OPERATIONS RESEARCH CENTER

Working Paper

AdaBoost and Forward Stagewise Regression are First-Order Convex Optimization Methods

by

Robert M. Freund
Paul Grigas
Rahul Mazumder

OR 397-14 **June 2013**

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

AdaBoost and Forward Stagewise Regression are First-Order Convex Optimization Methods

Robert M. Freund* Paul Grigas† Rahul Mazumder‡

June 29, 2013

Abstract

Boosting methods are highly popular and effective supervised learning methods which combine weak learners into a single accurate model with good statistical performance. In this paper, we analyze two well-known boosting methods, AdaBoost and Incremental Forward Stagewise Regression (FS_ε), by establishing their precise connections to the Mirror Descent algorithm, which is a first-order method in convex optimization. As a consequence of these connections we obtain novel computational guarantees for these boosting methods. In particular, we characterize convergence bounds of AdaBoost, related to both the margin and log-exponential loss function, for any step-size sequence. Furthermore, this paper presents, for the first time, precise computational complexity results for FS_ε.

1 Introduction

Boosting is a widely popular and successful supervised learning method which combines weak learners in a greedy fashion to deliver accurate statistical models. For an overview of the boosting approach, see, for example, Freund and Schapire [5] and Schapire [18,20]. Though boosting (and in particular AdaBoost [5]) was originally developed in the context of classification problems, it is much more widely applicable [6]. An important application of the boosting methodology in the context of linear regression leads to Incremental Forward Stagewise Regression (FS_ε) [4,7,8]. In this paper, we establish the equivalence of two boosting algorithms, AdaBoost and FS_ε, to specific realizations of the Mirror Descent algorithm, which is a first-order method in convex optimization. Through exact interpretations of these well-known boosting algorithms as specific first-order methods, we leverage our understanding of computational complexity for first-order methods to derive new computational guarantees for these algorithms. Such understanding of algorithmic computational complexity is also helpful from a statistical learning perspective, since it enables one to derive bounds on the number of base models that need to be combined to get a "reasonable" fit to the data.

*MIT Sloan School of Management, 77 Massachusetts Avenue, Cambridge, MA 02139 (mailto: rfreund@mit.edu). This author's research is supported by AFOSR Grant No. FA9550-11-1-0141 and the MIT-Chile-Pontificia Universidad Católica de Chile Seed Fund.

†MIT Operations Research Center, 77 Massachusetts Avenue, Cambridge, MA 02139 (mailto: pgrigas@mit.edu). This author's research has been partially supported through an NSF Graduate Research Fellowship and the MIT-Chile-Pontificia Universidad Católica de Chile Seed Fund.

‡MIT Operations Research Center, 77 Massachusetts Avenue, Cambridge, MA 02139 (mailto: rahulmaz@mit.edu)

Related Work, and Contributions

We briefly outline some of the main developments in the complexity analysis of AdaBoost and FS_ε that appear to be closely related to the topic of this paper.

AdaBoost and Optimization Perspectives: There has been a lot of interesting work connecting AdaBoost and related boosting methods to specific optimization problems and in understanding the computational guarantees of these methods with respect to these optimization problems. In particular, much of the work has focused on two problems: maximizing the margin and minimizing the exponential loss. Mason et al. [9] develop a general framework of boosting methods that correspond to coordinate gradient descent methods to minimize arbitrary loss functions, of which AdaBoost is a particular case with the exponential loss function. For the problem of minimizing the exponential loss, Mukherjee et al. [10] give precise convergence rates for the version of AdaBoost with step-sizes determined by a line-search, see also Telgarsky [22]. Schapire et al. [19] show that the margin is inherently linked to the generalization error of the models produced by AdaBoost, thus it is highly desirable to maximize the margin in order to build predictive models. Several variants of AdaBoost have been developed specifically with this goal in mind, and these methods have been shown to converge to the maximum margin solution at appropriate rates (Ratsch and Warmuth [15], Rudin et al. [17], Shalev-Shwartz and Singer [21]).

Under the assumption that the weak learner oracle returns the optimal base feature (also called weak hypothesis) for any distribution over the training data, we show herein that AdaBoost corresponds exactly to an instance of the Mirror Descent method [2, 11] for the primal/dual paired problem of edge minimization and margin maximization; the primal iterates w^k are distributions over the examples and are attacking the edge minimization problem, and the dual iterates λ^k are nonnegative combinations of classifiers that are attacking the maximum margin problem. In this minmax setting, the Mirror Descent method (and correspondingly AdaBoost) guarantees a certain bound on the duality gap $f(w^k) - p(\lambda^k)$ and hence on the optimality gap as a function of the step-size sequence, and for a simply chosen constant step-size the bound is $\sqrt{\frac{2\ln(m)}{k+1}}$.

In the case of separable data, we use a bound on the duality gap to directly infer a bound on the optimality gap for the problem of maximizing the margin. We show precise rates of convergence for the optimal version of AdaBoost (without any modifications) with respect to the maximum margin problem for any given step-size rule. Our results seem apparently contradictory to Rudin et al. [16], who show that even in the optimal case considered herein (where the weak learner always returns the best feature) AdaBoost may fail to converge to a maximum margin solution. However, in [16] their analysis is limited to the case where AdaBoost uses the originally prescribed step-size $\alpha_k := \frac{1}{2}\ln\left(\frac{1+r_k}{1-r_k}\right)$, where r_k is the edge at iteration k, which can be interpreted as a line-search with respect to the exponential loss (not the margin) in the coordinate direction of the base feature chosen at iteration k (see [9] for a derivation of this). Our interpretation of AdaBoost in fact shows that the algorithm is structurally built to work on the maximum margin problem, and it is only the selection of the step-sizes that can cause convergence for this problem to fail.

