Proceedings of the Thirteenth Annual Conference on Computational Learning Theory, 2000.

Logistic Regression, AdaBoost and Bregman Distances

Michael Collins
AT&T Labs − Research
Shannon Laboratory
180 Park Avenue, Room A253
Florham Park, NJ 07932
mcollins@research.att.com

Robert E. Schapire
AT&T Labs − Research
Shannon Laboratory
180 Park Avenue, Room A203
Florham Park, NJ 07932
schapire@research.att.com

Yoram Singer
School of Computer Science & Engineering
Hebrew University, Jerusalem 91904, Israel
singer@cs.huji.ac.il

Abstract. We give a unified account of boosting and logistic regression in which each learning problem is cast in terms of optimization of Bregman distances. The striking similarity of the two problems in this framework allows us to design and analyze algorithms for both simultaneously, and to easily adapt algorithms designed for one problem to the other. For both problems, we give new algorithms and explain their potential advantages over existing methods. These algorithms can be divided into two types based on whether the parameters are iteratively updated sequentially (one at a time) or in parallel (all at once). We also describe a parameterized family of algorithms which interpolates smoothly between these two extremes. For all of the algorithms, we give convergence proofs using a general formalization of the auxiliary-function proof technique. As one of our sequential-update algorithms is equivalent to AdaBoost, this provides the first general proof of convergence for AdaBoost. We show that all of our algorithms generalize easily to the multiclass case, and we contrast the new algorithms with iterative scaling. We conclude with preliminary experimental results.

1 INTRODUCTION

We give a unified account of boosting and logistic regression in which we show that both learning problems can be cast in terms of optimization of Bregman distances. In our framework, the two problems become extremely similar, the only real difference being in the choice of Bregman distance: unnormalized relative entropy for boosting, and binary relative entropy for logistic regression.

The fact that the two problems are so similar in our framework allows us to design and analyze algorithms for both simultaneously. We are now able to borrow methods from the maximum-entropy literature for logistic regression and apply them to the exponential loss used by AdaBoost, especially convergence-proof techniques. Conversely, we can now easily adapt boosting methods to the problem of minimizing the logistic loss used in logistic regression. The result is a family of new algorithms for both problems together with convergence proofs for the new algorithms as well as AdaBoost.

For both AdaBoost and logistic regression, we attempt to choose the parameters or weights associated with a given family of functions called *features* or *weak hypotheses*. AdaBoost works by sequentially updating these parameters one by one, whereas methods for logistic regression, most notably iterative scaling [9, 10], are iterative but update all parameters in parallel on each iteration.

Our first new algorithm is a method for optimizing the exponential loss using parallel updates. It seems plausible that a parallel-update method will often converge faster than a sequential-update method, provided that the number of features is not so large as to make parallel updates infeasible. Some preliminary experiments suggest that this is the case.

Our second algorithm is a parallel-update method for the logistic loss. Although parallel-update algorithms are well known for this function, the updates that we derive are new, and preliminary experiments indicate that these new updates may also be much faster. Because of the unified treatment we give to the exponential and logistic loss functions, we are able to present and prove the convergence of the algorithms for these two losses simultaneously. The same is true for the other algorithms presented in this paper as well.

We next describe and analyze sequential-update algorithms for the two loss functions. For exponential loss, this algorithm is equivalent to the AdaBoost algorithm of Freund and Schapire [13]. By viewing the algorithm in our framework, we are able to prove that AdaBoost correctly converges to the minimum of the exponential loss function. This is a new result: Although Kivinen and Warmuth [16] and Mason et al. [19] have given convergence proofs for AdaBoost, their proofs depend on assumptions about the given minimization problem which may not hold in all cases. Our proof holds in general without assumptions.

Our unified view leads instantly to a sequential-update algorithm for logistic regression that is only a minor modification of AdaBoost and which is very similar to one proposed by Duffy and Helmbold [12]. Like AdaBoost, this algorithm can be used in conjunction with any classification algorithm, usually called the weak learning algorithm, that can accept a distribution over examples and return a weak hypothesis with low error rate with respect to the distribution. However, this new algorithm provably minimizes the logistic loss rather than the arguably less natural exponential loss used by AdaBoost.

Another potentially important advantage of this new algorithm is that the weights that it places on examples are bounded in $[0, 1]$. This suggests that it may be possible to use the new algorithm in a setting in which the boosting algorithm selects examples to present to the weak learning algorithm by filtering a stream of examples (such as a very large dataset). As pointed out by Watanabe [22] and Domingo and Watanabe [11], this is not possible with AdaBoost since its weights may become extremely large. They provide a modification of AdaBoost for this purpose in which the weights are truncated at 1. The new algorithm may be a viable and cleaner alternative.

We next describe a parameterized family of iterative algorithms that includes both parallel- and sequential-update algorithms and that also interpolates smoothly between the two extremes. The convergence proof that we give holds for this entire family of algorithms.

Although most of this paper considers only the binary case in which there are just two possible labels associated with each example, it turns out that the multiclass case requires no additional work. That is, all of the algorithms and convergence proofs that we give for the binary case turn out to be directly applicable to the multiclass case without modification.

For comparison, we also describe the generalized iterative scaling algorithm of Darroch and Ratcliff [9]. In rederiving this procedure in our setting, we are able to relax one of the main assumptions usually required by this algorithm.

The paper is organized as follows: Section 2 describes the boosting and logistic regression models as they are usually formulated. Section 3 gives background on optimization using Bregman distances, and Section 4 then describes how boosting and logistic regression can be cast within this framework. Section 5 gives our parallel-update algorithms and proofs of their convergence, while Section 6 gives the sequential-update algorithms and convergence proofs. The parameterized family of iterative algorithms is described in Section 7. The extension to multiclass problems is given in Section 8. In Section 9, we contrast our methods with iterative scaling. In Section 10, we give some preliminary experiments.

Previous work. Variants of our sequential-update algorithms fit into the general family of "arcing" algorithms presented by Breiman [4, 3], as well as Mason et al.'s "AnyBoost" family of algorithms [19]. The information-geometric view that we take also shows that the algorithms we study, including AdaBoost, fit into a family of algorithms described in 1967 by Bregman [2] for satisfying a set of constraints.

Our work is based directly on the general setting of Lafferty, Della Pietra and Della Pietra [18] in which one attempts to solve optimization problems based on general Bregman distances. They gave a method for deriving and analyzing parallel-update algorithms in this setting through the use of auxilliary functions. All of our algorithms and convergence proofs are based on this method.

Our work builds on several previous papers which have compared boosting approaches to logistic regression. Friedman, Hastie and Tibshirani [14] first noted the similarity between the boosting and logistic regression loss functions, and derived the sequential-update algorithm LogitBoost for the logistic loss. However, unlike our algorithm, theirs requires that the weak learner solve least-squares problems rather than classification problems. Another sequential-update algorithm for a different but related problem was proposed by Cesa-Bianchi, Krogh and Warmuth [5].

Duffy and Helmbold [12] gave conditions under which a loss function gives a boosting algorithm. They showed that minimizing logistic loss does lead to a boosting algorithm in the PAC sense, which suggests that our algorithm for this problem, which is very close to theirs, may turn out also to have the PAC boosting property.

Lafferty [17] went further in studying the relationship between logistic regression and the exponential loss through the use of a family of Bregman distances. However, the setting described in his paper apparently cannot be extended to precisely include the exponential loss. The use of Bregman distances that we describe has important differences leading to a natural treatment of the exponential loss and a new view of logistic regression.

Our work builds heavily on that of Kivinen and Warmuth [16] who, along with Lafferty, were the first to make a connection between AdaBoost and information geometry. They showed that the update used by AdaBoost is a form of "entropy projection." However, the Bregman distance that they used differed slightly from the one that we have chosen (normalized relative entropy rather than unnormalized relative entropy) so that AdaBoost's fit in this model was not quite complete; in particular, their convergence proof depended on assumptions that do not hold in general. Kivinen and Warmuth also described updates for general Bregman distances including, as one of their examples, the Bregman distance that we use to capture logistic regression.

2 BOOSTING, LOGISTIC MODELS AND LOSS FUNCTIONS

Let $S = \langle (x_1, y_1), \ldots, (x_m, y_m) \rangle$ be a set of training examples where each instance x_i belongs to a domain or instance space $\mathcal{X}$, and each label $y_i \in \{-1, +1\}$.

We assume that we are also given a set of real-valued functions on $\mathcal{X}$, $h_1, \ldots, h_n$. Following convention in the MaxEnt literature, we call these functions *features*; in the boosting literature, these would be called *weak* or *base hypotheses*.

We study the problem of approximating the y_i's using a linear combination of features. That is, we are interested in the problem of finding a vector of parameters $\boldsymbol{\lambda} \in \mathbb{R}^n$ such that $f_{\lambda}(x_i) = \sum_{j=1}^{n} \lambda_j h_j(x_i)$ is a "good approximation" of y_i. How we measure the goodness of such an approximation varies with the task that we have in mind.

For classification problems, it is natural to try to match the sign of $f_{\lambda}(x_i)$ to y_i, that is, to attempt to minimize

$$\sum_{i=1}^{m} [\![y_i f_{\lambda}(x_i) \le 0]\!] \tag{1}$$

where $[\![\pi]\!]$ is 1 if π is true and 0 otherwise. Although minimization of the number of classification errors may be a worthwhile goal, in its most general form, the problem is intractable (see, for instance, [15]). It is therefore often advantageous to instead minimize some other nonnegative loss function. For instance, the boosting algorithm AdaBoost [13, 20] is based on the exponential loss

$$\sum_{i=1}^{m} \exp\left(-y_i f_{\lambda}(x_i)\right). \tag{2}$$

It can be verified that Eq. (1) is upper bounded by Eq. (2); however, the latter loss is much easier to work with as demonstrated by AdaBoost. Briefly, on each of a series of rounds, AdaBoost uses an oracle or subroutine called the weak learning algorithm to pick one feature (weak hypothesis) h_j, and the associated parameter λ_j is then updated. It has been noted by Breiman [3, 4] and various later authors that both of these steps are done in such a way as to (approximately) cause the greatest decrease in the exponential loss. In this paper, we show for the first time that AdaBoost is in fact a provably effective method for finding parameters $\boldsymbol{\lambda}$ which minimize the exponential loss (assuming the weak learner always chooses the "best" h_j).

