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

July 3, 2001

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 are iterative and can be divided into two types based on whether the parameters are updated sequentially (one at a time) or in parallel (all at once). We also describe a parameterized family of algorithms that includes both a sequential- and a parallel-update algorithm as special cases, thus showing how the sequential and parallel approaches can themselves be unified. 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 the iterative scaling algorithm. We conclude with a few experimental results with synthetic data that highlight the behavior of the old and newly proposed algorithms in different settings.

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 very 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 similarity of the two problems 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, in the boosting literature, *weak hypotheses*. AdaBoost works by sequentially updating these parameters one by one. That is, on each of a series of iterations, a single feature (weak hypothesis) is chosen and the parameter associated with that single feature is adjusted. In contrast, methods for logistic regression, most notably iterative scaling (Darroch & Ratcliff, 1972; Della Pietra, Della Pietra, & Lafferty, 1997), 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. A few experiments described at the end of this paper 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. 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 (1997). 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 (1999) and Mason et al. (1999) 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 such assumptions.

Our unified view leads directly to a sequential-update algorithm for logistic regression that is only a minor modification of AdaBoost and which is very similar to the algorithm proposed by Duffy and Helmbold (1999). 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.

A potentially important advantage of the new algorithm for logistic regression 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 (1999) and Domingo and Watanabe (2000), 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. We speculate that our new algorithm may lead to a viable and mathematically cleaner alternative.

We next describe a parameterized family of iterative algorithms that includes both parallel- and sequential-update algorithms as well as a whole range of algorithms between these 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 (1972). 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 the iterative scaling algorithm. In Section 10, we discuss various notions of convergence of AdaBoost and relate our results to previous work on boosting. In Section 11, we give some initial experiments that demonstrate the qualitative behavior of the various algorithms in different settings.

Previous work

Variants of our sequential-update algorithms fit into the general family of "arcing" algorithms presented by Breiman (1997b, 1997a), as well as Mason et al.'s "AnyBoost" family of algorithms (Mason et al., 1999). The information-geometric view that we take also shows that some of the algorithms we study, including AdaBoost, fit into a family of algorithms described in 1967 by Bregman (1967), and elaborated upon by Censor and Lent (1981), for satisfying a set of constraints.[1]

Our work is based directly on the general setting of Lafferty, Della Pietra and Della Pietra (1997) 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 auxiliary 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 (2000) 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.

Duffy and Helmbold (1999) 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. This suggests that the logistic loss algorithm of section 6 of this paper, which is close to theirs, may turn out also to have the PAC boosting property. We leave this as an open problem.

Lafferty (1999) 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

[1] More specifically, Bregman (1967) and later Censor and Lent (1981) describe optimization methods based on Bregman distances where one constraint is satisfied at each iteration, for example, a method where the constraint which makes the most impact on the objective function is greedily chosen at each iteration. The simplest version of AdaBoost, which assumes weak hypotheses with values in $\{-1, +1\}$, is an algorithm of this type if we assume that the weak learner is always able to choose the weak hypothesis with minimum weighted error.

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 (1999) 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 an assumption that does not hold in general.[2] 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.

Cesa-Bianchi, Krogh and Warmuth (1994) describe an algorithm for a closely related problem to ours: minimization of a relative entropy subject to linear constraints. In related work, Littlestone, Long and Warmuth (1995) describe algorithms where convergence properties are analyzed through a method that is similar to the auxiliary function techniques. A variety of work in the online learning literature, such as the work by Littlestone, Long, and Warmuth (1995) and the work by Kivinen and Warmuth (1997, to appear) on exponentiated gradient methods, also use Bregman divergences, and techniques that are related to the auxiliary function method.

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 Maximum-Entropy 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, a natural goal is to try to match the sign of $f_\lambda(x_i)$ to y_i, that is, to attempt to minimize

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

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, (Höffgen & Simon, 1992)). It is therefore often advantageous to instead minimize some other nonnegative loss function. For instance, the boosting algorithm AdaBoost (Freund & Schapire, 1997; Schapire & Singer, 1999) 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.

AdaBoost is usually described as a procedure that works together with an oracle or subroutine called the *weak learner*. Briefly, on each of a series of rounds, the weak learner picks one feature (weak hypothesis)

[2] Specifically, their assumption is equivalent to the infimum of the exponential loss being strictly positive (when the data is separable it can be shown that the infimum is zero).

h_j. Note that the features $h_1, \ldots, h_n$ correspond to the *entire* space of weak hypotheses rather than merely the weak hypotheses that were previously found up to that point by the weak learner. Of course, this will often be an enormous space, but one, nevertheless, that can be discussed mathematically. In practice, it may often be necessary to rely on a weak learner that can only approximately search the entire space. For instance, greedy algorithms such as C4.5 are often used for this purpose to find a "good" decision tree from the space of all possible decision trees.

To simplify the discussion, let us suppose for the moment that all of the weak hypotheses are Boolean, i.e., with range $\{-1, +1\}$. In this case, the weak learner attempts to choose the weak hypothesis with smallest error rate, that is, with the smallest weighted number of mistakes (in which $h_j(x_i) \neq y_i$) relative to a distribution over training examples selected by AdaBoost. Given the choice of weak hypothesis h_j, AdaBoost then updates the associated parameter λ_j by adding some value α to it where α is a simple formula of this weighted error rate (note that a parameter may be updated more than once in this framework).

As mentioned above, in practice, the weak learner may not always succeed in finding the "best" h_j (in the sense of minimizing weighted error rate), for instance, if the size of the space of weak hypotheses precludes an exhaustive search. However, in this paper, we make the idealized assumption that the weak learner always chooses the best h_j. Given this assumption, it has been noted by Breiman (1997a, 1997b) and various later authors (Friedman et al., 2000; Mason et al., 1999; Rätsch, Onoda, & Müller, 2001; Schapire & Singer, 1999) that the choice of both h_j and α are done in such a way as to cause the greatest decrease in the exponential loss induced by $\boldsymbol{\lambda}$, given that only a single component of $\boldsymbol{\lambda}$ is to be updated. 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, as noted above, that the weak learner always chooses the "best" h_j).

In practice, early stopping (limiting the number of rounds of boosting, rather than running the algorithm to convergence) is often used to mitigate problems with overtraining. In this case the sequential algorithms in this paper can be considered to be feature selection methods, in that only a subset of the parameters will obtain non-zero values. Thus, the sequential methods can be used both for feature selection, or for search for the minimum of the loss function.

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 in some situations this parallel-update algorithm will be faster than the sequential-update algorithm. See Section 11 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 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 (Darroch & Ratcliff, 1972; Della Pietra et al., 1997) 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 11.

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 (1997).

Let $F : \Delta \to \mathbb{R}$ be a strictly convex function defined on a closed, convex set $\Delta \subseteq \mathbb{R}^m$. Assume F is differentiable at all points of Δ_{int}, the interior of Δ, which we assume is nonempty. The *Bregman distance* associated with F is defined for $\mathbf{p} \in \Delta$ and $\mathbf{q} \in \Delta_{\text{int}}$ to be

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

Thus, B_F measures the difference between F and its first-order Taylor expansion about $\mathbf{q}$, evaluated at $\mathbf{p}$. Bregman distances, first introduced by Bregman (1967), generalize some commonly studied distance measures. For instance, when $\Delta = \mathbb{R}^m_+$ and

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

B_F becomes the (unnormalized) relative entropy

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

(We follow the standard convention that $0 \log 0 = 0$.) Generally, although not always a metric or even symmetric, it can be shown that every Bregman distance is nonnegative and is equal to zero if and only if its two arguments are equal. We assume that B_F can be extended to a continuous extended real-valued function over all of $\Delta \times \Delta$.