In the case of non-separable data, a maximum margin solution is no longer informative; instead, we show that the edge $f(w^k)$ at iteration k is exactly the ℓ_∞ norm of the gradient of the log-exponential loss evaluated at the current classifier, and we infer a bound on this norm through the bound on the duality gap. This bound quantifies the rate at which the classifiers produced by AdaBoost approach the first-order optimality condition for minimizing the log-exponential loss.

Although precise objective function bounds on the optimality gap with respect to the exponential loss were given in [10], their analysis is limited to the case of step-sizes determined by a line-search, as mentioned above. The step-sizes suggested by our Mirror Descent interpretation are quite different from those determined by a line-search, and furthermore although our bounds are specific to either the separable or non-separable case, the step-sizes we suggest do not depend on which case applies to a particular data set.

Forward Stagewise Regression and Optimization Perspectives: The Incremental Forward Stagewise algorithm [4,7,8] (FS_ε) with shrinkage parameter ε is a boosting algorithm for the linear regression problem that iteratively updates (by a small amount ε) the coefficient of the variable most correlated with the current residuals. A principal reason behind why FS_ε is attractive from a statistical viewpoint is because of its ability to deliver *regularized* solutions (24) by controlling the number of iterations k along with the shrinkage parameter ε with proper bias-variance tradeoff [8]. The choice of the step-size plays an important role in the algorithm and has a bearing on the statistical properties of the type of solutions produced. For example, a step-size chosen by exact line-search on the least-squares loss function leads to the well known Forward Stagewise Algorithm—a greedy version of best subset selection [8]. *Infinitesimal* Incremental Forward Stagewise Regression (FS_0, i.e., the limit of FS_ε as $\varepsilon \to 0+$) under some additional conditions on the data leads to a coefficient profile that is exactly the same as the LASSO solution path [7, 8]. It is thus natural to ask what criterion might the FS_ε algorithm optimize?, and is it possible to have computational complexity guarantees for FS_ε — and that can accommodate a flexible choice of steps-sizes? To the best of our knowledge, a simple and complete answer to the above questions are heretofore unknown. In this paper, we answer these questions by showing that FS_ε is working towards minimizing the maximum correlation between the residuals and the predictors, which can also be interpreted as the ℓ_∞ norm of the gradient of the least-squares loss function. Our interpretation yields a precise bound on this quantity for any choice of the shrinkage parameter ε, in addition to the regularization/sparsity properties (24).

1.1 Notation

For a vector $x \in \mathbb{R}^n$, x_i denotes the i^{th} coordinate; we use superscripts to index vectors in a sequence $\{x^k\}$. Let e_j denote the j^{th} unit vector in $\mathbb{R}^n$, $e = (1, \ldots, 1)$, and $\Delta_n = \{x \in \mathbb{R}^n : e^T x = 1, x \geq 0\}$ is the $(n-1)$-dimensional unit simplex. Let $\|\cdot\|_q$ denote the q-norm for $q \in [1, \infty]$ with unit ball B_q, and let $\|v\|_0$ denote the number of non-zero coefficients of the vector v. For $A \in \mathbb{R}^{m \times n}$, let $\|A\|_{q_1,q_2} := \max_{x:\|x\|_{q_1} \leq 1} \|Ax\|_{q_2}$ be the operator norm. For a given norm $\|\cdot\|$ on $\mathbb{R}^n$, $\|\cdot\|_*$ denotes the dual norm defined by $\|s\|_* = \max_{x:\|x\| \leq 1} s^T x$. Let $\partial f(\cdot)$ denote the subdifferential operator of a convex function $f(\cdot)$. The notation "$\tilde{v} \leftarrow \arg\max_{v \in S}\{f(v)\}$" denotes assigning $\tilde{v}$ to be any optimal solution of the problem $\max_{v \in S}\{f(v)\}$. For a convex set P let $\Pi_P(\cdot)$ denote the Euclidean projection operator onto P, namely $\Pi_P(\bar{x}) := \arg\min_{x \in P} \|x - \bar{x}\|_2$.

2 Subgradient and Generalized Mirror Descent Methods: A Brief Review

Suppose we are interested in solving the following optimization problem:

$$\text{(Primal):} \quad \min_{x \in P} \; f(x) \; , \tag{1}$$

where $P \subseteq \mathbb{R}^n$ is a closed convex set, $\mathbb{R}^n$ is considered with the given norm $\|\cdot\|$, and $f(\cdot) : P \to \mathbb{R}$ is a (possibly non-smooth) convex function. Recall that g is a subgradient of $f(\cdot)$ at x if $f(y) \geq f(x) + g^T(y - x)$ for all $y \in P$, and we denote the set of subgradients of $f(\cdot)$ at x by $\partial f(x)$. We assume with no loss of generality that $\partial f(x) \neq \emptyset$ for all $x \in P$. We presume that computation of a subgradient at $x \in P$ is not a burdensome task. Furthermore, we assume that $f(\cdot)$ has Lipschitz function values with Lipschitz constant L_f, i.e., we have $|f(x) - f(y)| \leq L_f \|x - y\|$ for all $x, y \in P$.