We also give an entirely new algorithm for minimizing exponential loss in which, on each round, *all* of the parameters λ_j are updated in parallel rather than one at a time. Our hope is that this parallel-update algorithm will be faster than the sequential-update algorithm; see Section 10 for preliminary experiments in this regard.

Instead of using f_λ as a classification rule, we might instead postulate that the y_i's were generated stochastically as a function of the x_i's and attempt to use $f_\lambda(x)$ to estimate the probability of the associated label y. A very natural and well-studied way of doing this is to pass f_λ through a logistic function, that is, to use the estimate

$$\hat{\Pr}[y = +1 \mid x] = \frac{1}{1 + e^{-f_\lambda(x)}}.$$

The likelihood of the labels occuring in the sample then is

$$\prod_{i=1}^{m} \frac{1}{1 + \exp\left(-y_i f_\lambda(x_i)\right)}.$$

Maximizing this likelihood then is equivalent to minimizing the log loss of this model

$$\sum_{i=1}^{m} \ln\left(1 + \exp\left(-y_i f_\lambda(x_i)\right)\right). \tag{3}$$

Generalized and improved iterative scaling [9, 10] are popular parallel-update methods for minimizing this loss. In this paper, we give an alternative parallel-update algorithm which we compare to iterative scaling techniques in preliminary experiments in Section 10.

3 BREGMAN-DISTANCE OPTIMIZATION

In this section, we give background on optimization using Bregman distances. This will form the unifying basis for our study of boosting and logistic regression. The particular set-up that we follow is taken primarily from Lafferty, Della Pietra and Della Pietra [18].

Let $F : \Delta \to \mathbb{R}$ be a continuously differentiable and strictly convex function defined on a closed, convex set $\Delta \subseteq \mathbb{R}_+^m$. The Bregman distance associated with F is defined for $\mathbf{p}, \mathbf{q} \in \Delta$ to be

$$B_F\left(\mathbf{p} \parallel \mathbf{q}\right) \doteq F(\mathbf{p}) - F(\mathbf{q}) - \nabla F(\mathbf{q}) \cdot (\mathbf{p} - \mathbf{q}).$$

For instance, when

$$F(\mathbf{p}) = \sum_{i=1}^{m} p_i \ln p_i, \tag{4}$$

B_F is the (unnormalized) relative entropy

$$\mathrm{D}_U\left(\mathbf{p} \parallel \mathbf{q}\right) = \sum_{i=1}^{m} \left(p_i \ln\left(\frac{p_i}{q_i}\right) + q_i - p_i\right).$$

It can be shown that, in general, every Bregman distance is nonnegative and is equal to zero if and only if its two arguments are equal.

There is a natural optimization problem that can be associated with a Bregman distance, namely, to find the vector $\mathbf{p} \in \Delta$ that is closest to a given vector $\mathbf{q}_0 \in \Delta$ subject to a set of linear constraints. These constraints are specified by an $m \times n$ matrix $\mathbf{M}$ and a vector $\tilde{\mathbf{p}} \in \Delta$. The vectors $\mathbf{p}$ satisfying these constraints are those for which $\mathbf{p}^{\mathrm{T}}\mathbf{M} = \tilde{\mathbf{p}}^{\mathrm{T}}\mathbf{M}$. Thus, the problem is to find

$$\arg\min_{\mathbf{p} \in \mathcal{P}} B_F\left(\mathbf{p} \parallel \mathbf{q}_0\right)$$

where

$$\mathcal{P} \doteq \left\{\mathbf{p} \in \Delta : \mathbf{p}^{\mathrm{T}}\mathbf{M} = \tilde{\mathbf{p}}^{\mathrm{T}}\mathbf{M}\right\}. \tag{5}$$

The "convex dual" of this problem gives an alternative formulation. Here, the problem is to find the vector of a particular form that is closest to a given vector $\tilde{\mathbf{p}}$. The form of such vectors is defined via the *Legendre transform*, written $\mathbf{v} \circ_F \mathbf{q}$ (or simply $\mathbf{v} \circ \mathbf{q}$ when F is clear from context):

$$\mathbf{v} \circ_F \mathbf{q} \doteq \arg\min_{\mathbf{p} \in \Delta} \left(B_F\left(\mathbf{p} \parallel \mathbf{q}\right) + \mathbf{v} \cdot \mathbf{p}\right).$$

Using calculus, this can be seen to be equivalent to

$$\nabla F(\mathbf{v} \circ \mathbf{q}) = \nabla F(\mathbf{q}) - \mathbf{v}. \tag{6}$$

For instance, when B_F is unnormalized relative entropy, it can be verified using calculus that

$$(\mathbf{v} \circ \mathbf{q})_i = q_i e^{-v_i}. \tag{7}$$

From Eq. (6), it is useful to note that

$$\mathbf{v} \circ (\mathbf{w} \circ \mathbf{q}) = (\mathbf{v} + \mathbf{w}) \circ \mathbf{q}. \tag{8}$$

For a given $m \times n$ matrix $\mathbf{M}$ and vector $\mathbf{q}_0 \in \Delta$, we consider vectors obtained by taking the Legendre transform of a linear combination of columns of $\mathbf{M}$ with the vector $\mathbf{q}_0$, that is, vectors in the set

$$\mathcal{Q} \doteq \{(\mathbf{M}\boldsymbol{\lambda}) \circ \mathbf{q}_0 \mid \boldsymbol{\lambda} \in \mathbb{R}^n\}. \tag{9}$$

The dual optimization problem now can be stated to be the problem of finding

$$\arg\min_{\mathbf{q} \in \overline{\mathcal{Q}}} B_F\left(\tilde{\mathbf{p}} \parallel \mathbf{q}\right)$$

where $\overline{\mathcal{Q}}$ is the closure of $\mathcal{Q}$.

The remarkable fact about these two optimization problems is that their solutions are the same, and, moreover, this solution turns out to be the unique point at the intersection of $\mathcal{P}$ and $\overline{\mathcal{Q}}$. We take the statement of this theorem from Lafferty, Della Pietra and Della Pietra [18]. The result appears to be due to Csiszár [6, 7] and Topsoe [21]. A proof for the case of (normalized) relative entropy is given by Della Pietra, Della Pietra and Lafferty [10]. See also Csiszár's survey article [8].

Theorem 1 *Let $\tilde{\mathbf{p}}$, $\mathbf{q}_0$, $\mathbf{M}$, Δ, F, B_F, $\mathcal{P}$ and $\mathcal{Q}$ be as above. Assume $B_F\left(\tilde{\mathbf{p}} \parallel \mathbf{q}_0\right) < \infty$. Then there exists a unique $\mathbf{q}_\star \in \Delta$ satisfying:*

1. $\mathbf{q}_\star \in \mathcal{P} \cap \overline{\mathcal{Q}}$

2. $B_F\left(\mathbf{p} \;\|\; \mathbf{q}\right) = B_F\left(\mathbf{p} \;\|\; \mathbf{q}_\star\right) + B_F\left(\mathbf{q}_\star \;\|\; \mathbf{q}\right)$ *for any* $\mathbf{p} \in \mathcal{P}$ *and* $\mathbf{q} \in \mathcal{Q}$

3. $\mathbf{q}_\star = \arg\min_{\mathbf{q}\in\overline{\mathcal{Q}}} B_F\left(\tilde{\mathbf{p}} \;\|\; \mathbf{q}\right)$

4. $\mathbf{q}_\star = \arg\min_{\mathbf{p}\in\mathcal{P}} B_F\left(\mathbf{p} \;\|\; \mathbf{q}_0\right)$.

Moreover, any one of these four properties determines $\mathbf{q}_\star$ *uniquely.*

This theorem will be extremely useful in proving the convergence of the algorithms described below. We will show in the next section how boosting and logistic regression can be viewed as optimization problems of the type given in part 3 of the theorem. Then, to prove optimality, we only need to show that our algorithms converge to a point in $\mathcal{P}\cap\overline{\mathcal{Q}}$.

4 BOOSTING AND LOGISTIC REGRESSION REVISITED

We return now to the boosting and logistic regression problems outlined in Section 2, and show how these can be cast in the form of the optimization problems outlined above.

Recall that for boosting, our goal is to find $\boldsymbol{\lambda}$ such that

$$\sum_{i=1}^{m} \exp\left(-y_i \sum_{j=1}^{n} \lambda_j h_j(x_i)\right) \tag{10}$$

is minimized, or, more precisely, if the minimum is not attained at a finite $\boldsymbol{\lambda}$, then we seek a procedure for finding a sequence $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots$ which causes this function to converge to its infimum. For shorthand, we call this the *ExpLoss* problem.

To view this problem in the form given in Section 3, we let $\tilde{\mathbf{p}} = \mathbf{0}$, $\mathbf{q}_0 = \mathbf{1}$ (the all 0's and all 1's vectors). We let $M_{ij} = y_i h_j(x_i)$, from which it follows that $(\mathbf{M}\boldsymbol{\lambda})_i = \sum_{j=1}^{n} \lambda_j y_i h_j(x_i)$. The space $\Delta = \mathbb{R}_+^m$. Finally, we take F to be as in Eq. (4) so that B_F is the unnormalized relative entropy.