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. In other words, the problem is to project $\mathbf{q}_0$ onto a linear subspace. The constraints defining the linear subspace are specified by some $m \times n$ matrix $\mathbf{M}$ and some vector $\tilde{\mathbf{p}} \in \Delta$. The vectors $\mathbf{p}$ satisfying these constraints are those for which

$$\mathbf{p}^{\mathrm{T}}\mathbf{M} = \tilde{\mathbf{p}}^{\mathrm{T}}\mathbf{M}. \tag{5}$$

This slightly odd way of writing the constraints ensures that the linear subspace is nonempty (i.e., there is at least one solution, $\mathbf{p} = \tilde{\mathbf{p}}$). Thus, the problem is to find

$$\arg\min_{\mathbf{p} \in \mathcal{P}} B_F(\mathbf{p} \parallel \mathbf{q}_0) \tag{6}$$

where

$$\mathcal{P} \doteq \left\{ \mathbf{p} \in \Delta : \mathbf{p}^{\mathrm{T}}\mathbf{M} = \tilde{\mathbf{p}}^{\mathrm{T}}\mathbf{M} \right\}. \tag{7}$$

At this point, we introduce a function $\mathcal{L}_F$ and a set $\mathcal{Q} \subseteq \Delta$ which are intimately related to the optimization problem in Eqs. (6) and (7). After giving formal definitions, we give informal arguments—through the use of Lagrange multipliers—for the relationships between $\mathcal{P}$, $\mathcal{Q}$ and $\mathcal{L}_F$. Finally, we state Theorem 1, which gives a complete connection between these concepts, and whose results will be used throughout this paper.

Let us define the function $\mathcal{L}_F : \Delta_{\text{int}} \times \mathbb{R}^m \to \Delta_{\text{int}}$ to be

$$\mathcal{L}_F(\mathbf{q}, \mathbf{v}) = (\nabla F)^{-1}(\nabla F(\mathbf{q}) - \mathbf{v}).$$

In order for this to be mathematically sound, we assume that ∇F is a bijective (one-to-one and onto) mapping from Δ_{int} to $\mathbb{R}^m$ so that its inverse $(\nabla F)^{-1}$ is defined. It is straightforward to verify that $\mathcal{L}_F$ has the following "additive" property:

$$\mathcal{L}_F(\mathcal{L}_F(\mathbf{q}, \mathbf{w}), \mathbf{v}) = \mathcal{L}_F(\mathbf{q}, \mathbf{v} + \mathbf{w}) \tag{8}$$

for $\mathbf{q} \in \Delta_{\text{int}}$ and $\mathbf{v}, \mathbf{w} \in \mathbb{R}^m$. We assume that $\mathcal{L}_F$ can be extended to a continuous function mapping $\Delta \times \mathbb{R}^m$ into Δ. For instance, when B_F is unnormalized relative entropy, it can be verified that

$$\mathcal{L}_F(\mathbf{q}, \mathbf{v})_i = q_i e^{-v_i}. \tag{9}$$

Next, let $\mathcal{Q}$ be the set of all vectors of the form:

$$\mathcal{Q} \doteq \{\mathcal{L}_F(\mathbf{q}_0, \mathbf{M}\boldsymbol{\lambda}) \mid \boldsymbol{\lambda} \in \mathbb{R}^n\}. \tag{10}$$

We now return to the optimization problem in Eqs. (6) and (7), and describe informally how it can be solved in some cases using the method of Lagrange multipliers. To use this method, we start by forming the Lagrangian:

$$K(\mathbf{p}, \boldsymbol{\lambda}) = B_F\left(\mathbf{p} \parallel \mathbf{q}_0\right) + (\mathbf{p}^{\mathrm{T}}\mathbf{M} - \tilde{\mathbf{p}}^{\mathrm{T}}\mathbf{M})\boldsymbol{\lambda} \tag{11}$$

where $\boldsymbol{\lambda} \in \mathbb{R}^n$ is a vector of Lagrange multipliers. By the usual theory of Lagrange multipliers, the solution to the original optimization problem is determined by the saddle point of this Lagrangian, where the minimum should be taken with respect to the parameters $\mathbf{p}$, and the maximum should be taken with respect to the Lagrange multipliers $\boldsymbol{\lambda}$.

Differentiating $K(\mathbf{p}, \boldsymbol{\lambda})$ with respect to $\mathbf{p}$ and setting the result equal to zero gives

$$\nabla F(\mathbf{p}) = \nabla F(\mathbf{q}_0) - \mathbf{M}\boldsymbol{\lambda}. \tag{12}$$

from which it follows that

$$\mathbf{p} = \mathcal{L}_F(\mathbf{q}_0, \mathbf{M}\boldsymbol{\lambda}) \tag{13}$$

which implies that $\mathbf{p} \in \mathcal{Q}$.

Differentiating $K(\mathbf{p}, \boldsymbol{\lambda})$ with respect to $\boldsymbol{\lambda}$ and setting the result equal to zero simply implies that $\mathbf{p}$ must satisfy the constraints in Eq. (5), and hence that $\mathbf{p} \in \mathcal{P}$. So we have shown that finding a saddle point of the Lagrangian—and thereby solving the constrained optimization problem in Eqs. (6) and (7)—is equivalent to finding a point in $\mathcal{P} \cap \mathcal{Q}$.

Finally, if we plug Eq. (13) into the Lagrangian in Eq. (11), we are left with the problem of maximizing

$$K(\mathcal{L}_F(\mathbf{q}_0, \mathbf{M}\boldsymbol{\lambda}), \boldsymbol{\lambda}).$$

By straightforward algebra, it can be verified that this quantity is equal to

$$B_F\left(\tilde{\mathbf{p}} \parallel \mathbf{q}_0\right) - B_F\left(\tilde{\mathbf{p}} \parallel \mathcal{L}_F(\mathbf{q}_0, \mathbf{M}\boldsymbol{\lambda})\right).$$

In other words, because $B_F\left(\tilde{\mathbf{p}} \parallel \mathbf{q}_0\right)$ is constant (relative to $\boldsymbol{\lambda}$), the original optimization problem has been reduced to the "dual" problem of minimizing $B_F\left(\tilde{\mathbf{p}} \parallel \mathbf{q}\right)$ over $\mathbf{q} \in \mathcal{Q}$.

To summarize, we have argued informally that if there is a point $\mathbf{q}_\star$ in $\mathcal{P} \cap \mathcal{Q}$ then this point minimizes $B_F\left(\mathbf{p} \parallel \mathbf{q}_0\right)$ over $\mathbf{p} \in \mathcal{P}$ and also minimizes $B_F\left(\tilde{\mathbf{p}} \parallel \mathbf{q}\right)$ over $\mathbf{q} \in \mathcal{Q}$. It turns out, however, that $\mathcal{P} \cap \mathcal{Q}$ can sometimes be empty, in which case this method does not yield a solution. Nevertheless, if we instead use the closure of $\mathcal{Q}$, which, intuitively, has the effect of allowing some or all of the Lagrange multipliers to be infinite, then there will always exist a unique solution. That is, as stated in the next theorem, for a large

family of Bregman distances, $\mathcal{P} \cap \overline{\mathcal{Q}}$ always contains exactly one point, and that one point is the unique solution of both optimization problems (where we also extend the constraint set of the dual problem from $\mathcal{Q}$ to $\overline{\mathcal{Q}}$).

We take Theorem 1 from Lafferty, Della Pietra and Della Pietra (1997). We do not give the full details of the conditions that F must satisfy for this theorem to hold since these go beyond the scope of the present paper. Instead, we refer the reader to Della Pietra, Della Pietra and Lafferty (2001) for a precise formulation of these conditions and a complete proof. A proof for the case of (normalized) relative entropy is given by Della Pietra, Della Pietra and Lafferty (1997). Moreover, their proof requires very minor modifications for all of the cases considered in the present paper. Closely related results are given by Censor and Lent (1981) and Csiszár (1991, 1995). See also Censor and Zenios's book (1997).

Theorem 1 *Let* $\tilde{\mathbf{p}}$, $\mathbf{q}_0$, $\mathbf{M}$, Δ, F, B_F, $\mathcal{P}$ *and* $\mathcal{Q}$ *be as above. Assume* $B_F(\tilde{\mathbf{p}} \parallel \mathbf{q}_0) < \infty$. *Then for a large family of functions* F*, including all functions considered in this paper, there exists a unique* $\mathbf{q}_\star \in \Delta$ *satisfying:*

1. $\mathbf{q}_\star \in \mathcal{P} \cap \overline{\mathcal{Q}}$
2. $B_F(\mathbf{p} \parallel \mathbf{q}) = B_F(\mathbf{p} \parallel \mathbf{q}_\star) + B_F(\mathbf{q}_\star \parallel \mathbf{q})$ *for any* $\mathbf{p} \in \mathcal{P}$ *and* $\mathbf{q} \in \overline{\mathcal{Q}}$
3. $\mathbf{q}_\star = \arg\min_{\mathbf{q} \in \overline{\mathcal{Q}}} B_F(\tilde{\mathbf{p}} \parallel \mathbf{q})$
4. $\mathbf{q}_\star = \arg\min_{\mathbf{p} \in \mathcal{P}} B_F(\mathbf{p} \parallel \mathbf{q}_0)$.