We are primarily interested in the case where $f(\cdot)$ is conveyed with minmax structure, namely:

$$f(x) := \max_{\lambda \in Q} \; \phi(x, \lambda) \; , \tag{2}$$

where $Q \subseteq \mathbb{R}^m$ is a convex and compact set and $\phi(\cdot, \cdot)$ is a differentiable function that is convex in the first argument and concave in the second argument. In the case when P is bounded, we define a dual function $p(\cdot) : Q \to \mathbb{R}$ by

$$p(\lambda) := \min_{x \in P} \; \phi(x, \lambda) \; , \tag{3}$$

for which we may be interested in solving the dual problem:

$$\text{(Dual):} \quad \max_{\lambda \in Q} \; p(\lambda) \; . \tag{4}$$

Let f^* denote the optimal value of (1). When P is bounded let p^* denote the optimal value of (4), and the compactness of P and Q ensure that weak and strong duality hold: $p(\lambda) \leq p^* = f^* \leq f(x)$ for all $\lambda \in Q$ and $x \in P$. The choice to call (1) the primal and (4) the dual is of course arbitrary, but this choice is relevant since the algorithms reviewed herein are *not* symmetric in their treatment of the primal and dual computations.

The classical subgradient descent method for solving (1) determines the next iterate by taking a step α in the negative direction of a subgradient at the current point, and then projecting the resulting point back onto the set P. If x^k is the current iterate, subgradient descent proceeds by computing a subgradient $g^k \in \partial f(x^k)$, and determines the next iterate as $x^{k+1} \leftarrow \Pi_P(x^k - \alpha_k g^k)$, where α_k is the step-length, and $\Pi_P(\cdot)$ is the Euclidean projection onto the set P.

Note that in the case when $f(\cdot)$ has minmax structure (2), the ability to compute subgradients depends very much on the ability to solve the subproblem in the definition (2). Indeed,

$$\text{if} \;\; \tilde{\lambda}^k \in \arg\max_{\lambda \in Q} \phi(x^k, \lambda) \;\; , \;\; \text{then} \;\; g^k \leftarrow \nabla_x \phi(x^k, \tilde{\lambda}^k) \in \partial f(x^k) \; , \tag{5}$$

that is, g^k is a subgradient of $f(\cdot)$ at x^k. This fact is very easy to derive, and is a special case of the more general result known as Danskin's Theorem, see [3].

In consideration of the computation of the subgradient (5) for problems with minmax structure (2), the formal statement of the subgradient descent method is presented in Algorithm 1.

Method 1 Subgradient Descent Method (for problems with minmax structure)

Initialize at $x^0 \in P$, $k \leftarrow 0$

At iteration k:

1. Compute:

$$\tilde{\lambda}^k \leftarrow \arg\max_{\lambda \in Q} \ \phi(x^k, \lambda)$$

$$g^k \leftarrow \nabla_x \phi(x^k, \tilde{\lambda}^k)$$

2. Choose $\alpha_k \geq 0$ and set:

$$x^{k+1} \leftarrow \Pi_P(x^k - \alpha_k g^k)$$

The Mirror Descent method [2, 11] is a generalization of the subgradient descent method. The Mirror Descent method requires the selection of a differentiable "1-strongly convex" function $d(\cdot) : P \to \mathbb{R}$ which is defined to be a function with the following (strong) convexity property:

$$d(x) \geq d(y) + \nabla d(y)^T (x - y) + \frac{1}{2}\|x - y\|^2 \text{ for all } x, y \in P \ .$$

The function $d(\cdot)$ is typically called the "prox function." The given prox function $d(\cdot)$ is also used to define a distance function:

$$D(x, y) := d(x) - d(y) - \nabla d(y)^T (x - y) \geq \frac{1}{2}\|x - y\|^2 \text{ for all } x, y \in P \ . \tag{6}$$

One can think of $D(x, y)$ as a not-necessarily-symmetric generalization of a distance metric (induced by a norm), in that $D(x, y) \geq \frac{1}{2}\|x - y\|^2 \geq 0$, $D(x, y) = 0$ if and only if $x = y$, but it is not generally true (nor is it useful) that $D(x, y) = D(y, x)$. $D(x, y)$ is called the Bregman function or the Bregman distance. With these objects in place, the Mirror Descent (proximal subgradient) method for solving (1) is presented in Algorithm 2.

The sequence $\{\lambda^k\}$ constructed in the last line of Step (2.) of Mirror Descent plays no role in the actual dynamics of Algorithm 2 and so could be ignored; however λ^k is a feasible solution to the dual problem (4) and we will see that the sequence $\{\lambda^k\}$ has precise computational guarantees with respect to problem (4). The construction of x^{k+1} in Step (2.) of Mirror Descent involves the solution of an optimization subproblem; the prox function $d(\cdot)$ should be chosen so that this subproblem can be easily solved, i.e., in closed form or with a very efficient algorithm.