As noted earlier, in this case, $\mathbf{v} \circ \mathbf{q}$ is as given in Eq. (7). In particular, this means that

$$\mathcal{Q} = \left\{ \mathbf{q} \in \mathbb{R}_+^m \;\middle|\; q_i = \exp\left(-\sum_{j=1}^{n} \lambda_j y_i h_j(x_i)\right), \boldsymbol{\lambda} \in \mathbb{R}^n \right\}.$$

Furthermore, it is trivial to see that

$$\mathrm{D}_U\left(\mathbf{0} \;\|\; \mathbf{q}\right) = \sum_{i=1}^{m} q_i \tag{11}$$

so that $\mathrm{D}_U\left(\mathbf{0} \;\|\; (\mathbf{M}\boldsymbol{\lambda}) \circ \mathbf{q}_0\right)$ is equal to Eq. (10). Thus, minimizing $\mathrm{D}_U\left(\mathbf{0} \;\|\; \mathbf{q}\right)$ over $\mathbf{q} \in \overline{\mathcal{Q}}$ is equivalent to minimizing Eq. (10). By Theorem 1, this is equivalent to finding $\mathbf{q} \in \overline{\mathcal{Q}}$ satisfying the constraints

$$\sum_{i=1}^{m} q_i M_{ij} = \sum_{i=1}^{m} q_i y_i h_j(x_i) = 0 \tag{12}$$

for $j = 1, \ldots, n$.

Logistic regression can be reduced to an optimization problem of this form in nearly the same way. Recall that here our goal is to find $\boldsymbol{\lambda}$ (or a sequence of $\boldsymbol{\lambda}$'s) which minimize

$$\sum_{i=1}^{m} \ln\left(1 + \exp\left(-y_i \sum_{j=1}^{n} \lambda_j h_j(x_i)\right)\right). \tag{13}$$

For shorthand, we call this the *LogLoss* problem. We define $\tilde{\mathbf{p}}$ and $\mathbf{M}$ exactly as for exponential loss. The vector $\mathbf{q}_0$ is still constant, but now is defined to be $(1/2)\mathbf{1}$, and the space Δ is now restricted to be $[0,1]^m$. These are minor differences, however. The only important difference is in the choice of the function F, namely,

$$F(\mathbf{p}) = \sum_{i=1}^{m} \left(p_i \ln p_i + (1-p_i)\ln(1-p_i)\right).$$

The resulting Bregman distance is

$$\mathrm{D}_B\left(\mathbf{p} \;\|\; \mathbf{q}\right) = \sum_{i=1}^{m} \left(p_i \ln\left(\frac{p_i}{q_i}\right) + (1-p_i)\ln\left(\frac{1-p_i}{1-q_i}\right)\right).$$

Trivially,

$$\mathrm{D}_B\left(\mathbf{0} \;\|\; \mathbf{q}\right) = -\sum_{i=1}^{m} \ln(1-q_i). \tag{14}$$

For this choice of F, it can be verified using calculus that

$$(\mathbf{v} \circ \mathbf{q})_i = \frac{q_i e^{-v_i}}{1 - q_i + q_i e^{-v_i}} \tag{15}$$

so that

$$\mathcal{Q} = \left\{ \mathbf{q} \in [0,1]^m \;\middle|\; q_i = \sigma\left(\sum_{j=1}^{n} \lambda_j y_i h_j(x_i)\right), \boldsymbol{\lambda} \in \mathbb{R}^n \right\}.$$

where $\sigma(x) = (1+e^x)^{-1}$. Thus, $\mathrm{D}_B\left(\mathbf{0} \;\|\; (\mathbf{M}\boldsymbol{\lambda}) \circ \mathbf{q}_0\right)$ is equal to Eq. (13) so minimizing $\mathrm{D}_B\left(\mathbf{0} \;\|\; \mathbf{q}\right)$ over $\mathbf{q} \in \overline{\mathcal{Q}}$ is equivalent to minimizing Eq. (13). As before, this is the same as finding $q \in \overline{\mathcal{Q}}$ satisfying the constraints in Eq. (12).

5 PARALLEL OPTIMIZATION METHODS

In this section, we describe a new algorithm for the *ExpLoss* and *LogLoss* problems using an iterative method in which all weights λ_j are updated on each iteration. The algorithm is shown in Fig. 1. The algorithm can be used with any function F satisfying certain conditions described below; in particular, we will see that it can be used with the choices of F given in Section 4. Thus, this is really a single algorithm that can be used for both loss-minimization problems by setting the parameters appropriately. Note that, without loss of generality, we assume in this section that for all instances i, $\sum_{j=1}^{n} |M_{ij}| \le 1$.

The algorithm is very simple. On each iteration, the vector $\boldsymbol{\delta}_t$ is computed as shown and added to the parameter vector $\boldsymbol{\lambda}_t$. We assume for all our algorithms that the inputs are such that infinite-valued updates never occur.

This algorithm is new for both minimization problems. Optimization methods for *ExpLoss*, notably AdaBoost, have

Parameters: $\Delta \subseteq \mathbb{R}_+^m$
$F : \Delta \to \mathbb{R}$ satisfying Assumptions 1 and 2
$\mathbf{q}_0 \in \Delta$ such that $B_F\left(\mathbf{0} \parallel \mathbf{q}_0\right) < \infty$

Input: Matrix $\mathbf{M} \in [-1,1]^{m\times n}$ where, for all i, $\sum_{j=1}^n |M_{ij}| \le 1$

Output: $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots$ such that

$$\lim_{t\to\infty} B_F\left(\mathbf{0} \parallel (\mathbf{M}\boldsymbol{\lambda}_t)\circ \mathbf{q}_0\right) = \inf_{\lambda\in\mathbb{R}^n} B_F\left(\mathbf{0} \parallel (\mathbf{M}\boldsymbol{\lambda})\circ \mathbf{q}_0\right).$$

Let $\boldsymbol{\lambda}_1 = \mathbf{0}$
For $t = 1, 2, \ldots$:

- $\mathbf{q}_t = (\mathbf{M}\boldsymbol{\lambda}_t) \circ_F \mathbf{q}_0$
- For $j = 1, \ldots, n$:

$$\begin{aligned} W_{t,j}^+ &= \sum_{i:\mathrm{sign}(M_{ij})=+1} q_{t,i}|M_{ij}| \\ W_{t,j}^- &= \sum_{i:\mathrm{sign}(M_{ij})=-1} q_{t,i}|M_{ij}| \\ \delta_{t,j} &= \frac{1}{2}\ln\left(\frac{W_{t,j}^+}{W_{t,j}^-}\right) \end{aligned}$$

- Update parameters: $\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t + \boldsymbol{\delta}_t$

Figure 1: The parallel-update optimization algorithm.

generally involved updates of one feature at a time. Parallel-update methods for *LogLoss* are well known (see, for example, [9, 10]). However, our updates take a different form from the usual updates derived for logistic models.

A useful point is that the distribution $\mathbf{q}_{t+1}$ is a simple function of the previous distribution $\mathbf{q}_t$. By Eq. (8),

$$\begin{aligned} \mathbf{q}_{t+1} = (\mathbf{M}(\boldsymbol{\lambda}_t + \boldsymbol{\delta}_t)) \circ \mathbf{q}_0 &= (\mathbf{M}\boldsymbol{\delta}_t) \circ ((\mathbf{M}\boldsymbol{\lambda}_t)\circ \mathbf{q}_0) \\ &= (\mathbf{M}\boldsymbol{\delta}_t) \circ \mathbf{q}_t. \end{aligned} \tag{16}$$

This gives

$$q_{t+1,i} = \begin{cases} q_{t,i}\exp\left(-\sum_{j=1}^n \delta_{t,j}M_{ij}\right) \\ q_{t,i}\left[(1-q_{t,i})\exp\left(\sum_{j=1}^n \delta_{t,j}M_{ij}\right) + q_{t,i}\right]^{-1} \end{cases} \tag{17}$$

for *ExpLoss* and *LogLoss* respectively.

We will prove next that the algorithm given in Fig. 1 converges to optimality for either loss. We prove this abstractly for any matrix $\mathbf{M}$ and vector $\mathbf{q}_0$, and for any function F satisfying the following assumptions:

Assumption 1 *For any* $\mathbf{v} \in \mathbb{R}^m$, $\mathbf{q} \in \Delta$,

$$B_F\left(\mathbf{0} \parallel \mathbf{v}\circ\mathbf{q}\right) - B_F\left(\mathbf{0} \parallel \mathbf{q}\right) \le \sum_{i=1}^m q_i(e^{-v_i} - 1).$$

Assumption 2 *For any* $c < \infty$*, the set*

$$\left\{\mathbf{q} \in \Delta \mid B_F\left(\mathbf{0} \parallel \mathbf{q}\right) \le c\right\}$$

is bounded.

We will show later that the choices of F given in Section 4 satisfy these assumptions which will allow us to prove convergence for *ExpLoss* and *LogLoss*.

To prove convergence, we use the auxiliary-function technique of Della Pietra, Della Pietra and Lafferty [10]. Very roughly, the idea of the proof is to derive a nonnegative lower bound called an auxiliary function on how much the loss decreases on each iteration. Since the loss never increases and is lower bounded by zero, the auxiliary function must converge to zero. The final step is to show that when the auxiliary function is zero, the constraints defining the set $\mathcal{P}$ must be satisfied, and therefore, by Theorem 1, we must have converged to optimality.

More formally, we define an *auxiliary function* for a sequence $\mathbf{q}_1, \mathbf{q}_2, \ldots$ and matrix $\mathbf{M}$ to be a continuous function $A : \Delta \to \mathbb{R}$ satisfying the two conditions:

$$B_F\left(\mathbf{0} \parallel \mathbf{q}_{t+1}\right) - B_F\left(\mathbf{0} \parallel \mathbf{q}_t\right) \le A(\mathbf{q}_t) \le 0 \tag{18}$$

and

$$A(\mathbf{q}) = 0 \Rightarrow \mathbf{q}^{\mathrm{T}}\mathbf{M} = \mathbf{0}. \tag{19}$$

Before proving convergence of specific algorithms, we prove the following lemma which shows, roughly, that if a sequence has an auxiliary function, then the sequence converges to the optimum point $\mathbf{q}_\star$. Thus, proving convergence of a specific algorithm reduces to simply finding an auxiliary function.