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

Proof sketch: As noted above, a complete and general proof is given by Della Pietra, Della Pietra and Lafferty (2001). However, the proof given by Della Pietra, Della Pietra and Lafferty (1997) for normalized relative entropy can be modified very easily for all of the cases of interest in the present paper. The only step that needs slight modification is in showing that the minimum in part 3 exists. For this, we note in each case that the set

$$\{\mathbf{q} \in \Delta \mid B_F(\tilde{\mathbf{p}} \parallel \mathbf{q}) \leq B_F(\tilde{\mathbf{p}} \parallel \mathbf{q}_0)\}$$

is bounded. Therefore, we can restrict the minimum in part 3 to the intersection of $\overline{\mathcal{Q}}$ with the closure of this set. Since this smaller set is compact and since $B_F(\tilde{\mathbf{p}} \parallel \cdot)$ is continuous, the minimum must be attained at some point $\mathbf{q}$.

The rest of the proof is essentially identical (modulo superficial changes in notation). ■

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 $\mathcal{P} \cap \overline{\mathcal{Q}}$.

Part 2 of Theorem 1 is a kind of Pythagorean theorem that is often very useful (for instance, in the proof of the theorem), though not used directly in this paper.

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

$$\sum_{i=1}^{m} \exp\left(-y_i \sum_{j=1}^{n} \lambda_j h_j(x_i)\right) \tag{14}$$

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 $(\mathbf{M}\boldsymbol{\lambda})_i = \sum_{j=1}^{n} \lambda_j y_i h_j(x_i)$. We let 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, $\mathcal{L}_F(\mathbf{q}, \mathbf{v})$ is as given in Eq. (9). 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

$$\mathrm{D}_U\left(\mathbf{0} \;\|\; \mathbf{q}\right) = \sum_{i=1}^{m} q_i \tag{15}$$

so that $\mathrm{D}_U\left(\mathbf{0} \;\|\; \mathcal{L}_F(\mathbf{q}_0, \mathbf{M}\boldsymbol{\lambda})\right)$ is equal to Eq. (14). Thus, minimizing $\mathrm{D}_U\left(\mathbf{0} \;\|\; \mathbf{q}\right)$ over $\mathbf{q} \in \overline{\mathcal{Q}}$ is equivalent to minimizing Eq. (14). By Theorem 1, this is equivalent to finding $\mathbf{q} \in \overline{\mathcal{Q}}$ satisfying the constraints

$$\sum_{i=1}^{m} q_i M_{ij} = \sum_{i=1}^{m} q_i y_i h_j(x_i) = 0 \tag{16}$$

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

$$\sum_{i=1}^{m} \ln\left(1 + \exp\left(-y_i \sum_{j=1}^{n} \lambda_j h_j(x_i)\right)\right). \tag{17}$$

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,

$$\mathrm{D}_B\left(\mathbf{0} \;\|\; \mathbf{q}\right) = -\sum_{i=1}^{m} \ln(1 - q_i). \tag{18}$$

Parameters: $\Delta \subseteq \mathbb{R}_+^m$
$F : \Delta \to \mathbb{R}$ for which Theorem 1 holds and satisfying Conditions 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, $\sum_{j=1}^n |M_{ij}| \leq 1$

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

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

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

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

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

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

Figure 1: The parallel-update optimization algorithm.

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

$$\mathcal{L}_F(\mathbf{q}, \mathbf{v})_i = \frac{q_i e^{-v_i}}{1 - q_i + q_i e^{-v_i}} \tag{19}$$

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, $\mathrm{D}_B\left(\mathbf{0} \parallel \mathcal{L}_F(\mathbf{q}_0, \mathbf{M}\boldsymbol{\lambda})\right)$ is equal to Eq. (17) so minimizing $\mathrm{D}_B\left(\mathbf{0} \parallel \mathbf{q}\right)$ over $\mathbf{q} \in \overline{\mathcal{Q}}$ is equivalent to minimizing Eq. (17). As before, this is the same as finding $q \in \overline{\mathcal{Q}}$ satisfying the constraints in Eq. (16).

Thus, the exponential loss and logistic loss problems fit into our general framework using nearly identical settings of the parameters. The main difference is in the choice of Bregman distance—unnormalized relative entropy for exponential loss and binary relative entropy for logistic loss. The former measures distance between nonnegative vectors representing weights over the instances, while the latter measures distance between distributions on possible labels, summed over all of the instances.

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

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 generally involved updates of one feature at a time. Parallel-update methods for *LogLoss* are well known (see, for example, (Darroch & Ratcliff, 1972; Della Pietra et al., 1997)). However, our updates take a different form from the usual updates derived for logistic models. We discuss the differences in Section 9.

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),

$$\begin{aligned} \mathbf{q}_{t+1} = \mathcal{L}_F(\mathbf{q}_0, \mathbf{M}(\boldsymbol{\lambda}_t + \boldsymbol{\delta}_t)) &= \mathcal{L}_F(\mathcal{L}_F(\mathbf{q}_0, \mathbf{M}\boldsymbol{\lambda}_t), \mathbf{M}\boldsymbol{\delta}_t) \\ &= \mathcal{L}_F(\mathbf{q}_t, \mathbf{M}\boldsymbol{\delta}_t). \end{aligned} \tag{20}$$

This gives

$$q_{t+1,i} = \begin{cases} q_{t,i} \exp\left(-\sum_{j=1}^{n} \delta_{t,j} M_{ij}\right) & for ExpLoss \\ q_{t,i}\Big[(1 - q_{t,i}) \exp\left(\sum_{j=1}^{n} \delta_{t,j} M_{ij}\right) + q_{t,i}\Big]^{-1} & for LogLoss. \end{cases} \tag{21}$$

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 Theorem 1 and the following conditions:

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

$$B_F\left(\mathbf{0} \parallel \mathcal{L}_F(\mathbf{q}, \mathbf{v})\right) - B_F\left(\mathbf{0} \parallel \mathbf{q}\right) \leq \sum_{i=1}^{m} q_i (e^{-v_i} - 1).$$

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

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

is bounded.

We will show later that the choices of F given in Section 4 satisfy these conditions 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 (1997). 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:

$$B_F\left(\mathbf{0} \parallel \mathbf{q}_{t+1}\right) - B_F\left(\mathbf{0} \parallel \mathbf{q}_t\right) \leq A(\mathbf{q}_t) \leq 0 \tag{22}$$

and

$$A(\mathbf{q}) = 0 \Rightarrow \mathbf{q}^{\mathrm{T}}\mathbf{M} = \mathbf{0}^{\mathrm{T}} . \tag{23}$$

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. (10). Assum F satisfies Theorem 1. Then*

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

Note that the $\mathbf{q}_t$'s will lie in a compact subspace of $\mathcal{Q}$ if Condition 2 holds and $B_F\left(\mathbf{0} \parallel \mathbf{q}_1\right) < \infty$. In the algorithm in Figure 1, and in general in the algorithms in this paper, $\boldsymbol{\lambda}_1 = \mathbf{0}$, so that $\mathbf{q}_1 = \mathbf{q}_0$ and the condition $B_F\left(\mathbf{0} \parallel \mathbf{q}_0\right) < \infty$ implies $B_F\left(\mathbf{0} \parallel \mathbf{q}_1\right) < \infty$. $B_F\left(\mathbf{0} \parallel \mathbf{q}_0\right) < \infty$ is an input condition for all of the algorithms in this paper.