Note that the subgradient descent method described in Algorithm 1 is a special case of Mirror Descent using the "Euclidean" prox function $d(x) := \frac{1}{2}\|x\|_2^2$. With this choice of prox function, Step (2.) of Algorithm 2 becomes:

$$x^{k+1} \leftarrow \arg\min_{x \in P} \left\{ \left(\alpha_k g^k - x^k\right)^T x + \frac{1}{2} x^T x \right\} = \Pi_P(x^k - \alpha_k g^k) \ ,$$

(since $D(x, x^k) = (-x^k)^T x + \frac{1}{2} x^T x + \frac{1}{2}\|x^k\|_2^2$), and is precisely the subgradient descent method with step-size sequence $\{\alpha_k\}$. Indeed, the sequence $\{\alpha_k\}$ in the Mirror Descent method is called the

Method 2 Mirror Descent Method (applied to problems with minmax structure)

Initialize at $x^0 \in P$, $\lambda^0 = 0, k = 0$

At iteration k:

1. Compute:

$$\tilde{\lambda}^k \leftarrow \arg\max_{\lambda \in Q} \ \phi(x^k, \lambda)$$

$$g^k \leftarrow \nabla_x \phi(x^k, \tilde{\lambda}^k)$$

2. Choose $\alpha_k \geq 0$ and set:

$$x^{k+1} \leftarrow \arg\min_{x \in P} \left\{ \alpha_k (g^k)^T x + D(x, x^k) \right\}$$

$$\lambda^{k+1} \leftarrow \frac{\sum_{i=0}^{k} \alpha_i \tilde{\lambda}^i}{\sum_{i=0}^{k} \alpha_i}$$

"step-size" sequence in light of the analogy to subgradient descent. Below we present an example of a version of Mirror Descent with a prox function that is not Euclidean, which will be useful in the analysis of the algorithm AdaBoost.

Example 2.1. **Multiplicative Weight Updates for Optimization on the Standard Simplex in $\mathbb{R}^n$**
Consider optimization of $f(x)$ on $P = \Delta_n := \{x \in \mathbb{R}^n : e^T x = 1, x \geq 0\}$, the standard simplex in $\mathbb{R}^n$, and let $d(x) = e(x) := \sum_{i=1}^n x_i \ln(x_i) + \ln(n)$ be the entropy function. It is well-known that $e(\cdot)$ is a 1-strongly convex function on Δ_n with respect to the ℓ_1 norm, see for example [13] for a short proof. Given any $c \in \mathbb{R}^n$, it is straightforward to verify that the optimal solution $\bar{x}$ of a problem of format $\min_{x \in P} \left\{ c^T x + d(x) \right\}$ is given by:

$$\bar{x}_i = \frac{\exp(-c_i)}{\sum_{l=1}^n \exp(-c_l)} \quad i = 1, \ldots, n \ . \tag{7}$$

Using the entropy prox function, it follows that for each $i \in \{1, \ldots, n\}$, the update of x^k in Step (2.) of Algorithm 2 assigns:

$$x_i^{k+1} \propto \exp(-(\alpha_k g^k - \nabla e(x^k))_i) = \exp(1 + \ln(x_i^k) - \alpha_k g_i^k) \propto x_i^k \cdot \exp(-\alpha_k g_i^k) \ ,$$

which is an instance of the multiplicative weights update rule [1].

We now state two well-known complexity bounds for the Mirror Descent method (Algorithm 2), see for example [2]. In the general case we present a bound on the optimality gap of the sequence $\{x^k\}$ for the primal problem (1) that applies for any step-size sequence $\{\alpha_k\}$, and in the case when P is compact we present a similar bound on the duality gap of the sequences $\{x^k\}$ and $\{\lambda^k\}$. Both bounds can be specified to $O\left(\frac{1}{\sqrt{k}}\right)$ rates for particularly chosen step-sizes.

Theorem 2.1. (Complexity of Mirror Descent [2, 12, 14]) *Let $\{x^k\}$ and $\{\lambda^k\}$ be generated according to the Mirror Descent method (Algorithm 2). Then for each $k \geq 0$ and for any $x \in P$, the following inequality holds:*

$$\min_{i\in\{0,\ldots,k\}} f(x^i) - f(x) \quad \leq \quad \frac{D(x,x^0) + \frac{1}{2}L_f^2 \sum_{i=0}^k \alpha_i^2}{\sum_{i=0}^k \alpha_i} \ . \tag{8}$$

If P is compact and $\bar{D} \geq \max_{x\in P} D(x,x^0)$, then for each $k \geq 0$ the following inequality holds:

$$\min_{i\in\{0,\ldots,k\}} f(x^i) - p(\lambda^{k+1}) \quad \leq \quad \frac{\bar{D} + \frac{1}{2}L_f^2 \sum_{i=0}^k \alpha_i^2}{\sum_{i=0}^k \alpha_i} \ . \tag{9}$$

These bounds are quite general; one can deduce specific bounds, for example, by specifying a step-size sequence $\{\alpha^k\}$, a prox function $d(\cdot)$, a value of x in (8) such as $x = x^*$, etc., see Propositions 2.1 and 2.2 where these specifications are illustrated in the case when P is compact, for example.

Proposition 2.1. *Suppose we* a priori *fix the number of iterations k of Algorithm 2 and use a constant step-size sequence:*

$$\alpha_i = \bar{\alpha} = \frac{1}{L_f}\sqrt{\frac{2\bar{D}}{k+1}} \qquad \text{for} \quad i = 0,\ldots,k \ . \tag{10}$$

Then

$$\min_{i\in\{0,\ldots,k\}} f(x^i) - p(\lambda^{k+1}) \quad \leq \quad L_f\sqrt{\frac{2\bar{D}}{k+1}} \ . \tag{11}$$

Proof. This follows immediately from (9) by substituting in (10) and rearranging terms. □

Indeed, the bound (11) is in fact the best possible bound for a generic subgradient method, see [11].