Lemma 2 *Let A be an auxiliary function for $\mathbf{q}_1, \mathbf{q}_2, \ldots$ and matrix $\mathbf{M}$. Assume the $\mathbf{q}_t$'s lie in a compact subspace of $\mathcal{Q}$ where $\mathcal{Q}$ is as in Eq. (9); in particular, this will be the case if Assumption 2 holds and $B_F\left(\mathbf{0} \parallel \mathbf{q}_1\right) < \infty$. Then*

$$\lim_{t\to\infty} \mathbf{q}_t = \mathbf{q}_\star \doteq \arg\min_{\mathbf{q}\in\overline{\mathcal{Q}}} B_F\left(\mathbf{0} \parallel \mathbf{q}\right).$$

Proof: By condition (18), $B_F\left(\mathbf{0} \parallel \mathbf{q}_t\right)$ is a nonincreasing sequence which is bounded below by zero. Therefore, the sequence of differences $B_F\left(\mathbf{0} \parallel \mathbf{q}_{t+1}\right) - B_F\left(\mathbf{0} \parallel \mathbf{q}_t\right)$ must converge to zero. By condition (18), this means that $A(\mathbf{q}_t)$ must also converge to zero. Because we assume that the $\mathbf{q}_t$'s lie in a compact space, the sequence of $\mathbf{q}_t$'s must have a subsequence converging to some point $\hat{\mathbf{q}} \in \Delta$. By continuity of A, we have $A(\hat{\mathbf{q}}) = 0$. Therefore, $\hat{\mathbf{q}} \in \mathcal{P}$ by condition (19), where $\mathcal{P}$ is as in Eq. (5). On the other hand, $\hat{\mathbf{q}}$ is the limit of a sequence of points in $\mathcal{Q}$ so $\hat{\mathbf{q}} \in \overline{\mathcal{Q}}$. Thus, $\hat{\mathbf{q}} \in \mathcal{P}\cap\overline{\mathcal{Q}}$ so $\hat{\mathbf{q}} = \mathbf{q}_\star$ by Theorem 1.

This argument and the uniqueness of $\mathbf{q}_\star$ show that the $\mathbf{q}_t$'s have only a single limit point $\mathbf{q}_\star$. Suppose that the entire sequence did not converge to $\mathbf{q}_\star$. Then we could find an open set B containing $\mathbf{q}_\star$ such that $\{\mathbf{q}_1, \mathbf{q}_2, \ldots\} - B$ contains infinitely many points and therefore has a limit point which must be in the closed set $\Delta - B$ and so must be different from $\mathbf{q}_\star$. This, we have already argued, is impossible. Therefore, the entire sequence converges to $\mathbf{q}_\star$. ■

We can now apply this lemma to prove the convergence of the algorithm of Fig. 1.

Theorem 3 *Let F satisfy Assumptions 1 and 2, and assume that $B_F\left(\mathbf{0} \parallel \mathbf{q}_0\right) < \infty$. Let the sequences $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots$ and $\mathbf{q}_1, \mathbf{q}_2, \ldots$ be generated by the algorithm of Fig. 1. Then*

$$\lim_{t\to\infty} \mathbf{q}_t = \arg\min_{\mathbf{q}\in\overline{\mathcal{Q}}} B_F\left(\mathbf{0} \parallel \mathbf{q}\right)$$

where Q is as in Eq. (9). That is,

$$\lim_{t\to\infty} B_F\left(\mathbf{0} \parallel (\mathbf{M}\boldsymbol{\lambda}_t)\circ \mathbf{q}_0\right) = \inf_{\lambda\in\mathbb{R}^n} B_F\left(\mathbf{0} \parallel (\mathbf{M}\boldsymbol{\lambda})\circ \mathbf{q}_0\right).$$

Proof: Let

$$\begin{aligned} W_j^+(\mathbf{q}) &= \sum_{i:\text{sign}(M_{ij})=+1} q_i|M_{ij}| \\ W_j^-(\mathbf{q}) &= \sum_{i:\text{sign}(M_{ij})=-1} q_i|M_{ij}| \end{aligned}$$

so that $W_{t,j}^+ = W_j^+(\mathbf{q}_t)$ and $W_{t,j}^- = W_j^-(\mathbf{q}_t)$. We claim that the function

$$A(\mathbf{q}) = -\sum_{j=1}^n \left(\sqrt{W_j^+(\mathbf{q})} - \sqrt{W_j^-(\mathbf{q})}\right)^2$$

is an auxiliary function for $\mathbf{q}_1, \mathbf{q}_2, \ldots$. Clearly, A is continuous and nonpositive.

Let $s_{ij} \doteq \text{sign}(M_{ij})$. We can upper bound the change in $B_F\left(\mathbf{0} \parallel \mathbf{q}_t\right)$ on round t by $A(\mathbf{q}_t)$ as follows:

$$\begin{aligned} & B_F\left(\mathbf{0} \parallel \mathbf{q}_{t+1}\right) - B_F\left(\mathbf{0} \parallel \mathbf{q}_t\right) \\ &= B_F\left(\mathbf{0} \parallel (\mathbf{M}\boldsymbol{\delta}_t)\circ\mathbf{q}_t\right) - B_F\left(\mathbf{0} \parallel \mathbf{q}_t\right) && (20) \\ &\le \sum_{i=1}^m q_{t,i}\left[\exp\left(-\sum_{j=1}^n \delta_{t,j}M_{ij}\right) - 1\right] && (21) \\ &= \sum_{i=1}^m q_{t,i}\left[\exp\left(-\sum_{j=1}^n \delta_{t,j}s_{ij}|M_{ij}|\right) - 1\right] \\ &\le \sum_{i=1}^m q_{t,i}\left[\sum_{j=1}^n |M_{ij}|(e^{-\delta_{t,j}s_{ij}} - 1)\right] && (22) \\ &= \sum_{j=1}^n \left(W_{t,j}^+ e^{-\delta_{t,j}} + W_{t,j}^- e^{\delta_{t,j}} - W_{t,j}^+ - W_{t,j}^-\right) && (23) \\ &= -\sum_{j=1}^n \left(\sqrt{W_{t,j}^+} - \sqrt{W_{t,j}^-}\right)^2 = A(\mathbf{q}_t). && (24) \end{aligned}$$

Eqs. (20) and (21) follow from Eq. (16) and Assumption 1, respectively. Eq. (22) uses the fact that, for any x_j's and for $p_j \ge 0$ with $\sum_j p_j \le 1$, we have

$$\begin{aligned} & \exp\left(\sum_j p_j x_j\right) - 1 \\ &= \exp\left(\sum_j p_j x_j + 0\cdot\left(1 - \sum_j p_j\right)\right) - 1 \\ &\le \sum_j p_j e^{x_i} + \left(1 - \sum_j p_j\right) - 1 = \sum_j p_j(e^{x_i} - 1) \end{aligned} \tag{25}$$

by Jensen's inequality applied to the convex function e^x. Eq. (23) uses the definitions of $W_{t,j}^+$ and $W_{t,j}^-$, and Eq. (24) uses our choice of $\boldsymbol{\delta}_t$ (indeed, $\boldsymbol{\delta}_t$ was chosen specifically to minimize Eq. (23)).

If $A(\mathbf{q}) = 0$ then for all j, $W_j^+(\mathbf{q}) = W_j^-(\mathbf{q})$, that is,

$$0 = W_j^+(\mathbf{q}) - W_j^-(\mathbf{q}) = \sum_{i=1}^m q_i s_{ij}|M_{ij}| = \sum_{i=1}^m q_i M_{ij}.$$

Thus, A is an auxiliary function for $\mathbf{q}_1, \mathbf{q}_2, \ldots$. The theorem now follows immediately from Lemma 2. ■

To apply this theorem to the *ExpLoss* and *LogLoss* problems, we only need to verify that Assumptions 1 and 2 are satisfied. For *ExpLoss*, Assumption 1 holds with equality. For *LogLoss*,

$$\begin{aligned} & \mathrm{D}_B\left(\mathbf{0} \parallel \mathbf{v}\circ\mathbf{q}\right) - \mathrm{D}_B\left(\mathbf{0} \parallel \mathbf{q}\right) \\ &= \sum_{i=1}^m \ln\left(\frac{1-q_i}{1-(\mathbf{v}\circ\mathbf{q})_i}\right) \\ &= \sum_{i=1}^m \ln\left(1 - q_i + q_i e^{-v_i}\right) \\ &\le \sum_{i=1}^m \left(-q_i + q_i e^{-v_i}\right). \end{aligned}$$

The first and second equalities use Eqs. (14) and (15), respectively. The final inequality uses $1 + x \le e^x$ for all x.

Assumption 2 holds trivially for *LogLoss* since $\Delta = [0,1]^m$ is bounded. For *ExpLoss*, if $\mathrm{D}_U\left(\mathbf{0} \parallel \mathbf{q}\right) \le c$ then

$$\sum_{i=1}^m q_i \le c$$

which clearly defines a bounded subset of $\mathbb{R}_+^m$.

6 SEQUENTIAL ALGORITHMS

In this section, we describe another algorithm for the same minimization problems described in Section 4. However, unlike the algorithm of Section 5, the one that we present now only updates the weight of one feature at a time. While the parallel-update algorithm may give faster convergence when there are not too many features, the sequential-update algorithm can be used when there are a very large number of features using an oracle for selecting which feature to update next. For instance, AdaBoost, which is essentially equivalent to the sequential-update algorithm for *ExpLoss*, uses an assumed weak learning algorithm to select a weak hypothesis, i.e., one of the features. The sequential algorithm that we present for *LogLoss* can be used in exactly the same way. The algorithm is shown in Fig. 2.

