(\boldsymbol{w}^{t+1}) \geq A(\boldsymbol{w}^t) \geq 0$. The A thus constructed are also continuous functions of $\boldsymbol{w}$ by Lemma 13, since the $\boldsymbol{\delta}^t$ are continuous in $\boldsymbol{w}^t$. In addition, $A(\boldsymbol{w}^t) = 0$ only when $\boldsymbol{\delta}^t = \mathbf{0}$, as each A is the supremum of a strictly concave function which attains a value of zero when $\boldsymbol{\delta} = \mathbf{0}$.

An inspection of subgradient conditions for optimality (Bertsekas, 1999) gives that for each algorithm, if $A(\boldsymbol{w}^t) = 0$, then $\boldsymbol{w}^t$ is an optimal point for the particular loss. For example, consider the ℓ_1-regularized logistic loss of Eq. (2). Suppose that for every template $\boldsymbol{a}$, no progress can be made and $\delta_j = 0$ for all j. Then, for all j we get that 0 belongs in the sub-differential set of the change in the loss at $\delta_j = 0$, namely,

$$0 \in -\mu_j^+ + \mu_j^- + \lambda\partial_{\delta_j}|w_j + \delta_j|\big|_{\delta_j=0} = -\mu_j^+ + \mu_j^- + \lambda\partial|w_j| \ \ .$$

Expanding the right hand side of the above equation, we get

$$\begin{aligned} -\mu_j^+ + \mu_j^- + \lambda\partial|w_j| &= -\sum_{i:x_{i,j}y_i>0} q(i)\,|x_{i,j}| + \sum_{i:x_{i,j}y_i<0} q(i)\,|x_{i,j}| + \lambda\partial|w_j| \\ &= -\sum_{i:x_{i,j}y_i>0} \frac{y_i x_{i,j}}{1+e^{y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i)}} - \sum_{i:x_{i,j}y_i<0} \frac{y_i x_{i,j}}{1+e^{y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i)}} + \lambda\partial|w_j| \\ &= -\sum_{i=1}^{m} \frac{1}{1+e^{y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i)}} y_i x_{i,j} + \lambda\partial|w_j| = \frac{\partial}{\partial w_j}L(\boldsymbol{w}) + \partial_j\lambda\left\|\boldsymbol{w}\right\|_1 \quad , \end{aligned}$$

where ∂_j is the subdifferential of $Q(\boldsymbol{w})$ for the j^{th} element of $\boldsymbol{w}$. Thus, when the change in the loss is 0, we have that 0 is in the subdifferential of $Q(\boldsymbol{w})$ for each w_j and therefore $\boldsymbol{w}$ is an optimum of loss function. Lemma 13 guarantees that each algorithm's iterates lie in a compact space, and therefore Lemma 14 applies. ■

Appendix D. Experimental Setup

Feature Selection for ℓ_2-Regularized Logistic Regression: Here we describe the feature scoring scheme to select the features used by ℓ_2-regularized logistic regression. We first learned the base regressor (which can be done exactly), then we chose the 800 features that gave the best decrease in the loss. We show next that the latter stage reduces to selection based on an information gain criterion. First, for selecting the bias weight b, we minimize

$$\sum_{i=1}^{m} \log(1+\exp(-y_i b)) = m^+ \log(1+\exp(-b)) + m^- \log(1+\exp(b))$$

where m^+ and m^- are the number of positive and negative examples, respectively. Taking the derivative and setting the result to zero, we get

$$-m^+\frac{1}{1+e^b} + m^-\frac{e^b}{1+e^b} = 0 \ \ \Rightarrow \ \ -m^+ + m^- e^b = 0 \ \ \Rightarrow \ \ b = \log\left(\frac{m^+}{m^-}\right). \tag{51}$$

For the Reuters data, our features took values in $\{0,1\}$. Let us denote by n_j^+ the number of examples for which feature j was 1 and $y_i = 1$ and by n_j^- the examples for which feature j was 1 while y_i was -1. Using these definitions, we want to minimize, w.r.t α, the following expression,

$$\sum_{i:x_{i,j}\neq 0} \log(1+\exp(-y_i(b+\alpha x_{i,j}))) = n_j^+ \log(1+\exp(-b-\alpha)) + n_j^- \log(1+\exp(b+\alpha)).$$

Again taking the derivative w.r.t α, we need solve $-\frac{n_j^+}{1+e^{b+\alpha}} + \frac{n_j^-}{1+e^{-b-\alpha}} = 0$. Note that this expression takes the same form as Eq. (51) for b. Thus, we obtain that $b+\alpha = \log\left(\frac{n_j^+}{n_j^-}\right)$. To score a feature, we simply compute the score Δ_j, which is, ignoring the terms that only have to do with the bias b,

$$\begin{aligned}
\Delta_j &= n_j^+ \log\left(1 + \frac{n_j^-}{n_j^+}\right) + n_j^- \log\left(1 + \frac{n_j^+}{n_j^-}\right) - n_j^+ \log\left(1 + \frac{m^-}{m^+}\right) - n_j^- \log\left(1 + \frac{m^+}{m^-}\right) \\
&= -n_j^+ \log\left(\frac{n_j^+}{n_j^+ + n_j^-}\right) - n_j^- \log\left(\frac{n_j^-}{n_j^+ + n_j^-}\right) + n_j^+ \log\left(\frac{m^+}{m^+ + m^-}\right) + n_j^- \log\left(\frac{m^-}{m^+ + m^-}\right) \\
&= -D_{\mathrm{KL}}\left(\frac{n_j^+}{n_j^+ + n_j^-} \middle\| \frac{m^+}{m^+ + m^-}\right)(n_j^+ + n_j^-) \quad .
\end{aligned}$$

To recap, we choose the features with maximum KL-divergence from the base distribution of positive and negative examples, weighted by the number of times the feature is on.