Proposition 2.2. *Suppose we use the dynamic step-size sequence:*

$$\alpha_i := \frac{1}{L_f}\sqrt{\frac{2\bar{D}}{i+1}} \qquad \text{for} \quad i \geq 0 \ . \tag{12}$$

Then after k iterations the following holds:

$$\min_{i\in\{0,\ldots,k\}} f(x^i) - p(\lambda^{k+1}) \quad \leq \quad \frac{L_f\sqrt{\frac{1}{2}\bar{D}}\,(2+\ln(k+1))}{2(\sqrt{k+2}-1)} \ = \ O\left(\frac{L_f\sqrt{\bar{D}}\ln(k)}{\sqrt{k}}\right) \ . \tag{13}$$

Proof. Substituting (12) in (9) and rearranging yields:

$$\min_{i\in\{0,\ldots,k\}} f(x^i) - p(\lambda^{k+1}) \quad \leq \quad \frac{L_f\sqrt{\frac{1}{2}\bar{D}}\left(1+\sum_{i=0}^k \frac{1}{i+1}\right)}{\sum_{i=0}^k \frac{1}{\sqrt{i+1}}} \ .$$

The proof is completed by using the integral bounds

$$1 + \sum_{i=0}^{k} \frac{1}{i+1} \leq 2 + \int_1^{k+1} \frac{1}{t} dt = 2 + \ln(k+1) \ ,$$

and

$$\sum_{i=0}^{k} \frac{1}{\sqrt{i+1}} = \sum_{i=1}^{k+1} \frac{1}{\sqrt{i}} \geq \int_1^{k+2} \frac{1}{\sqrt{t}} dt = 2\sqrt{k+2} - 2 \ .$$

□

Finally, consider the subgradient descent method (Algorithm 1), which is Mirror Descent using $d(x) = \frac{1}{2}\|x\|_2^2$ in the case when the optimal value f^* of (1) is known. Suppose that the step-sizes are given by $\alpha_k := \frac{f(x^k) - f^*}{\|g^k\|_2^2}$, then it is shown in Polyak [14] that for any optimal solution x^* of (1) it holds that:

$$\min_{i \in \{0,\ldots,k\}} f(x^i) - f^* \quad \leq \quad \frac{L_f \|x^0 - x^*\|_2}{\sqrt{k+1}} \ . \tag{14}$$

3 AdaBoost as Mirror Descent

We are given a set of base classifiers (also called weak hypotheses) $\mathcal{H} = \{h_1, \ldots, h_n\}$ where each $h_j : \mathcal{X} \to \{-1, 1\}$, and we are given training data (examples) $(x_1, y_1), \ldots, (x_m, y_m)$ where each $x_i \in \mathcal{X}$ ($\mathcal{X}$ is some measurement space) and each $y_i \in \{-1, +1\}$.[1] We have access to a weak learner $\mathcal{W}(\cdot) : \Delta_m \to \{1, \ldots, n\}$ that, for any distribution w on the examples ($w \in \Delta_m$), returns an index j^* of a base classifier h_{j^*} in $\mathcal{H}$ that does best on the weighted example determined by w. That is, the weak learner $\mathcal{W}(w)$ computes $j^* \in \arg\max_{j \in \{1,\ldots,n\}} \sum_{i=1}^m w_i y_i h_j(x_i)$ and we write " $j^* \in \mathcal{W}(w)$ " in a slight abuse of notation. Even though n may be extremely large, we assume that it is easy to compute an index $j^* \in \mathcal{W}(w)$ for any $w \in \Delta_m$. Algorithm 3 is the algorithm AdaBoost, which constructs a sequence of distributions $\{w^k\}$ and a sequence $\{H_k\}$ of nonnegative combinations of base classifiers with the intent of designing a classifier $\text{sign}(H_k)$ that performs significantly better than any base classifier in $\mathcal{H}$.

Algorithm 3 AdaBoost

Initialize at $w^0 = (1/m, \ldots, 1/m), H_0 = 0, k = 0$

At iteration k:

1. Compute $j_k \in \mathcal{W}(w^k)$
2. Choose $\alpha_k \geq 0$ and set:

$H_{k+1} \leftarrow H_k + \alpha_k h_{j_k}$

$w_i^{k+1} \leftarrow w_i^k \exp(-\alpha_k y_i h_{j_k}(x_i)) \ \ i = 1, \ldots, m$, and re-normalize w^{k+1} so that $e^T w^{k+1} = 1$

[1] Actually our results also hold for the more general confidence-rated classification setting, where $h_j : \mathcal{X} \to [-1, 1]$ and $y_i \in [-1, 1]$.

Notice that AdaBoost maintains a sequence of classifiers $\{H_k\}$ that are (nonnegative) linear combinations of base classifiers in $\mathcal{H}$. Strictly speaking, a linear combination $H = \sum_{j=1}^n \lambda_j h_j$ of base classifiers in $\mathcal{H}$ is a function from $\mathcal{X}$ into the reals, and the classifier determined by H is $\text{sign}(H)$; however, for simplicity we will refer to the linear combination H as a classifier, and we say that the coefficient vector $\lambda \in \mathbb{R}^n$ determines the classifier H.