Theorem 4 *Given the assumptions of Theorem 3, the algorithm of Fig. 2 converges to optimality in the sense of Theorem 3.*

Proof: For this theorem, we use the auxiliary function

$$A(\mathbf{q}) = \sqrt{\left(\sum_{i=1}^m q_i\right)^2 - \max_j\left(\sum_{i=1}^m q_i M_{ij}\right)^2} - \sum_{i=1}^m q_i.$$

Parameters: (same as in Fig. 1)
Input: Matrix $\mathbf{M} \in [-1,1]^{m\times n}$
Output: (same as in Fig. 1)
Let $\boldsymbol{\lambda}_1 = \mathbf{0}$
For $t = 1, 2, \ldots$:

- $\mathbf{q}_t = (\mathbf{M}\boldsymbol{\lambda}_t) \circ \mathbf{q}_0$
- $j_t = \arg\max_j \left| \sum_{i=1}^m q_{t,i} M_{ij} \right|$
- $r_t = \sum_{i=1}^m q_{t,i} M_{ij_t}$
- $Z_t = \sum_{i=1}^m q_{t,i}$
- $\alpha_t = \frac{1}{2} \ln\left(\frac{Z_t + r_t}{Z_t - r_t}\right)$
- $\delta_{t,j} = \begin{cases} \alpha_t & \text{if } j = j_t \\ 0 & \text{else} \end{cases}$
- Update parameters: $\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t + \boldsymbol{\delta}_t$

Figure 2: The sequential-update optimization algorithm.

This function is clearly continuous and nonpositive. We have that

$$\begin{aligned}
& B_F\left(\mathbf{0} \;\|\; \mathbf{q}_{t+1}\right) - B_F\left(\mathbf{0} \;\|\; \mathbf{q}_t\right) \\
\leq\ & \sum_{i=1}^m q_{t,i}\left(\exp\left(-\sum_{j=1}^n \delta_{t,j} M_{ij}\right) - 1\right) \\
=\ & \sum_{i=1}^m q_{t,i}\left(\exp\left(-\alpha_t M_{ij_t}\right) - 1\right) && (26) \\
\leq\ & \sum_{i=1}^m q_{t,i}\left(\frac{1+M_{ij_t}}{2} e^{-\alpha_t} + \frac{1-M_{ij_t}}{2} e^{\alpha_t} - 1\right) && (27) \\
=\ & \frac{Z_t + r_t}{2} e^{-\alpha_t} + \frac{Z_t - r_t}{2} e^{\alpha_t} - Z_t && (28) \\
=\ & \sqrt{Z_t^2 - r_t^2} - Z_t = A(\mathbf{q}_t) && (29)
\end{aligned}$$

where Eq. (27) uses the convexity of $e^{-\alpha_t x}$, and Eq. (29) uses our choice of α_t (as before, we chose α_t to minimize the bound in Eq. (28)).

If $A(\mathbf{q}) = 0$ then

$$0 = \max_j \left| \sum_{i=1}^m q_i M_{ij} \right|$$

so $\sum_i q_i M_{ij} = 0$ for all j. Thus, A is an auxiliary function for $\mathbf{q}_1, \mathbf{q}_2, \ldots$ and the theorem follows immediately from Lemma 2. ■

As mentioned above, this algorithm is essentially equivalent to AdaBoost, specifically, the version of AdaBoost first presented by Freund and Schapire [13]. In AdaBoost, on each iteration, a distribution D_t over the training examples is computed and the weak learner seeks a weak hypothesis with low error with respect to this distribution. The algorithm presented in this section assumes that the space of weak hypotheses consists of the features $h_1, \ldots, h_n$, and that the weak learner always succeeds in selecting the feature with lowest error (or, more accurately, with error farthest from $1/2$). Translating to our notation, the weight $D_t(i)$ assigned to example (x_i, y_i) by AdaBoost is exactly equal to $q_{t,i}/Z_t$, and the weighted error of the t-th weak hypothesis is equal to

$$\frac{1}{2}\left(1 - \frac{r_t}{Z_t}\right).$$

Theorem 4 then is the first proof that AdaBoost always converges to the minimum of the exponential loss (assuming an idealized weak learner of the form above). Note that when $\mathbf{q}_\star \neq \mathbf{0}$, this theorem also tells us the exact form of $\lim D_t$. However, we do not know what the limiting behavior of D_t is when $\mathbf{q}_\star = \mathbf{0}$, nor do we know about the limiting behavior of the parameters $\boldsymbol{\lambda}_t$ (whether or not $\mathbf{q}_\star = \mathbf{0}$).

We have also presented in this section a new algorithm for logistic regression. In fact, this algorithm is the same as one given by Duffy and Helmbold [12] except for the choice of α_t. In practical terms, very little work would be required to alter an existing learning system based on AdaBoost so that it uses logistic loss rather than exponential loss—the only difference is in the manner in which $\mathbf{q}_t$ is computed from $\boldsymbol{\lambda}_t$. We can even do this for systems based on "confidence-rated" boosting [20] in which α_t and j_t are chosen together on each round to minimize Eq. (26) rather than an approximation of this expression as used in the algorithm of Fig. 2. (Note that the proof of Theorem 4 can easily be modified to prove the convergence of such an algorithm using the same auxiliary function.)

7 A PARAMETERIZED FAMILY OF ITERATIVE ALGORITHMS

In previous sections, we described separate parallel- and sequential-update algorithms. In this section, we describe a parameterized family of algorithms that includes the parallel-update algorithm of Section 5 as well as a sequential-update algorithm that is different from the one in Section 6. This family of algorithms also includes other algorithms that may be more appropriate than either in certain situations as we explain below.

The algorithm, which is shown in Fig. 3, is similar to the parallel-update algorithm of Fig. 1. On each round, the quantities $W_{t,j}^+$ and $W_{t,j}^-$ are computed as before, and the vector $\mathbf{d}_t$ is computed as $\boldsymbol{\delta}_t$ was computed in Fig. 1. Now, however, this vector $\mathbf{d}_t$ is not added directly to $\boldsymbol{\lambda}_t$. Instead, another vector $\mathbf{a}_t$ is selected which provides a "scaling" of the features. This vector is chosen to maximize a measure of progress while restricted to belong to the set $\mathcal{A}_{\mathrm{M}}$. The allowed form of these scaling vectors is given by the set $\mathcal{A}$, a parameter of the algorithm; $\mathcal{A}_{\mathrm{M}}$ is the restriction of $\mathcal{A}$ to those vectors $\mathbf{a}$ satisfying the constraint that for all i,

$$\sum_{j=1}^n a_j |M_{ij}| \leq 1.$$

The parallel-update algorithm of Fig. 1 is obtained by choosing $\mathcal{A} = \{\mathbf{1}\}$ and assuming that $\sum_j |M_{ij}| \leq 1$ for all i. (Equivalently, we can make no such assumption, and choose $\mathcal{A} = \{c\mathbf{1} \mid c > 0\}$.)

Parameters: (same as in Fig. 1)
$\mathcal{A} \subseteq \mathbb{R}_+^n$

Input: Matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$ satisfying the condition that $\forall j, \exists \mathbf{a} \in \mathcal{A}_{\mathrm{M}}$ for which $a_j > 0$ where
$\mathcal{A}_{\mathrm{M}} \doteq \{\mathbf{a} \in \mathcal{A} \mid \forall i : \sum_j a_j |M_{ij}| \leq 1\}$

Output: (same as in Fig. 1)
Let $\boldsymbol{\lambda}_1 = \mathbf{0}$
For $t = 1, 2, \ldots$:

- $\mathbf{q}_t = (\mathbf{M}\boldsymbol{\lambda}_t) \circ_F \mathbf{q}_0$
- For $j = 1, \ldots, n$:

$$\begin{aligned} W_{t,j}^+ &= \sum_{i:\mathrm{sign}(M_{ij})=+1} q_{t,i} |M_{ij}| \\ W_{t,j}^- &= \sum_{i:\mathrm{sign}(M_{ij})=-1} q_{t,i} |M_{ij}| \\ d_{t,j} &= \frac{1}{2} \ln \left(\frac{W_{t,j}^+}{W_{t,j}^-} \right) \end{aligned}$$

- $\mathbf{a}_t = \arg\max_{\mathbf{a} \in \mathcal{A}_{\mathrm{M}}} \sum_{j=1}^n a_j \left(\sqrt{W_{t,j}^+} - \sqrt{W_{t,j}^-} \right)^2$
- $\forall j : \delta_{t,j} = a_{t,j} d_{t,j}$
- Update parameters: $\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t + \boldsymbol{\delta}_t$

Figure 3: A parameterized family of iterative optimization algorithms.

We can obtain a sequential-update algorithm by choosing $\mathcal{A}$ to be the set of unit vectors (i.e., with one component equal to 1 and all others equal to 0), and assuming that $M_{ij} \in [-1, +1]$ for all i, j. The update then becomes

$$\delta_{t,j} = \begin{cases} d_{t,j} & \text{if } j = j_t \\ 0 & \text{else} \end{cases}$$

where

$$j_t = \arg\max_j \left| \sqrt{W_{t,j}^+} - \sqrt{W_{t,j}^-} \right|.$$

Another interesting case is when we assume that $\sum_j M_{ij}^2 \leq 1$ for all i. It is then natural to choose

$$\mathcal{A} = \{\mathbf{a} \in \mathbb{R}_+^n \mid ||\mathbf{a}||_2 = 1\}$$

which ensures that $\mathcal{A}_{\mathrm{M}} = \mathcal{A}$. Then the maximization over $\mathcal{A}_{\mathrm{M}}$ can be solved analytically giving the update

$$\delta_{t,j} = \frac{b_j d_{t,j}}{||\mathbf{b}||_2}$$

where $b_j = \left(\sqrt{W_{t,j}^+} - \sqrt{W_{t,j}^-} \right)^2$. (This idea generalizes easily to the case in which $\sum_j M_{ij}^p \leq 1$ and $||\mathbf{a}||_q = 1$ for any dual norms p and q.)

A final case is when we do not restrict the scaling vectors at all, i.e., we choose $\mathcal{A} = \mathbb{R}_+^n$. In this case, the maximization problem that must be solved to choose each $\mathbf{a}_t$ is a linear programming problem with n variables and m constraints.

We now prove the convergence of this entire family of algorithms.