Proof: By condition (22), $B_F\left(\mathbf{0} \parallel \mathbf{q}_t\right)$ is a nonincreasing sequence. As is the case for all Bregman distances, $B_F\left(\mathbf{0} \parallel \mathbf{q}_t\right)$ is also 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. Using the condition of Eq. (22), 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, $\hat{\mathbf{q}} \in \mathcal{P}$ from the condition given by Eq. (23), where $\mathcal{P}$ is as in Eq. (7). 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 Theorem 1 and Conditions 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 $\mathcal{Q}$ is as in Eq. (10). That is,

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

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:

$$\begin{aligned} B_F\left(\mathbf{0} \parallel \mathbf{q}_{t+1}\right) - B_F\left(\mathbf{0} \parallel \mathbf{q}_t\right) &= B_F\left(\mathbf{0} \parallel \mathcal{L}_F(\mathbf{q}_t, \mathbf{M}\boldsymbol{\delta}_t)\right) - B_F\left(\mathbf{0} \parallel \mathbf{q}_t\right) && (24) \\ &\leq \sum_{i=1}^{m} q_{t,i} \left[\exp\left(-\sum_{j=1}^{n} \delta_{t,j} M_{ij} \right) - 1 \right] && (25) \\ &= \sum_{i=1}^{m} q_{t,i} \left[\exp\left(-\sum_{j=1}^{n} \delta_{t,j} s_{ij} |M_{ij}| \right) - 1 \right] \\ &\leq \sum_{i=1}^{m} q_{t,i} \left[\sum_{j=1}^{n} |M_{ij}| (e^{-\delta_{t,j} s_{ij}} - 1) \right] && (26) \\ &= \sum_{j=1}^{n} \left(W_{t,j}^+ e^{-\delta_{t,j}} + W_{t,j}^- e^{\delta_{t,j}} - W_{t,j}^+ - W_{t,j}^- \right) && (27) \\ &= -\sum_{j=1}^{n} \left(\sqrt{W_{t,j}^+} - \sqrt{W_{t,j}^-} \right)^2 = A(\mathbf{q}_t). && (28) \end{aligned}$$

Eqs. (24) and (25) follow from Eq. (20) and Condition 1, respectively. Eq. (26) uses the fact that, for any x_j's and for $p_j \geq 0$ with $\sum_j p_j \leq 1$, we have

$$\begin{aligned} \exp\left(\sum_j p_j x_j \right) - 1 &= \exp\left(\sum_j p_j x_j + 0 \cdot \left(1 - \sum_j p_j \right) \right) - 1 \\ &\leq \sum_j p_j e^{x_i} + \left(1 - \sum_j p_j \right) - 1 = \sum_j p_j (e^{x_i} - 1) && (29) \end{aligned}$$

by Jensen's inequality applied to the convex function e^x. Eq. (27) uses the definitions of $W_{t,j}^+$ and $W_{t,j}^-$, and Eq. (28) uses our choice of $\boldsymbol{\delta}_t$ (indeed, $\boldsymbol{\delta}_t$ was chosen specifically to minimize Eq. (27)).

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 Conditions 1 and 2 are satisfied. Starting with Condition 1, for *ExpLoss*, we have

$$\mathrm{D}_U\left(\mathbf{0} \;\|\; \mathcal{L}_F(\mathbf{q},\mathbf{v}))\right) - \mathrm{D}_U\left(\mathbf{0} \;\|\; \mathbf{q}\right) = \sum_{i=1}^m q_i e^{-v_i} - \sum_{i=1}^m q_i.$$

For *LogLoss*,

$$\begin{aligned}
\mathrm{D}_B\left(\mathbf{0} \;\|\; \mathcal{L}_F(\mathbf{q},\mathbf{v})\right) - \mathrm{D}_B\left(\mathbf{0} \;\|\; \mathbf{q}\right) &= \sum_{i=1}^m \ln\left(\frac{1-q_i}{1-(\mathcal{L}_F(\mathbf{q},\mathbf{v}))_i}\right) \\
&= \sum_{i=1}^m \ln\left(1 - q_i + q_i e^{-v_i}\right) \\
&\leq \sum_{i=1}^m \left(-q_i + q_i e^{-v_i}\right).
\end{aligned}$$

The first and second equalities use Eqs. (18) and (19), respectively. The final inequality uses $1 + x \leq e^x$ for all x.

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

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

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

Note that while Condition 1 holds for the loss functions we are considering, it may not hold for all Bregman distances. Lafferty, Della Pietra and Della Pietra (1997) describe parallel update algorithms for Bregman distances, using the auxiliary function technique. Their method does not require Condition 1, and therefore applies to arbitrary Bregman distances; however, each iteration of the algorithm requires solution of a system of equations that requires a numerical search technique such as Newton's method.

6 Sequential algorithms

In this section, we describe another algorithm for the 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. On each round, a single feature j_t is first chosen to maximize the inner product of the corresponding column of the matrix $\mathbf{M}$ with the vector $\mathbf{q}_t$. The quantity α_t is then computed and added to the j_t'th component of $\boldsymbol{\lambda}$.

It may seem surprising or even paradoxical that the algorithm does not explicitly guarantee that all of the components of $\boldsymbol{\lambda}$ are eventually updated, and yet we are able to prove convergence to optimality. Apparently, all components which "need" to be nonzero will eventually be selected by the algorithm for updating. Moreover, on each iteration, although only one component is actually updated, in fact, *all* of the components are *considered* for updating which means that all of them are implicitly used in the computation of the eventual update to $\boldsymbol{\lambda}$.

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

Input: Matrix $\mathbf{M} \in [-1, 1]^{m \times n}$
Output: $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots$ such that

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

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

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

Figure 2: The sequential-update optimization algorithm.

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

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

This function is clearly continuous and nonpositive. We have that

$$\begin{aligned}
B_F\left(\mathbf{0} \parallel \mathbf{q}_{t+1}\right) - B_F\left(\mathbf{0} \parallel \mathbf{q}_t\right) &\leq \sum_{i=1}^m q_{t,i} \left(\exp\left(-\sum_{j=1}^n \delta_{t,j} M_{ij} \right) - 1 \right) \\
&= \sum_{i=1}^m q_{t,i} \left(\exp\left(-\alpha_t M_{ij_t}\right) - 1 \right) \qquad (30) \\
&\leq \sum_{i=1}^m q_{t,i} \left(\frac{1 + M_{ij_t}}{2} e^{-\alpha_t} + \frac{1 - M_{ij_t}}{2} e^{\alpha_t} - 1 \right) \qquad (31) \\
&= \frac{Z_t + r_t}{2} e^{-\alpha_t} + \frac{Z_t - r_t}{2} e^{\alpha_t} - Z_t \qquad (32) \\
&= \sqrt{Z_t^2 - r_t^2} - Z_t = A(\mathbf{q}_t) \qquad (33)
\end{aligned}$$

where Eq. (31) uses the convexity of $e^{-\alpha_t x}$, and Eq. (33) uses our choice of α_t (as before, we chose α_t to minimize the bound in Eq. (32)).

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 (1997). 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 (1999) 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 $\boldsymbol{\lambda}_t$. Thus, we could easily convert any system such as SLIPPER (Cohen & Singer, 1999), BoosTexter (Schapire & Singer, 2000) or alternating trees (Freund & Mason, 1999) to use logistic loss. We can even do this for systems based on "confidence-rated" boosting (Schapire & Singer, 1999) in which α_t and j_t are chosen together on each round to minimize Eq. (30) 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-update 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. Thus, in this section, we show how the parallel and sequential viewpoints can themselves be unified in a manner that admits a unified presentation and unified convergence proofs. Moreover, the family of algorithms that we present includes a number of new algorithms including, as just mentioned, a sequential-update algorithm that, in our experiments, consistently performed better than the one in Section 6. This family of algorithms also includes other algorithms that may in certain situations be more appropriate than any of the algorithms presented up to this point. For instance, one of these algorithms is tailored for the case when the Euclidean norm of each row of the matrix $\mathbf{M}$ is bounded by a constant, in other words, for when the feature-vectors associated with the examples are known to lie in a Euclidean ball (centered at the origin) of bounded radius.