3.1 AdaBoost is a Specific Case of Mirror Descent

Here we show that AdaBoost corresponds to a particular instance of the Mirror Descent method (Algorithm 2) applied to the particular primal problem of minimizing the *edge* in the space of "primal" variables $w \in \Delta_m$ which are distributions over the training data; and through duality, maximizing the *margin* in the space of "dual" variables of normalized classifiers represented by vectors $\lambda \in \mathbb{R}^n$ of coefficients which determine classifiers $\sum_{j=1}^n \lambda_j h_j$. We also show that the edge of w^k is exactly the ℓ_∞ norm of the gradient of the log-exponential loss function. Utilizing the computational complexity results for Mirror Descent (Theorem 2.1), we then establish guarantees on the duality gap for these duality paired problems. When the data are separable, these guarantees imply that the sequence of classifiers $\{H_k\}$ constructed in AdaBoost are in fact working on the problem of maximizing the *margin*, with specific computational guarantees thereof for any step-size sequence $\{\alpha_k\}$. When the data is not separable, these guarantees imply that the classifiers $\{H_k\}$ are in fact working on the problem of driving the ℓ_∞ norm of the gradient of the log-exponential loss function to zero, with specific computational guarantees thereof for any step-size sequence $\{\alpha_k\}$. Let us see how this works out.

For convenience define the feature matrix $A \in \mathbb{R}^{m \times n}$ componentwise by $A_{ij} := y_i h_j(x_i)$, and let A_j denote the jth column of A, and define $\phi(w, \lambda) = w^T A \lambda$ where we use w instead of x to represent the primal variable. For any distribution $w \in \Delta_m$, $w^T A_j$ is the *edge* of classifier h_j with respect to w, and

$$f(w) := \max_{j \in \{1,\ldots,n\}} w^T A_j = \max_{\lambda \in \Delta_n} w^T A \lambda = \max_{\lambda \in \Delta_n} \phi(w, \lambda) \tag{15}$$

is the maximum edge over all base classifiers, and we call $f(w)$ the edge with respect to w. The optimization problem of minimizing the edge over all distributions w is:

$$\text{(Primal):} \quad \min_{w \in \Delta_m} f(w) \ . \tag{16}$$

Here (15) and (16) are precisely in the format of (2) and (1) with $P = \Delta_m$ and $Q = \Delta_n$. We can construct the dual of the edge minimization problem following (3) and (4), whereby we see that the dual function is:

$$p(\lambda) := \min_{w \in \Delta_m} \phi(w, \lambda) = \min_{w \in \Delta_m} w^T A \lambda = \min_{i \in \{1,\ldots,m\}} (A\lambda)_i \ , \tag{17}$$

and the dual problem is:

$$\text{(Dual):} \quad \max_{\lambda \in \Delta_n} p(\lambda) \ . \tag{18}$$

The *margin* achieved by the λ on example i is $(A\lambda)_i$, whereby $p(\lambda)$ is the least margin achieved by λ over all examples, and is simply referred to as the margin of λ. Because $p(\lambda)$ is positively homogeneous ($p(\beta\lambda) = \beta p(\lambda)$ for $\beta \geq 0$), it makes sense to normalize λ when measuring the margin,

which we do by re-scaling λ so that $\lambda \in \Delta_n$. Therefore the dual problem is that of maximizing the margin over all normalized nonnegative classifiers. Note also that it is without loss of generality that we assume $\lambda \geq 0$ since for any base classifier $h_j \in \mathcal{H}$ we may add the classifier $-h_j$ to the set $\mathcal{H}$ if necessary. Consider the classifier H_k constructed in Step (2.) of AdaBoost. It follows inductively that $H_k = \sum_{i=0}^{k-1} \alpha_i h_{j_i}$, and we define the normalization of H_k as:

$$\bar{H}_k := \frac{H_k}{\sum_{i=0}^{k-1} \alpha_i} = \frac{\sum_{i=0}^{k-1} \alpha_i h_{j_i}}{\sum_{i=0}^{k-1} \alpha_i} \ . \tag{19}$$

In addition to the margin function $p(\lambda)$, it will be useful to look at log-exponential loss function $L(\cdot) : \mathbb{R}^n \to \mathbb{R}$ which is defined as:

$$L(\lambda) = \log \left(\frac{1}{m} \sum_{i=1}^{m} \exp\left(-(A\lambda)_i\right) \right) \ . \tag{20}$$

It is well-known that $L(\cdot)$ and $p(\cdot)$ are related by: $-p(\lambda) - \ln(m) \leq L(\lambda) \leq -p(\lambda)$ for any λ.

We establish the following equivalence result.

Theorem 3.1. *The sequence of weight vectors $\{w^k\}$ in AdaBoost arise as the sequence of primal variables in Mirror Descent applied to the primal problem* (16)*, using the entropy prox function $d(w) := e(w) = \sum_{i=1}^{m} w_i \ln(w_i) + \ln(m)$, with step-size sequence $\{\alpha_k\}$ and initialized at $w^0 = (1/m, \ldots, 1/m)$. Furthermore, the sequence of normalized classifiers $\{\bar{H}_k\}$ produced by AdaBoost arise as the sequence of dual variables $\{\lambda^k\}$ in Mirror Descent, and the margin of the classifier $\bar{H}_k$ is $p(\lambda^k)$.*