Theorem 5 *Given the assumptions of Theorem 3, the algorithm of Fig. 3 converges to optimality in the sense of Theorem 3.*

Proof: We use the auxiliary function

$$A(\mathbf{q}) = -\max_{\mathbf{a} \in \mathcal{A}_{\mathrm{M}}} \sum_{j=1}^n a_j \left(\sqrt{W_j^+(\mathbf{q})} - \sqrt{W_j^-(\mathbf{q})} \right)^2$$

where W_j^+ and W_j^- are as in Theorem 3. This function is continuous and nonpositive. We can bound the change in $B_F(\mathbf{0} \parallel \mathbf{q}_t)$ using the same technique given in Theorem 3:

$$\begin{aligned} & B_F(\mathbf{0} \parallel \mathbf{q}_{t+1}) - B_F(\mathbf{0} \parallel \mathbf{q}_t) \\ \leq & \sum_{i=1}^m q_{t,i} \left[\exp\left(-\sum_{j=1}^n \delta_{t,j} M_{ij} \right) - 1 \right] \\ = & \sum_{i=1}^m q_{t,i} \left[\exp\left(-\sum_{j=1}^n a_{t,j} d_{t,j} s_{ij} |M_{ij}| \right) - 1 \right] \\ \leq & \sum_{i=1}^m q_{t,i} \left[\sum_{j=1}^n a_{t,j} |M_{ij}| (e^{-d_{t,j} s_{ij}} - 1) \right] \\ = & \sum_{j=1}^n a_{t,j} (W_{t,j}^+ e^{-d_{t,j}} + W_{t,j}^- e^{d_{t,j}} - W_{t,j}^+ - W_{t,j}^-) \\ = & -\sum_{j=1}^n a_{t,j} \left(\sqrt{W_{t,j}^+} - \sqrt{W_{t,j}^-} \right)^2 = A(\mathbf{q}_t). \end{aligned}$$

Finally, if $A(\mathbf{q}) = 0$ then

$$\max_{\mathbf{a} \in \mathcal{A}_{\mathrm{M}}} \sum_{j=1}^n a_j \left(\sqrt{W_j^+(\mathbf{q})} - \sqrt{W_j^-(\mathbf{q})} \right)^2 = 0.$$

Since for every j there exists $\mathbf{a} \in \mathcal{A}_{\mathrm{M}}$ with $a_j > 0$, this implies $W_j^+(\mathbf{q}) = W_j^-(\mathbf{q})$ for all j, i.e., $\sum_i q_i M_{ij} = 0$. Applying Lemma 2 completes the theorem. ■

8 MULTICLASS PROBLEMS

In this section, we show how all of our results can be extended to the multiclass case. Because of the generality of the preceding results, we will see that no new algorithms need be devised and no new convergence proofs need be proved for this case. Rather, all of the preceding algorithms and proofs can be directly applied to the multiclass case.

In the multiclass case, the label set $\mathcal{Y}$ has cardinality k. Each feature is of the form $h_j : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. In logistic regression, we use a model

$$\hat{\Pr}[y|x] = \frac{e^{f_\lambda(x,y)}}{\sum_{\ell \in \mathcal{Y}} e^{f_\lambda(x,\ell)}} = \frac{1}{1 + \sum_{\ell \neq y} e^{f_\lambda(x,\ell) - f_\lambda(x,y)}} \tag{30}$$

where $f_\lambda(x, y) = \sum_{j=1}^n \lambda_j h_j(x, y)$. The loss on a training set then is

$$\sum_{i=1}^m \ln \left[1 + \sum_{\ell \neq y_i} e^{f_\lambda(x_i,\ell) - f_\lambda(x_i,y_i)} \right]. \tag{31}$$

We transform this into our framework as follows: Let

$$\mathcal{B} = \{(i,\ell) \mid 1 \le i \le m, \ell \in \mathcal{Y} - \{y_i\}\}.$$

The vectors $\mathbf{p}$, $\mathbf{q}$, etc. that we work with are in $\mathbb{R}_+^{\mathcal{B}}$. That is, they are $(k-1)m$-dimensional and are indexed by pairs in $\mathcal{B}$. Let $\bar{p}_i$ denote $\sum_{\ell \neq y_i} p_{i,\ell}$. The convex function F that we use for this case is

$$F(\mathbf{p}) = \sum_{i=1}^{m} \left[\sum_{\ell \neq y_i} p_{i,\ell} \ln p_{i,\ell} + (1-\bar{p}_i)\ln(1-\bar{p}_i) \right]$$

which is defined over the space

$$\Delta = \left\{ \mathbf{p} \in \mathbb{R}_+^{\mathcal{B}} \mid \forall i : \bar{p}_i \le 1 \right\}.$$

The resulting Bregman distance is

$$\begin{aligned} &B_F\left(\mathbf{p} \parallel \mathbf{q}\right) \\ &= \sum_{i=1}^{m} \left[\sum_{\ell \neq y} p_{i,\ell} \ln\left(\frac{p_{i,\ell}}{q_{i,\ell}}\right) + (1-\bar{p}_i)\ln\left(\frac{1-\bar{p}_i}{1-\bar{q}_i}\right) \right]. \end{aligned}$$

Clearly,

$$B_F\left(\mathbf{0} \parallel \mathbf{q}\right) = -\sum_{i=1}^{m} \ln(1-\bar{q}_i).$$

It can be shown that

$$(\mathbf{v} \circ \mathbf{q})_{(i,\ell)} = \frac{q_{i,\ell} e^{-v_{i,\ell}}}{1 - \bar{q}_i + \sum_{\ell \neq y_i} q_{i,\ell} e^{-v_{i,\ell}}}.$$

Assumption 1 can be verified by noting that

$$\begin{aligned} &B_F\left(\mathbf{0} \parallel \mathbf{v} \circ \mathbf{q}\right) - B_F\left(\mathbf{0} \parallel \mathbf{q}\right) \\ &= \sum_{i=1}^{m} \ln\left(\frac{1-\bar{q}_i}{1-\overline{(\mathbf{v} \circ \mathbf{q})}_i}\right) \\ &= \sum_{i=1}^{m} \ln\left(1 - \bar{q}_i + \sum_{\ell \neq y_i} q_{i,\ell} e^{-v_{i,\ell}}\right) \qquad (32) \\ &\le \sum_{i=1}^{m} \left(-\bar{q}_i + \sum_{\ell \neq y_i} q_{i,\ell} e^{-v_{i,\ell}}\right) \\ &= \sum_{(i,\ell)\in\mathcal{B}} q_{i,\ell}(e^{-v_{i,\ell}} - 1). \end{aligned}$$

Now let $M_{(i,\ell),j} = h_j(x_i, y_i) - h_j(x_i, \ell)$, and let $\mathbf{q}_0 = (1/k)\mathbf{1}$. Plugging in these definitions gives that $B_F\left(\mathbf{0} \parallel (\mathbf{M}\boldsymbol{\lambda}) \circ \mathbf{q}_0\right)$ is equal to Eq. (31). Thus, the algorithms of Sections 5, 6 and 7 can all be used to solve this minimization problem, and the corresponding convergence proofs are also directly applicable.

There are several multiclass versions of AdaBoost. AdaBoost.M2 [13] (a special case of AdaBoost.MR [20]), is based on the loss function

$$\sum_{(i,\ell)\in\mathcal{B}} \exp\left(f_\lambda(x_i,\ell) - f_\lambda(x_i,y_i)\right). \qquad (33)$$

For this loss, we can use a similar set up except for the choice of F. We instead use

$$F(\mathbf{p}) = \sum_{(i,\ell)\in\mathcal{B}} p_{i,\ell} \ln p_{i,\ell}$$

for $\mathbf{p} \in \Delta = \mathbb{R}_+^{\mathcal{B}}$. In fact, this is actually the same F used for (binary) AdaBoost. We have merely changed the index set to $\mathcal{B}$. Thus, as before,

$$B_F\left(\mathbf{0} \parallel \mathbf{q}\right) = \sum_{(i,\ell)\in\mathcal{B}} q_{i,\ell}$$

and

$$(\mathbf{v} \circ \mathbf{q})_{i,\ell} = q_{i,\ell} e^{-v_{i,\ell}}.$$

Choosing $\mathbf{M}$ as we did for multiclass logistic regression and $\mathbf{q}_0 = \mathbf{1}$, we have that $B_F\left(\mathbf{0} \parallel (\mathbf{M}\boldsymbol{\lambda}) \circ \mathbf{q}_0\right)$ is equal to the loss in Eq. (33). We can thus use the preceding algorithms to solve this multiclass problem as well. In particular, the sequential-update algorithm gives AdaBoost.M2.

AdaBoost.MH [20] is another multiclass version of AdaBoost. For AdaBoost.MH, we replace $\mathcal{B}$ by the index set

$$\{1, \ldots, m\} \times \mathcal{Y},$$

and for each example i and label $\ell \in \mathcal{Y}$, we define

$$\tilde{y}_{i,\ell} = \begin{cases} +1 & \text{if } \ell = y_i \\ -1 & \text{if } \ell \neq y_i. \end{cases}$$

The loss function for AdaBoost.MH is

$$\sum_{i=1}^{m} \sum_{\ell \in \mathcal{Y}} \exp\left(-\tilde{y}_{i,\ell} f_\lambda(x_i,\ell)\right). \qquad (34)$$

We now let $M_{(i,\ell),j} = \tilde{y}_{i,\ell} h_j(x_i,\ell)$ and use again the same F as in binary AdaBoost with $\mathbf{q}_0 = \mathbf{1}$ to obtain this multiclass version of AdaBoost.

9 A COMPARISON TO ITERATIVE SCALING

In this section, we describe the generalized iterative scaling (GIS) procedure of Darroch and Ratcliff [9] for comparison to our algorithms. We largely follow the description of GIS given by Berger, Della Pietra and Della Pietra [1] for the multiclass case. To make the comparison as stark as possible, we present GIS in our notation and prove its convergence using the methods developed in previous sections. In doing so, we are also able to relax one of the key assumptions traditionally used in studying GIS.