Parameters: $\Delta \subseteq \mathbb{R}_+^m$
$F : \Delta \to \mathbb{R}$ for which Theorem 1 holds and satisfying Conditions 1 and 2
$\mathbf{q}_0 \in \Delta$ such that $B_F\left(\mathbf{0} \;\|\; \mathbf{q}_0\right) < \infty$
$\mathcal{A} \subseteq \mathbb{R}_+^n$

Input: Matrix $\mathbf{M} \in \mathbb{R}^{m \times n}$ satisfying the condition that if we define

$$\mathcal{A}_{\mathbf{M}} \doteq \{\mathbf{a} \in \mathcal{A} \mid \forall i : \sum_j a_j |M_{ij}| \leq 1\}$$

then $\forall 1 \leq j \leq n, \exists \mathbf{a} \in \mathcal{A}_{\mathbf{M}}$ for which $a_j > 0$

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

$$\lim_{t \to \infty} B_F\left(\mathbf{0} \;\|\; \mathcal{L}_F(\mathbf{q}_0, \mathbf{M}\boldsymbol{\lambda}_t)\right) = \inf_{\lambda \in \mathbb{R}^n} B_F\left(\mathbf{0} \;\|\; \mathcal{L}_F(\mathbf{q}_0, \mathbf{M}\boldsymbol{\lambda})\right).$$

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

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

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

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

Figure 3: A parameterized family of iterative optimization algorithms.

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 $\mathcal{A}_{\mathbf{M}}$. The allowed form of these scaling vectors is given by the set $\mathcal{A}$, a parameter of the algorithm; $\mathcal{A}_{\mathbf{M}}$ 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 $\mathcal{A} = \{c\mathbf{1} \mid c > 0\}$.) An alternative is to not restrict the scaling vectors at all, i.e., we set $\mathcal{A}$ to be $\mathbb{R}_+^n$. In this case, finding $\mathbf{a}_t$ is a linear programming problem with n variables and m constraints, and the features are dynamically scaled to make

optimal progress on each iteration. There may be computational reasons for doing this, in that the rate of convergence may depend on the relative scaling of the features.

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 ($\frac{1}{p} + \frac{1}{q} = 1$).

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:

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

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

$$\hat{\Pr}[y|x] = \frac{e^{f_\lambda(x,y)}}{\sum_{\ell \in \mathcal{Y}} e^{f_\lambda(x,\ell)}} = \frac{1}{1 + \sum_{\ell \neq y} e^{f_\lambda(x,\ell) - f_\lambda(x,y)}} \tag{34}$$

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

$$\sum_{i=1}^{m} \ln \left[1 + \sum_{\ell \neq y_i} e^{f_\lambda(x_i,\ell) - f_\lambda(x_i,y_i)} \right]. \tag{35}$$

We transform this into our framework as follows: Let

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

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

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

The resulting Bregman distance is

$$B_F(\mathbf{p} \parallel \mathbf{q}) = \sum_{i=1}^{m} \left[\sum_{\ell \neq y_i} p_{i,\ell} \ln \left(\frac{p_{i,\ell}}{q_{i,\ell}} \right) + (1 - \bar{p_i}) \ln \left(\frac{1 - \bar{p_i}}{1 - \bar{q_i}} \right) \right].$$

This distance measures the relative entropy between the distributions over labels for instance i defined by $\mathbf{p}$ and $\mathbf{q}$, summed over all instances i. Clearly,

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

It can be shown that

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

Condition 1 can be verified by noting that

$$\begin{aligned} B_F\left(\mathbf{0} \;\|\; \mathcal{L}_F(\mathbf{q}, \mathbf{v})\right)) - B_F\left(\mathbf{0} \;\|\; \mathbf{q}\right) &= \sum_{i=1}^{m} \ln \left(\frac{1 - \bar{q_i}}{1 - \overline{(\mathcal{L}_F(\mathbf{q}, \mathbf{v}))_i}} \right) \\ &= \sum_{i=1}^{m} \ln \left(1 - \bar{q_i} + \sum_{\ell \neq y_i} q_{i,\ell} e^{-v_{i,\ell}} \right) \qquad (36) \\ &\leq \sum_{i=1}^{m} \left(-\bar{q_i} + \sum_{\ell \neq y_i} q_{i,\ell} e^{-v_{i,\ell}} \right) \\ &= \sum_{(i,\ell) \in \mathcal{B}} q_{i,\ell} (e^{-v_{i,\ell}} - 1). \end{aligned}$$

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 $B_F\left(\mathbf{0} \;\|\; \mathcal{L}_F(\mathbf{q}_0, \mathbf{M}\boldsymbol{\lambda})\right)$ is equal to Eq. (35). 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 (Freund & Schapire, 1997) (a special case of AdaBoost.MR (Schapire & Singer, 1999)), is based on the loss function

$$\sum_{(i,\ell) \in \mathcal{B}} \exp\left(f_\lambda(x_i, \ell) - f_\lambda(x_i, y_i)\right). \qquad (37)$$

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,

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

and

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

Choosing $\mathbf{M}$ as we did for multiclass logistic regression and $\mathbf{q}_0 = \mathbf{1}$, we have that $B_F\left(\mathbf{0} \;\|\; \mathcal{L}_F(\mathbf{q}_0, \mathbf{M}\boldsymbol{\lambda})\right)$ is equal to the loss in Eq. (37). 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 (Schapire & Singer, 1999) 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

$$\sum_{i=1}^{m} \sum_{\ell \in \mathcal{Y}} \exp\left(-\tilde{y}_{i,\ell}\, f_\lambda(x_i, \ell)\right). \tag{38}$$

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 (1972) for comparison to our algorithms. We largely follow the description of GIS given by Berger, Della Pietra and Della Pietra (1996) 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. (34) if $\mathbf{q} = \mathcal{L}_F(\mathbf{q}_0, \mathbf{M}\boldsymbol{\lambda})$.

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

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

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 (1997) also requires only these milder assumptions but is 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

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

where

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

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

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

$$\begin{aligned} I_j(\mathbf{q}) &= \sum_{i=1}^{m} \sum_{\ell \in \mathcal{Y}} q_{i,\ell}\, h_j(x_i, \ell) \\ &= \sum_{i=1}^{m} h_j(x_i, y_i) \\ &\quad + \sum_{i=1}^{m} \sum_{\ell \in \mathcal{Y}} q_{i,\ell}\ [h_j(x_i, \ell) - h_j(x_i, y_i)] \\ &= H_j - \sum_{(i,\ell) \in \mathcal{B}} q_{i,\ell} M_{(i,\ell),j} \\ &= H_j - (W_j^+(\mathbf{q}) - W_j^-(\mathbf{q}))\ , \end{aligned} \tag{39}$$

where we define $\mathcal{B} = \{(i, \ell) \mid 1 \leq i \leq m, \ell \in \mathcal{Y} - \{y_i\}\}$, as in the case of logistic regression.

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

$$\begin{aligned} A(\mathbf{q}) &\doteq -\mathrm{D}_U\left(\langle H_1, \ldots, H_n\rangle \ \|\ \langle I_1(\mathbf{q}), \ldots, I_n(\mathbf{q})\rangle\right) \\ &= -\sum_{j=1}^{n} \left(H_j \ln \frac{H_j}{I_j(\mathbf{q})} + I_j(\mathbf{q}) - H_j \right) \end{aligned} \tag{40}$$

is an auxiliary 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 $H_j = I_j(\mathbf{q})$ for all j. From Eq. (39), $H_j = I_j(\mathbf{q})$ if and only if $W_j^+(\mathbf{q}) = W_j^-(\mathbf{q})$. Thus, $A(\mathbf{q}) = 0$ implies that the constraints $\mathbf{q}^{\mathrm{T}}\mathbf{M} = \mathbf{0}^{\mathrm{T}}$ as in the proof of Theorem 3. All that remains to be shown is that

$$B_F\left(\mathbf{0} \ \|\ \mathcal{L}_F(\mathbf{q}, \mathbf{M}\boldsymbol{\delta})\right) - B_F\left(\mathbf{0} \ \|\ \mathbf{q}\right) \leq A(\mathbf{q}) \tag{41}$$

where

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

We introduce the notation

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

and then rewrite the left hand side of Eq. (41) as follows using Eq. (36):

$$\begin{aligned} B_F\left(\mathbf{0} \ \|\ \mathcal{L}_F(\mathbf{q}, \mathbf{M}\boldsymbol{\delta})\right) - B_F\left(\mathbf{0} \ \|\ \mathbf{q}\right) &= \sum_{i=1}^{m} \ln\left(q_{i,y_i} + \sum_{\ell \neq y_i} q_{i,\ell} \exp\left(-\sum_{j=1}^{n} \delta_j M_{(i,\ell),j}\right)\right) \\ &= -\sum_{i=1}^{m} \Delta_i(y_i) \\ &\quad + \sum_{i=1}^{m} \ln\left[e^{\Delta_i(y_i)}\left(q_{i,y_i} + \sum_{\ell \neq y_i} q_{i,\ell} e^{-\sum_{j=1}^{n} \delta_j M_{(i,\ell),j}}\right)\right]. \end{aligned} \tag{42}$$