Proof. By definition of the weak learner and (15) combined with (5), we have for any $w \in \Delta_m$

$$j^* \in \mathcal{W}(w) \Longleftrightarrow j^* \in \operatorname*{arg\,max}_{j \in \{1,\ldots,n\}} w^T A_j \Longleftrightarrow e_{j^*} \in \operatorname*{arg\,max}_{\lambda \in \Delta_n} w^T A\lambda \Longleftrightarrow A_{j^*} \in \partial f(w) \ ,$$

whereby Step (1.) of AdaBoost is identifying a vector $g^k := A_{j_k} \in \partial f(w^k)$. Moreover, since $g_i^k = y_i h_{j_k}(x_i) = A_{i,j_k}$, the construction of w^{k+1} in Step (2.) of AdaBoost is exactly setting $w^{k+1} \leftarrow \arg\min_{w \in \Delta_m} \left\{ \alpha_k (g^k)^T x + D(w, w^k) \right\}$ (where $D(\cdot,\cdot)$ is the Bregman distance function arising from the entropy function), as discussed in Example 2.1. Therefore the sequence $\{w^k\}$ is a sequence of primal variables in Mirror Descent with the entropy prox function. Also notice from Step (1.) of AdaBoost and the output of the weak learner $\mathcal{W}(w^k)$ that $e_{j_k} \in \operatorname*{arg\,max}_{\lambda \in \Delta_n} (w^k)^T A\lambda$, which gives the correspondence $\tilde{\lambda}^k = e_{j_k}$ at Step (1.) of Mirror Descent. Let $\{\lambda^k\}$ denote the corresponding sequence of dual variables defined in Step (2.) of Mirror Descent; it therefore follows that:

$$\lambda^k := \frac{\sum_{i=0}^{k-1} \alpha_i \tilde{\lambda}^i}{\sum_{i=0}^{k-1} \alpha_i} = \frac{\sum_{i=0}^{k-1} \alpha_i e_{j_i}}{\sum_{i=0}^{k-1} \alpha_i} \ ,$$

whereby $\bar{H}_k$ defined in (19) is precisely the classifier determined by λ^k, and it follows that the margin of $\bar{H}_k$ is $p(\lambda^k)$. □

Let $\{\hat{\lambda}^k\}$ denote the sequence of coefficient vectors of the un-normalized classifiers $\{H_k\}$ produced by AdaBoost, where $\hat{\lambda}^k = \sum_{i=0}^{k-1} \alpha_i e_{j_i}$. We also have the following relationship concerning the norm of the gradient of log-exponential loss function.

Lemma 3.1. *For every iteration $k \geq 0$ of AdaBoost, the edge $f(w^k)$ and the un-normalized classifier H_k with coefficient vector $\hat{\lambda}^k$ satisfy:*

$$f(w^k) = \|\nabla L(\hat{\lambda}^k)\|_\infty \ .$$

Proof. By our assumption that the set of base classifiers $\mathcal{H}$ is closed under negation, we have for any w that $f(w) = \max_{\lambda \in \Delta_n} w^T A\lambda = \max_{\lambda : \|\lambda\|_1 \leq 1} w^T A\lambda = \|A^T w\|_\infty$. It remains to show that $-A^T w^k = \nabla L(\hat{\lambda}^k)$. To do so, first note that

$$\nabla L(\hat{\lambda}^k)_j = \frac{\sum_{i=1}^m -A_{ij} \exp(-(A\hat{\lambda}^k)_i)}{\sum_{\ell=1}^m \exp(-(A\hat{\lambda}^k)_\ell)} \ .$$

Thus, defining a vector $\hat{w}^k \in \Delta_m$ by

$$\hat{w}_i^k := \frac{\exp(-(A\hat{\lambda}^k)_i)}{\sum_{\ell=1}^m \exp(-(A\hat{\lambda}^k)_\ell)} \ ,$$

then we have that $\nabla L(\hat{\lambda}^k) = -A^T \hat{w}^k$. Clearly, we have $\hat{w}^0 = w^0$. By way of induction, supposing that $\hat{w}^k = w^k$, then by the update in step (2.) of AdaBoost we have that

$$\begin{aligned} w_i^{k+1} &\propto w_i^k \exp(-\alpha_k A_{ij_k}) \\ &= \hat{w}_i^k \exp(-\alpha_k A_{ij_k}) \\ &\propto \exp(-(A\hat{\lambda}^k)_i - \alpha_k A_{ij_k}) \\ &= \exp(-(A(\hat{\lambda}^k + \alpha_k e_{j_k}))_i) \\ &= \exp(-(A\hat{\lambda}^{k+1})_i) \ . \end{aligned}$$

Since both w^{k+1} and $\hat{w}^{k+1}$ are normalized, we have that $w^{k+1} = \hat{w}^{k+1}$. Therefore, we have that $-A^T w^k = \nabla L(\hat{\lambda}^k)$ for all $k \geq 0$, and in particular $\|A^T w^k\|_\infty = \|\nabla L(\hat{\lambda}^k)\|_\infty$. □

The equivalences given by Theorem 3.1 and Lemma 3.1 imply computational complexity results for AdaBoost for both the margin $p(\lambda)$ and the gradient of the log-exponential loss function, for a variety of step-size rules via Theorem 2.1, as follows.

Theorem 3.2. (Complexity of AdaBoost) *For all $k \geq 1$, the sequence of classifiers $\{H_k\}$, with coefficient vectors $\{\hat{\lambda}^k\}$, and their normalizations $\{\bar{H}_k\}$, with coefficient vectors $\{\lambda^k\}$, produced by AdaBoost satisfy:*

$$\min_{i \in \{0,\ldots,k-1\}} \|\nabla L(\hat{\lambda}^i)\|_\infty - p(\lambda^k) \leq \frac{\ln(m) + \frac{1}{2}\sum_{i=0}^{k-1} \alpha_i^2}{\sum_{i=0}^{k-1} \alpha_i} \ . \tag{21}$$

If we decide a priori to run AdaBoost for $k \geq 1$ iterations and use a constant step-size $\alpha_i := \sqrt{\frac{2\ln(m)}{k}}$ for all $i = 0, \ldots, k-1$, then:

$$\min_{i \in \{0,\ldots,k-1\}} \|\nabla L(\hat{\lambda}^i)\|_\infty - p(\lambda^k) \leq \sqrt{\frac{2\ln(m)}{k}} \ . \tag{22}$$