We adopt the notation and set-up used for multiclass logistic regression in Section 8. (To our knowledge, there is no analog of GIS for the exponential loss so we only consider the case of logistic loss.) We also extend this notation by defining $q_{i,y_i} = 1 - \bar{q}_i$ so that $q_{i,\ell}$ is now defined for all $\ell \in \mathcal{Y}$. Moreover, it can be verified that $q_{i,\ell} = \hat{\Pr}[\ell | x_i]$ as defined in Eq. (30) if $\mathbf{q} = (\mathbf{M}\boldsymbol{\lambda}) \circ \mathbf{q}_0$.

In GIS, the following assumptions regarding the features are usually made:

$$\forall i,j,\ell:\ h_j(x_i,\ell) \ge 0 \quad \text{and} \quad \forall i,\ell:\ \sum_{j=1}^{n} h_j(x_i,\ell) = 1\,.$$

In this section, we prove that GIS converges with the second condition replaced by a milder one, namely, that

$$\forall i, \ell : \sum_{j=1}^{n} h_j(x_i, \ell) \leq 1 \ .$$

Since, in the multiclass case, a constant can be added to all features h_j without changing the model or loss function, and since the features can be scaled by any constant, the two assumptions we consider clearly can be made to hold without loss of generality. The improved iterative scaling algorithm of Della Pietra, Della Pietra and Lafferty [10] also requires only these milder assumptions but is much more complicated to implement, requiring a numerical search (such as Newton-Raphson) for each feature on each iteration.

GIS works much like the parallel-update algorithm of Section 5 with F, $\mathbf{M}$ and $\mathbf{q}_0$ as defined for multiclass logistic regression in Section 8. The only difference is in the computation of the vector of updates $\boldsymbol{\delta}_t$, for which GIS requires direct access to the features h_j. Specifically, in GIS, $\boldsymbol{\delta}_t$ is defined to be

$$\delta_{t,j} = \ln\left(\frac{H_j}{P_j(\mathbf{q}_t)}\right)$$

where

$$\begin{aligned} H_j &= \sum_{i=1}^{m} h_j(x_i, y_i) \\ P_j(\mathbf{q}) &= \sum_{i=1}^{m} \sum_{\ell \in \mathcal{Y}} q_{i,\ell} h_j(x_i, \ell). \end{aligned}$$

Clearly, these updates are quite different from the updates described in this paper.

Using more notation from Sections 5 and 8, we can reformulate $P_j(\mathbf{q})$ within our framework as follows:

$$\begin{aligned} P_j(\mathbf{q}) &= \sum_{i=1}^{m} \sum_{\ell \in \mathcal{Y}} q_{i,\ell}\, h_j(x_i, \ell) \\ &= \sum_{i=1}^{m} h_j(x_i, y_i) \\ &\quad + \sum_{i=1}^{m} \sum_{\ell \in \mathcal{Y}} q_{i,\ell} \left[h_j(x_i, \ell) - h_j(x_i, y_i)\right] \\ &= H_j - \sum_{(i,\ell) \in \mathcal{B}} q_{i,\ell} M_{(i,\ell),j} \\ &= H_j - (W_j^+(\mathbf{q}) - W_j^-(\mathbf{q})). \end{aligned} \tag{35}$$

We can now prove the convergence of these updates using the usual auxiliary function method.

Theorem 6 *Let F, $\mathbf{M}$ and $\mathbf{q}_0$ be as above. Then the modified GIS algorithm described above converges to optimality in the sense of Theorem 3.*

Proof: We will show that

$$\begin{aligned} A(\mathbf{q}) &\doteq -\mathrm{D}_U\left(\langle H_1, \ldots, H_n \rangle \ \|\ \langle P_1(\mathbf{q}), \ldots, P_n(\mathbf{q})\rangle\right) \\ &= -\sum_{j=1}^{n} \left(H_j \ln \frac{H_j}{P_j(\mathbf{q})} + P_j(\mathbf{q}) - H_j\right) \end{aligned} \tag{36}$$

is an auxilliary function for the vectors $\mathbf{q}_1, \mathbf{q}_2, \ldots$ computed by GIS. Clearly, A is continuous, and the usual nonnegativity properties of unnormalized relative entropy imply that $A(\mathbf{q}) \leq 0$ with equality if and only if $H_j = P_j(\mathbf{q})$ for all j. From Eq. (35), $H_j = P_j(\mathbf{q})$ if and only if $W_j^+(\mathbf{q}) = W_j^-(\mathbf{q})$. Thus, $A(\mathbf{q}) = 0$ implies that the constraints $\mathbf{q}^{\mathrm{T}}\mathbf{M} = \mathbf{0}$ as in the proof of Theorem 3. All that remains to be shown is that

$$B_F\left(\mathbf{0} \ \|\ (\mathbf{M}\boldsymbol{\delta}) \circ \mathbf{q}\right) - B_F\left(\mathbf{0} \ \|\ \mathbf{q}\right) \leq A(\mathbf{q}) \tag{37}$$

where

$$\delta_j = \ln\left(\frac{H_j}{P_j(\mathbf{q})}\right).$$

We introduce the notation

$$\Delta_i(\ell) = \sum_{j=1}^{n} \delta_j h_j(x_i, \ell),$$

and then rewrite the gain as follows using Eq. (32):

$$\begin{aligned} &B_F\left(\mathbf{0} \ \|\ (\mathbf{M}\boldsymbol{\delta}) \circ \mathbf{q}\right) - B_F\left(\mathbf{0} \ \|\ \mathbf{q}\right) \\ &= \sum_{i=1}^{m} \ln\left(q_{i,y_i} + \sum_{\ell \neq y_i} q_{i,\ell} \exp\left(-\sum_{j=1}^{n} \delta_j M_{(i,\ell),j}\right)\right) \\ &= -\sum_{i=1}^{m} \Delta_i(y_i) \\ &\quad + \sum_{i=1}^{m} \ln\left[e^{\Delta_i(y_i)}\left(q_{i,y_i} + \sum_{\ell \neq y_i} q_{i,\ell} e^{-\sum_{j=1}^{n} \delta_j M_{(i,\ell),j}}\right)\right]. \end{aligned} \tag{38}$$

Plugging in definitions, the first term of Eq. (38) can be written as

$$\begin{aligned} \sum_{i=1}^{m} \Delta_i(y_i) &= \sum_{j=1}^{n} \left[\ln\left(\frac{H_j}{P_j(\mathbf{q})}\right) \sum_{i=1}^{m} h_j(x_i, y_i)\right] \\ &= \sum_{j=1}^{n} H_j \ln\left(\frac{H_j}{P_j(\mathbf{q})}\right). \end{aligned} \tag{39}$$

Next we derive an upper bound on the second term of Eq. (38):

$$\begin{aligned} &\sum_{i=1}^{m} \ln\left[e^{\Delta_i(y_i)}\left(q_{i,y_i} + \sum_{\ell \neq y_i} q_{i,\ell} e^{-\sum_{j=1}^{n} \delta_j M_{(i,\ell),j}}\right)\right] \\ &= \sum_{i=1}^{m} \ln\left(q_{i,y_i} e^{\Delta_i(y_i)} + \sum_{\ell \neq y_i} q_{i,\ell} e^{\Delta_i(\ell)}\right) \\ &= \sum_{i=1}^{m} \ln\left(\sum_{\ell \in \mathcal{Y}} q_{i,\ell} e^{\Delta_i(\ell)}\right) \end{aligned}$$

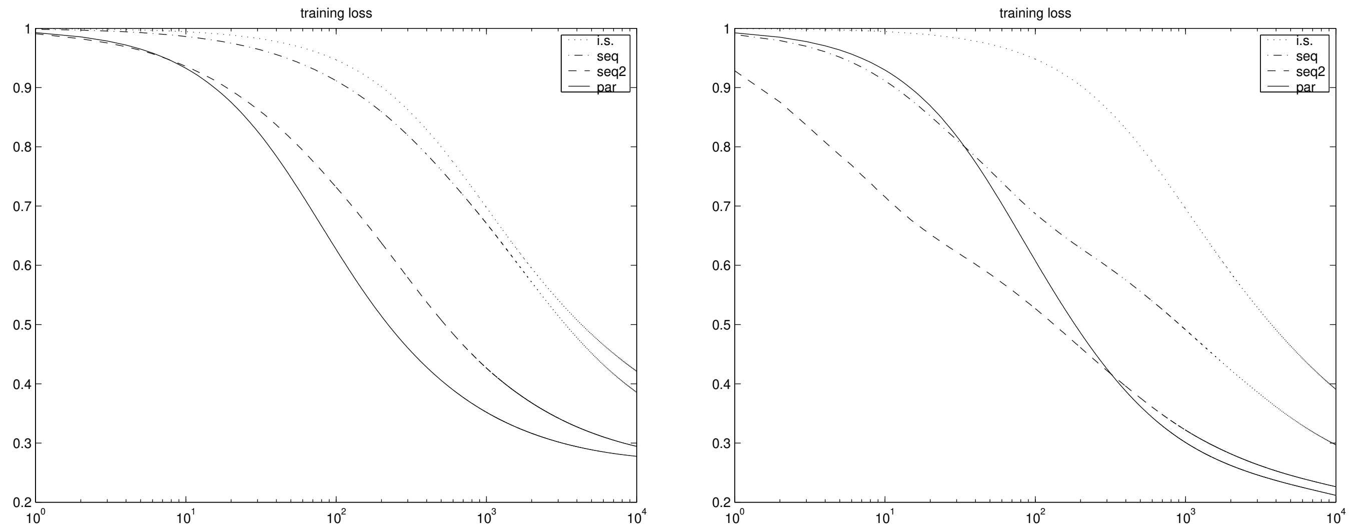

Figure 4: The training logistic loss on data generated by a noisy hyperplane with many (left) or few (right) relevant features.