Plugging in definitions, the first term of Eq. (42) can be written as

$$\begin{aligned}\sum_{i=1}^{m}\Delta_i(y_i) &= \sum_{j=1}^{n}\left[\ln\left(\frac{H_j}{I_j(\mathbf{q})}\right)\sum_{i=1}^{m}h_j(x_i,y_i)\right] \\ &= \sum_{j=1}^{n}H_j\ln\left(\frac{H_j}{I_j(\mathbf{q})}\right). \end{aligned} \tag{43}$$

Next we derive an upper bound on the second term of Eq. (42):

$$\begin{aligned}&\sum_{i=1}^{m}\ln\left[e^{\Delta_i(y_i)}\left(q_{i,y_i}+\sum_{\ell\neq y_i}q_{i,\ell}e^{-\sum_{j=1}^{n}\delta_j M_{(i,\ell),j}}\right)\right] \\ &\quad= \sum_{i=1}^{m}\ln\left(q_{i,y_i}e^{\Delta_i(y_i)}+\sum_{\ell\neq y_i}q_{i,\ell}e^{\Delta_i(\ell)}\right) \\ &\quad= \sum_{i=1}^{m}\ln\left(\sum_{\ell\in\mathcal{Y}}q_{i,\ell}e^{\Delta_i(\ell)}\right) \\ &\quad\leq \sum_{i=1}^{m}\left(\sum_{\ell\in\mathcal{Y}}q_{i,\ell}e^{\Delta_i(\ell)}-1\right) &(44)\\ &\quad= \sum_{i=1}^{m}\sum_{\ell\in\mathcal{Y}}q_{i,\ell}\left[\exp\left(\sum_{j=1}^{n}h_j(x_i,\ell)\delta_j\right)-1\right] &(45)\\ &\quad\leq \sum_{i=1}^{m}\sum_{\ell\in\mathcal{Y}}q_{i,\ell}\sum_{j=1}^{n}h_j(x_i,\ell)(e^{\delta_j}-1) &(46)\\ &\quad= \sum_{i=1}^{m}\sum_{\ell\in\mathcal{Y}}q_{i,\ell}\sum_{j=1}^{n}h_j(x_i,\ell)\left(\frac{H_j}{I_j(\mathbf{q})}-1\right) &(47)\\ &\quad= \sum_{j=1}^{n}\left(\frac{H_j}{I_j(\mathbf{q})}-1\right)\sum_{i=1}^{m}\sum_{\ell\in\mathcal{Y}}q_{i,\ell}h_j(x_i,\ell) \\ &\quad= \sum_{j=1}^{n}(H_j-I_j(\mathbf{q}))\,. &(48)\end{aligned}$$

Eq. (44) follows from the log bound $\ln x \leq x-1$. Eq. (46) uses Eq. (29) and our assumption on the form of the h_j's. Eq. (47) follows from our definition of the update $\boldsymbol{\delta}$.

Finally, combining Eqs. (40), (42), (43) and (48) gives Eq. (41) completing the proof. ■

It is clear that the differences between GIS and the updates given in this paper stem from Eq. (42), 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 ($\ln x = -C + \ln\left(e^C x\right) \leq -C + e^C x - 1$). 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 Discussion

In this section we discuss various notions of convergence of AdaBoost, relating the work in this paper to previous work on boosting, and in particular to previous work on the convergence properties of AdaBoost.

The algorithms in this paper define a sequence of parameter settings $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots$. There are various functions of the parameter settings, for which sequences are therefore also defined and for which convergence properties may be of interest. For instance, one can investigate convergence in value, i.e., convergence of the exponential loss function, as defined in Eq. (14); convergence of either the unnormalized distributions $\mathbf{q}_t$ or the normalized distributions $\mathbf{q}_t/(\sum_i q_i^t)$, over the training examples; and convergence in parameters, that is, convergence of $\boldsymbol{\lambda}_t$.

In this paper, we have shown that AdaBoost, and the other algorithms proposed, converge to the infimum of the exponential loss function. We have also shown that the unnormalized distribution converges to the distribution $\mathbf{q}_\star$ as defined in Theorem 1. The *normalized* distribution converges, provided that $\mathbf{q}_\star \neq \mathbf{0}$. In the case $\mathbf{q}_\star = \mathbf{0}$ the limit of $\mathbf{q}_t/(\sum_i q_i^t)$ is clearly not well defined.

Kivinen and Warmuth (1999) show that the normalized distribution converges in the case that $\mathbf{q}_\star \neq \mathbf{0}$. They also show that the resulting normalized distribution is the solution to

$$\min_{\mathbf{q} \in P_m, \mathbf{q}^{\mathrm{T}}\mathbf{M} = \mathbf{0}^{\mathrm{T}}} \mathrm{D}_R\left(\mathbf{q} \parallel \mathbf{q}_0\right) = \max_{\boldsymbol{\lambda} \in \mathbb{R}^n} \left(-\log\left(\mathit{ExpLoss}(\boldsymbol{\lambda})\right)\right)$$

Here P_m is the simplex over the m training examples (i.e., the space of possible normalized distributions); $\mathrm{D}_R\left(\mathbf{q} \parallel \mathbf{q}_0\right)$ is the relative entropy between distributions $\mathbf{q}$ and $\mathbf{q}_0$; and $\mathbf{q}_0$ is the uniform distribution over the training examples, $\mathbf{q}_0 = (1/m)\mathbf{1}$. This paper has discussed the properties of the unnormalized distribution: it is interesting that Kivinen and Warmuth's results imply analogous relations for the normalized distribution.

We should note that we have implicitly assumed in the algorithms that the weak learner can make use of an unnormalized distribution, rather than the normalized distribution over training examples that is usually used by boosting algorithms. We think this is a minor point though: indeed, there is nothing to prevent the normalized distribution being given to the weak learner instead (the algorithms would not change, and the normalized distribution is well defined unless $\sum q_i = 0$, in which case the algorithm has already converged). In our view, the use of the unnormalized rather than the normalized distribution is a minor change, although the use of the normalized distribution is perhaps more intuitive (for instance, the "edge" of a weak learner is defined with respect to the normalized distribution).

Finally, the convergence of the parameter values $\boldsymbol{\lambda}_t$ is problematic. In the case that $\mathbf{q}_\star = \mathbf{0}$, some of the parameter values must diverge to $+\infty$ or $-\infty$. In fact, the parameter values can diverge even if $\mathbf{q}_\star \neq \mathbf{0}$: all that is needed is that one or more of the components of $\mathbf{q}_\star$ be equal to zero. Even if $\mathbf{q}_\star$ is on the interior of Δ, there is no guarantee of convergence of the parameter values, for if the constraints are not linearly independent, there may be several parameter values which give the optimal point. Thus, the parameters may diverge under our assumptions, or even under the assumption that $\mathbf{q}_\star \neq \mathbf{0}$. This is problematic, as the values for $\boldsymbol{\lambda}$ are used to define the final hypothesis that is applied to test data examples.

11 Experiments

In this section, we briefly describe some experiments using synthetic data. We stress that 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.

In our experiments, we generated random data and classified it using a very noisy hyperplane. More specifically, in the 2-class case, 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 1000 points $\mathbf{x} \in \mathbb{R}^{100}$. In the case of real-valued features, each point was normally distributed $\mathbf{x} \sim N(\mathbf{0}, \mathbf{I})$. In the case of Boolean features, each point $\mathbf{x}$ was chosen uniformly at random from the Boolean hypercube

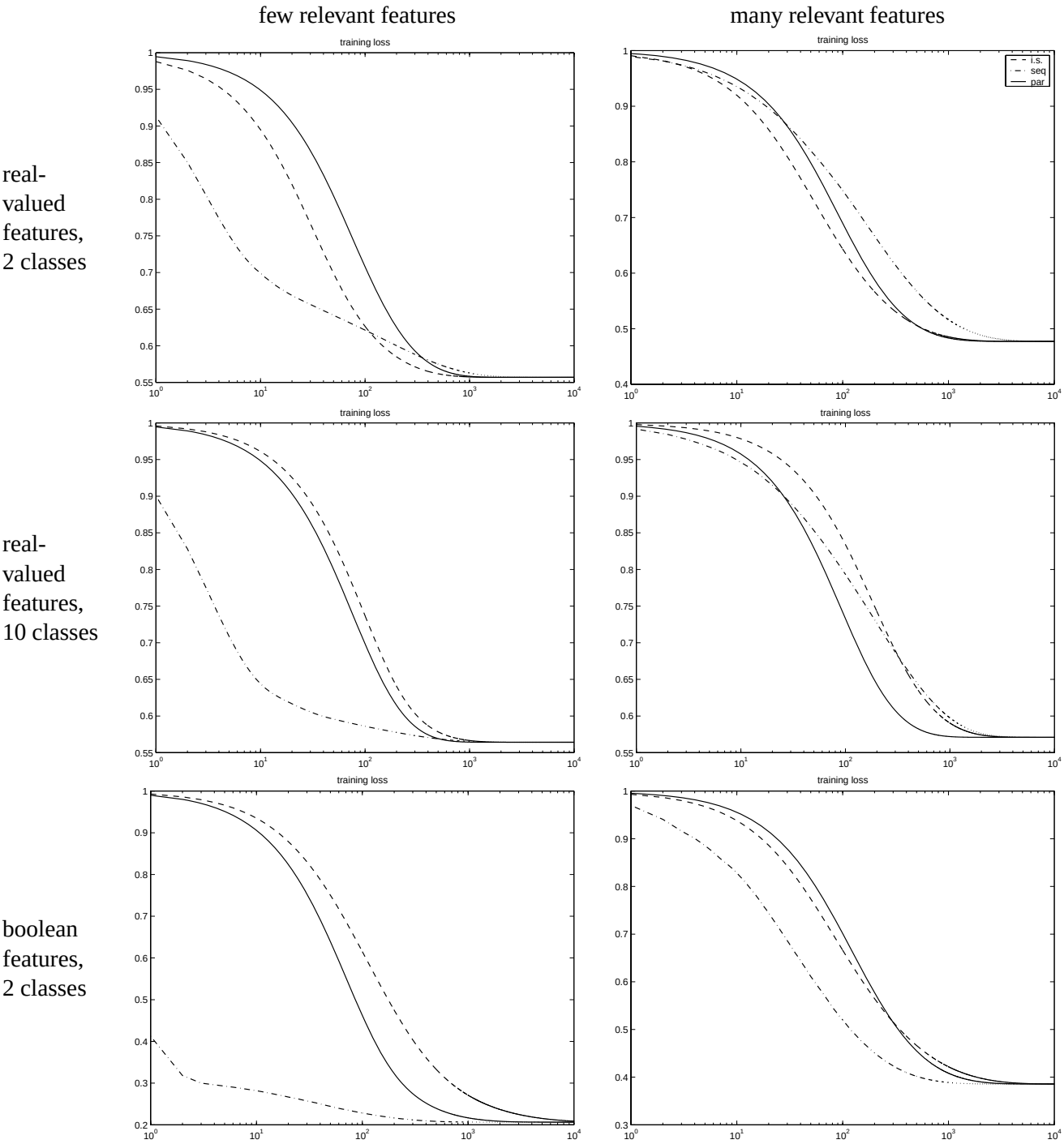

Figure 4: The training logistic loss on data generated by a noisy hyperplanes by various methods.

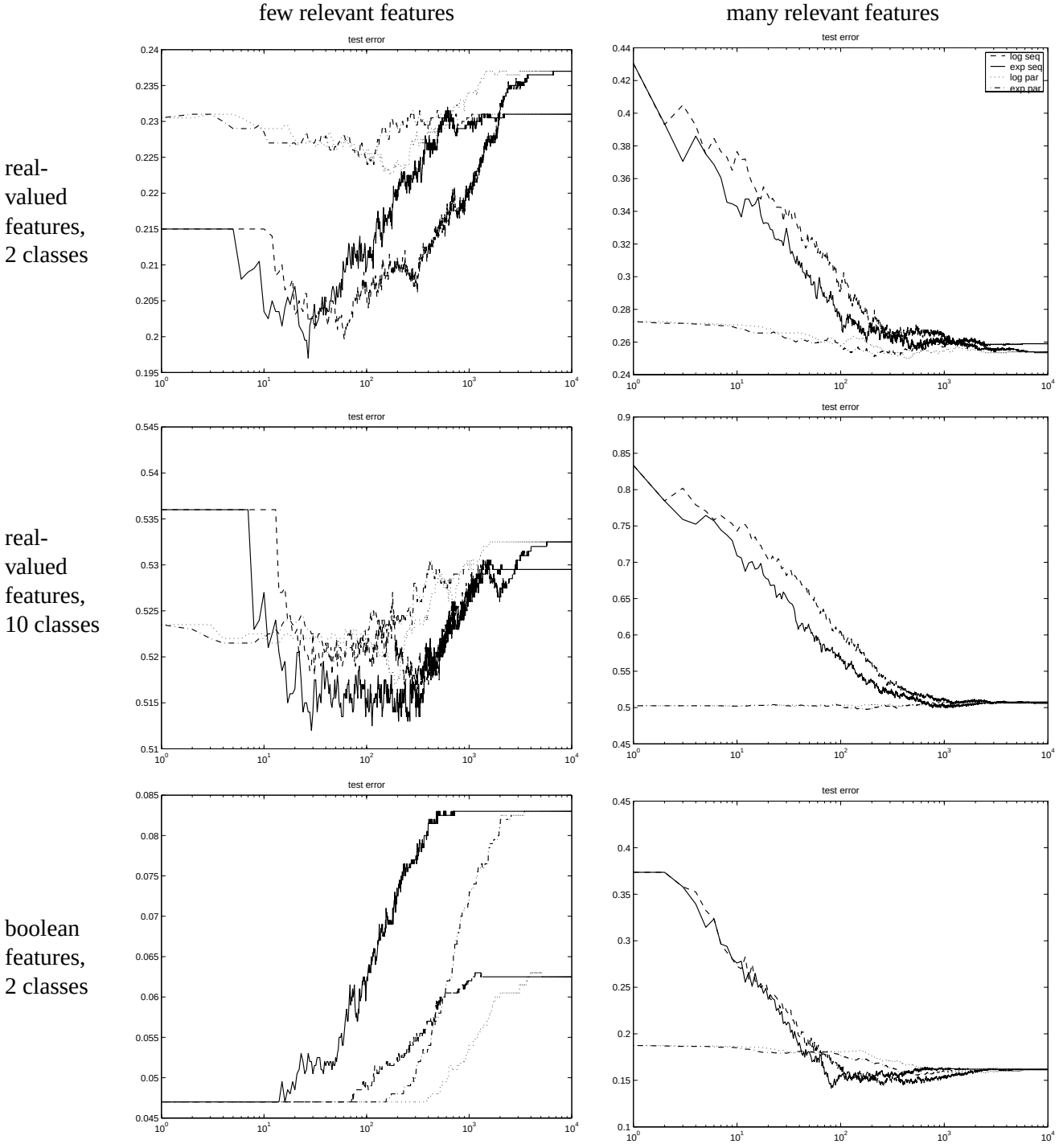

Figure 5: The test misclassification error on data generated by noisy hyperplanes.

$\{-1, +1\}^{100}$. 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 $\mathbf{x}$. In the case of real-valued features, we did this by adding a random amount $\boldsymbol{\varepsilon}$ to $\mathbf{x}$ where $\boldsymbol{\varepsilon} \sim N(\mathbf{0}, 0.8\ \mathbf{I})$. For Boolean features, we flipped each coordinate of $\mathbf{x}$ independently with probability 0.05. Note that both of these forms of perturbation have 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 $\mathbf{x}$.

For real-valued features, we also conducted a similar experiment involving ten classes rather than two. In this case, we generated ten random hyperplanes $\mathbf{w}_1, \ldots, \mathbf{w}_{10}$, each chosen uniformly at random from the unit sphere, and classified each point $\mathbf{x}$ by $\arg\max_y \mathbf{w}_y \cdot \mathbf{x}$ (prior to perturbing $\mathbf{x}$).