$$
\begin{aligned}
&\le \sum_{i=1}^{m} \left(\sum_{\ell \in \mathcal{Y}} q_{i,\ell} e^{\Delta_i(\ell)} - 1 \right) && (40) \\
&= \sum_{i=1}^{m} \sum_{\ell \in \mathcal{Y}} q_{i,\ell} \left[\exp\left(\sum_{j=1}^{n} h_j(x_i,\ell)\delta_j \right) - 1 \right] && (41) \\
&\le \sum_{i=1}^{m} \sum_{\ell \in \mathcal{Y}} q_{i,\ell} \sum_{j=1}^{n} h_j(x_i,\ell)(e^{\delta_j} - 1) && (42) \\
&= \sum_{i=1}^{m} \sum_{\ell \in \mathcal{Y}} q_{i,\ell} \sum_{j=1}^{n} h_j(x_i,\ell) \left(\frac{H_j}{P_j(\mathbf{q})} - 1 \right) && (43) \\
&= \sum_{j=1}^{n} \left(\frac{H_j}{P_j(\mathbf{q})} - 1 \right) \sum_{i=1}^{m} \sum_{\ell \in \mathcal{Y}} q_{i,\ell} h_j(x_i,\ell) \\
&= \sum_{j=1}^{n} (H_j - P_j(\mathbf{q})) \,. && (44)
\end{aligned}
$$

Eq. (40) follows from the log bound $\ln x \le x - 1$. Eq. (42) uses Eq. (25) and our assumption on the form of the h_j's. Eq. (43) follows from our definition of the update $\boldsymbol{\delta}$.

Finally, combining Eqs. (36), (38), (39) and (44) gives Eq. (37) completing the proof. ■

It is clear that the differences between GIS and the updates given in this paper stem from Eq. (38), which is derived from $\ln x = -C + \ln\left(e^C x\right)$, with $C = \Delta_i(y_i)$ on the i'th term in the sum. This choice of C effectively means that the log bound is taken at a different point ($\ln x = -C + \ln\left(e^C x\right) \le -C + e^C x - 1$). In this more general case, the bound is exact at $x = e^{-C}$; hence, varying C varies where the bound is taken, and thereby varies the updates.

10 EXPERIMENTS

In this section, we briefly describe some experiments using synthetic data. These experiments are preliminary and are only intended to suggest the possibility of these algorithms' having practical value. More systematic experiments are clearly needed using both real-world and synthetic data, and comparing the new algorithms to other commonly used procedures.

We first tested how effective the methods are at minimizing the logistic loss on the training data. In the first experiment, we generated data using a very noisy hyperplane. More specifically, we first generated a random hyperplane in 100-dimensional space represented by a vector $\mathbf{w} \in \mathbb{R}^{100}$ (chosen uniformly at random from the unit sphere). We then chose 300 points $\mathbf{x} \in \mathbb{R}^{100}$ where each point is normally distributed $\mathbf{x} \sim N(\mathbf{0}, \mathbf{I})$. We next assigned a label y to each point depending on whether it fell above or below the chosen hyperplane, i.e., $y = \text{sign}(\mathbf{w} \cdot \mathbf{x})$. After each label was chosen, we perturbed each point $\mathbf{x}$ by adding to it a random amount ε where $\varepsilon \sim N(\mathbf{0}, 0.8\ \mathbf{I})$. This had the effect of causing the labels of points near the separating hyperplane to be more noisy than points that are farther from it. The features were identified with coordinates of $\mathbf{x}$.

We ran the parallel- and sequential-update algorithms of Sections 5 and 6 (denoted "par" and "seq" in the figures) on this data. We also ran the sequential-update algorithm that is a special case of the parameterized family described in Section 7 (denoted "seq2"). Finally, we ran the iterative scaling algorithm described in Section 9 ("i.s.").

The results of this experiment are shown on the left of Fig. 4 which shows a plot of the logistic loss on the training set for each of the four methods as a function of the number of iterations. (The loss has been normalized to be 1 when $\boldsymbol{\lambda} = \mathbf{0}$.) All of our methods do very well in comparison to iterative scaling. The parallel-update method is clearly the best, followed closely by the second sequential-update algorithm. The parallel-update method can be as much as 30 times faster (in terms of number of iterations) than iterative scaling.

On the right of Fig. 4 are shown the results of a similar experiment in which all but four of the components of $\mathbf{w}$ were forced to be zero. In other words, there were only four relevant variables or features. In this experiment, the sequential-update algorithms, which perform a kind of feature selection, initially have a significant advantage over the

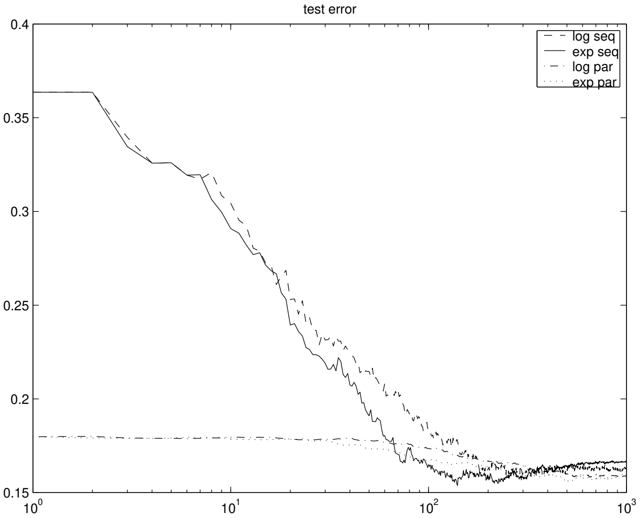

Figure 5: The test misclassification error on data generated by a noisy hyperplane with Boolean features.

parallel-update algorithm, but are eventually overtaken.

In the last experiment, we tested how effective the new competitors of AdaBoost are at minimizing the test misclassification error. In this experiment, we chose a separating hyperplane $\mathbf{w}$ as in the first experiment. Now, however, we chose 1000 points $\mathbf{x}$ uniformly at random from the Boolean hypercube $\{-1, +1\}^{100}$. The labels y were computed as before. After the labels y were chosen, we flipped each coordinate of each point $\mathbf{x}$ independently with probability 0.05. This noise model again has the effect of causing examples near the decision surface to be noisier than those far from it.

For this experiment, we used the parallel- and sequential-update algorithms of Sections 5 and 6 (denoted "par" and "seq"). In both cases, we used variants based on exponential loss ("exp") and logistic loss ("log"). (In this case, the sequential-update algorithms of Sections 6 and 7 are identical.)

Fig. 5 shows a plot of the classification error on a separate test set of 5000 examples. There is not a very large difference in the performance of the exponential and logistic variants of the algorithms. However, the parallel-update variants start out doing much better, although eventually all of the methods converge to roughly the same performance level.

ACKNOWLEDGMENTS

Many thanks to Manfred Warmuth for first teaching us about Bregman distances and for many comments on an earlier draft. Thanks also to Nigel Duffy, David Helmbold and Raj Iyer for helpful discussions and suggestions. Some of this research was done while Yoram Singer was at AT&T Labs.

References

[1] Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

[2] L. M. Bregman. The relaxation method of finding the common point of convex sets and its application to the solution of problems in convex programming. *U.S.S.R. Computational Mathematics and Mathematical Physics*, 7(1):200–217, 1967.

[3] Leo Breiman. Arcing the edge. Technical Report 486, Statistics Department, University of California at Berkeley, 1997.

[4] Leo Breiman. Prediction games and arcing classifiers. Technical Report 504, Statistics Department, University of California at Berkeley, 1997.

[5] Nicolò Cesa-Bianchi, Anders Krogh, and Manfred K. Warmuth. Bounds on approximate steepest descent for likelihood maximization in exponential families. *IEEE Transactions on Information Theory*, 40(4):1215–1220, July 1994.

[6] I. Csiszár. I-divergence geometry of probability distributions and minimization problems. *The Annals of Probability*, 3(1):146–158, 1975.

[7] I. Csiszár. Sanov property, generalized I-projection and a conditional limit theorem. *Annals of Probability*, 12:768–793, 1984.

[8] I. Csiszár. Maxent, mathematics, and information theory. In *Proceedings of the Fifteenth International Workshop on Maximum Entropy and Bayesian Methods*, pages 35–50, 1995.

[9] J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43(5):1470–1480, 1972.

[10] Stephen Della Pietra, Vincent Della Pietra, and John Lafferty. Inducing features of random fields. *IEEE Transactions Pattern Analysis and Machine Intelligence*, 19(4):1–13, April 1997.

[11] Carlos Domingo and Osamu Watanabe. Scaling up a boosting-based learner via adaptive sampling. In *Proceedings of the Fourth Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 2000.

[12] Nigel Duffy and David Helmbold. Potential boosters? In *Advances in Neural Information Processing Systems 11*, 1999.

[13] Yoav Freund and Robert 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.

[14] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, to appear.

[15] Klaus-U. Höffgen and Hans-U. Simon. Robust trainability of single neurons. In *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pages 428–439, July 1992.

[16] Jyrki Kivinen and Manfred K. Warmuth. Boosting as entropy projection. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, pages 134–144, 1999.

[17] John Lafferty. Additive models, boosting and inference for generalized divergences. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, pages 125–133, 1999.

[18] John D. Lafferty, Stephen Della Pietra, and Vincent Della Pietra. Statistical learning algorithms based on Bregman distances. In *Proceedings of the Canadian Workshop on Information Theory*, 1997.

[19] Llew Mason, Jonathan Baxter, Peter Bartlett, and Marcus Frean. Functional gradient techniques for combining hypotheses. In *Advances in Large Margin Classifiers*. MIT Press, 1999.

[20] Robert E. Schapire and Yoram Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):297–336, December 1999.

[21] F. Topsoe. Information theoretical optimization techniques. *Kybernetika*, 15:7–17, 1979.

[22] Osamu Watanabe. From computational learning theory to discovery science. In *Proceedings of the 26th International Colloquium on Automata, Languages and Programming*, pages 134–148, 1999.