Finally, in some of the experiments, we limited each weight vector to depend on just 4 of the 100 possible features.

In the first set of experiments, we tested the algorithms to see how effective they are at minimizing the logistic loss on the training data. (We did not run corresponding experiments for exponential loss since typically we are not interested in minimizing exponential loss per se, but rather in using it as a proxy for some other quantity that we do want to minimize, such as the classification error rate.) We ran the parallel-update algorithm of Section 5 (denoted "par" in the figures), as well as the sequential-update algorithm that is a special case of the parameterized family described in Section 7 (denoted "seq"). Finally, we ran the iterative scaling algorithm described in Section 9 ("i.s."). (We did not run the sequential-update algorithm of Section 6 since, in preliminary experiments, it seemed to consistently perform worse than the sequential-update algorithm of Section 7).

As noted in Section 9, GIS requires that all features be nonnegative. Given features that do not satisfy this constraint, one can subtract a constant c_j from each feature h_j without changing the model in Eq. (34); thus, one can use a new set of features

$$h'_j(x, y) = h_j(x, y) - c_j$$

where

$$c_j = \min_{i,\ell} h_j(x_i, \ell).$$

The new features define an identical model to that of the old features because the result of the change is that the denominator and numerator in Eq. (34) are both multiplied by the same constant, $\exp\left(-\sum_j \lambda_j c_j\right)$.

A slightly less obvious approach is to choose a feature transformation

$$h'_j(x, y) = h_j(x, y) - c_j(x)$$

where

$$c_j(x) = \min_{\ell} h_j(x, \ell).$$

Like the former approach, this causes h_j to be nonnegative without affecting the model of Eq. (34) (both denominator and numerator of Eq. (34) are now multiplied by $\exp\left(-\sum_j \lambda_j c_j(x)\right)$). Note that, in either case, the constants (c_j or $c_j(x)$) are of no consequence during testing and so can be ignored once training is complete.

In a preliminary version of this paper,[3] we did experiments using only the former approach and found that GIS performed uniformly and considerably worse than any of the other algorithms tested. After the publication of that version, we tried the latter method of making the features nonnegative and obtained much better performance. All of the experiments in the current paper, therefore, use this latter approach.

[3] Appeared in *Proceedings of the Thirteenth Annual Conference on Computational Learning Theory*, 2000.

The results of the first set of experiments are shown in Fig. 4. Each plot of this figure shows the logistic loss on the training set for each of the three methods as a function of the number of iterations. (The loss has been normalized to be 1 when $\boldsymbol{\lambda} = \mathbf{0}$.) Each plot corresponds to a different variation on generating the data, as described above. When there are only a small number of relevant features, the sequential-update algorithms seems to have a clear advantage, but when there are many relevant features, none of the methods seems to be best across-the-board. Of course, all methods eventually converge to the same level of loss.

In the second experiment, we tested how effective the new competitors of AdaBoost are at minimizing the test misclassification error. For this experiment, we used the same parallel- and sequential-update algorithms (denoted "par" and "seq"), and in both cases, we used variants based on exponential loss ("exp") and logistic loss ("log").

Fig. 5 shows a plot of the classification error on a separate test set of 2000 examples. When there are few relevant features, all of the methods overfit on this data, perhaps because of the high-level of noise. With many relevant features, there is not a very large difference in the performance of the exponential and logistic variants of the algorithms, but the parallel-update variants clearly do much better early on; they seem to "go right to the solution," exactly the kind of behavior we would hope for in such an algorithm.

Acknowledgments

Many thanks to Manfred Warmuth for first teaching us about Bregman distances and for many comments on an earlier draft. John Lafferty was also extraordinarily helpful, both in the feedback that he gave us on our results, and in helping us with Theorem 1. Thanks also to Michael Cameron-Jones, Sanjoy Dasgupta, Nigel Duffy, David Helmbold, Raj Iyer and the anonymous reviewers of this paper for helpful discussions and suggestions. Some of this research was done while Yoram Singer was at AT&T Labs.

References

Berger, A. L., Pietra, S. A. D., & Pietra, V. J. D. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, *22*(1), 39–71.

Bregman, L. M. (1967). 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.

Breiman, L. (1997a). Arcing the edge. Tech. rep. 486, Statistics Department, University of California at Berkeley.

Breiman, L. (1997b). Prediction games and arcing classifiers. Tech. rep. 504, Statistics Department, University of California at Berkeley.

Censor, Y., & Lent, A. (1981). An iterative row-action method for interval convex programming. *Journal of Optimization Theory and Applications*, *34*(3), 321–353.

Censor, Y., & Zenios, S. A. (1997). *Parallel Optimization: Theory, Algorithms, and Applications*. Oxford University Press.

Cesa-Bianchi, N., Krogh, A., & Warmuth, M. K. (1994). Bounds on approximate steepest descent for likelihood maximization in exponential families. *IEEE Transactions on Information Theory*, *40*(4), 1215–1220.

Cohen, W. W., & Singer, Y. (1999). A simple, fast, and effective rule learner. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*.

Csiszár, I. (1995). Generalized projections for non-negative functions. *Acta Mathematica Hungarica*, *68*(1-2), 161–185.

Csiszár, I. (1991). Why least squares and maximum entropy? An axiomatic approach to inference for linear inverse problems. *The Annals of Statistics*, *19*(4), 2032–2066.

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

Della Pietra, S., Della Pietra, V., & Lafferty, J. (1997). Inducing features of random fields. *IEEE Transactions Pattern Analysis and Machine Intelligence*, *19*(4), 1–13.

Della Pietra, S., Della Pietra, V., & Lafferty, J. (2001). Duality and auxiliary functions for Bregman distances. Tech. rep. CMU-CS-01-109, School of Computer Science, Carnegie Mellon University.

Domingo, C., & Watanabe, O. (2000). Scaling up a boosting-based learner via adaptive sampling. In *Proceedings of the Fourth Pacific-Asia Conference on Knowledge Discovery and Data Mining*.

Duffy, N., & Helmbold, D. (1999). Potential boosters?. In *Advances in Neural Information Processing Systems 11*.

Freund, Y., & Mason, L. (1999). The alternating decision tree learning algorithm. In *Machine Learning: Proceedings of the Sixteenth International Conference*, pp. 124–133.

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

Friedman, J., Hastie, T., & Tibshirani, R. (2000). Additive logistic regression: A statistical view of boosting. *The Annals of Statistics*, *38*(2), 337–374.

Höffgen, K.-U., & Simon, H.-U. (1992). Robust trainability of single neurons. In *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pp. 428–439.

Kivinen, J., & Warmuth, M. K. (to appear). Relative loss bounds for multidimensional regression problems. *Machine Learning*.

Kivinen, J., & Warmuth, M. K. (1997). Additive versus exponentiated gradient updates for linear prediction. *Information and Computation*, *132*(1), 1–64.

Kivinen, J., & Warmuth, M. K. (1999). Boosting as entropy projection. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, pp. 134–144.

Lafferty, J. (1999). Additive models, boosting and inference for generalized divergences. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, pp. 125–133.

Lafferty, J. D., Pietra, S. D., & Pietra, V. D. (1997). Statistical learning algorithms based on Bregman distances. In *Proceedings of the Canadian Workshop on Information Theory*.

Littlestone, N., Long, P. M., & Warmuth, M. K. (1995). On-line learning of linear functions. *Computational Complexity*, *5*(1), 1–23.

Mason, L., Baxter, J., Bartlett, P., & Frean, M. (1999). Functional gradient techniques for combining hypotheses. In Smola, A. J., Bartlett, P. J., Schölkopf, B., & Schuurmans, D. (Eds.), *Advances in Large Margin Classifiers*. MIT Press.

Rätsch, G., Onoda, T., & Müller, K.-R. (2001). Soft margins for AdaBoost. *Machine Learning*, *42*(3), 287–320.

Schapire, R. E., & Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, *37*(3), 297–336.

Schapire, R. E., & Singer, Y. (2000). BoosTexter: A boosting-based system for text categorization. *Machine Learning*, *39*(2/3), 135–168.

Watanabe, O. (1999). From computational learning theory to discovery science. In *Proceedings of the 26th International Colloquium on Automata, Languages and Programming*, pp. 134–148.