If instead we use the dynamic step-size $\alpha_i := \sqrt{\frac{2\ln(m)}{i+1}}$*, then:*

$$\min_{i\in\{0,\ldots,k-1\}} \|\nabla L(\hat{\lambda}^i)\|_\infty - p(\lambda^k) \leq \frac{\sqrt{\frac{\ln(m)}{2}}\,[2+\ln(k)]}{2(\sqrt{k+1}-1)}\ . \tag{23}$$

Proof. By weak duality and invoking Lemma 3.1 we have $p(\lambda^k) \leq \rho^* \leq \min_{i\in\{0,\ldots,k-1\}} f(w^i) = \min_{i\in\{0,\ldots,k-1\}} \|\nabla L(\hat{\lambda}^i)\|_\infty$. By Lemma A.1, we have that $f(\cdot)$ has Lipschitz function values with Lipschitz constant $L_f = \|A\|_{1,\infty} = 1$, and by Lemma A.2, we have that $\max_{w\in\Delta_m} D(w, w^0) = \ln(m)$. Thus (21) follows directly from (9) in Theorem 2.1. The bounds (22) and (23) follow from (11) and (13), respectively. □

Let us now discuss these results. Let $\rho^* := \max_{\lambda\in\Delta_n} p(\lambda)$ be the maximum margin over all normalized classifiers. Since we are assuming that the set of base classifiers $\mathcal{H}$ is closed under negation, it is always the case that $\rho^* \geq 0$. When $\rho^* > 0$, there is a vector $\lambda^* \in \Delta_n$ with $A\lambda^* > 0$, and thus the classifier determined by λ^* separates the data. In this separable case, it is both intuitively and theoretically desirable [19] to find a classifier with high margin, i.e., one that is close to the optimal value ρ^*. For any $k \geq 1$, by weak duality, we have that $\rho^* \leq \min_{i\in\{0,\ldots,k-1\}} \|\nabla L(\hat{\lambda}^i)\|_\infty$, whereby the bounds in (21), (22), and (23) hold for $\rho^* - p(\lambda^k)$, and thus provide exact computational guarantees that bound the optimality gap $\rho^* - p(\lambda^k)$ of the classifier $\bar{H}_k$ produced by AdaBoost.

When $\rho^* = 0$, then the data is not separable, and achieving the maximum margin is trivial; for example the classifier $\frac{1}{2}h_1 + \frac{1}{2}(-h_1)$ achieves the optimal margin. In this case the log-exponential loss function $L(\cdot)$ is a metric of algorithm performance. For any $k \geq 1$, by weak duality, we have that $0 = \rho^* \geq p(\lambda^k)$, whereby the bounds in (21), (22), and (23) hold for $\min_{i\in\{0,\ldots,k-1\}} \|\nabla L(\hat{\lambda}^i)\|_\infty$ - 0 and hence provide exact computational complexity bounds for the ℓ_∞ norm of the gradient of $L(\cdot)$ thereby guaranteeing the extent to which the classifiers H_k (equivalently $\hat{\lambda}^k$) produced by AdaBoost satisfy the first-order optimality condition for minimizing $L(\cdot)$.

4 FS_ε as Subgradient Descent

Here we consider the linear regression model $\mathbf{y} = \mathbf{X}\beta + \mathbf{e}$, with given response vector $\mathbf{y} \in \mathbb{R}^n$, given model matrix $\mathbf{X} \in \mathbb{R}^{n\times p}$, regression coefficients $\beta \in \mathbb{R}^p$ and errors $\mathbf{e} \in \mathbb{R}^n$. In the high-dimensional statistical regime, especially with $p \gg n$, a sparse linear model with few non-zero coefficients is often desirable. In this context, ℓ_1-penalized regression, i.e., LASSO [23], is often used to perform variable selection and shrinkage in the coefficients and is known to yield models with good predictive performance. The Incremental Forward Stagewise algorithm (FS_ε) [7, 8] with shrinkage factor ε is a type of boosting algorithm for the linear regression problem. FS_ε generates a coefficient profile[2] by repeatedly updating (by a small amount ε) the coefficient of the variable most correlated with the current residuals. A complete description of FS_ε is presented in Algorithm 4.

[2] A coefficient profile is a path of coefficients $\{\beta(\alpha)\}_{\alpha\in\boldsymbol{\alpha}}$ where α parameterizes the path. In the context of FS_ε, α indexes the ℓ_1 arc-length of the coefficients.

Algorithm 4 Incremental Forward Stagewise algorithm (FS_ε)

Initialize at $r^0 = \mathbf{y}$, $\beta^0 = 0, k = 0$

At iteration k:

1. Compute:

$$j_k \in \arg\max_{j \in \{1,\dots,p\}} |(r^k)^T \mathbf{X}_j|$$

2. Set:

$$r^{k+1} \leftarrow r^k - \varepsilon \ \text{sgn}((r^k)^T \mathbf{X}_{j_k}) \mathbf{X}_{j_k}$$

$$\beta_{j_k}^{k+1} \leftarrow \beta_{j_k}^k + \varepsilon \ \text{sgn}((r^k)^T \mathbf{X}_{j_k})$$

$$\beta_j^{k+1} \leftarrow \beta_j^k \ , j \neq j_k$$

As a consequence of the update scheme in Step (2.) of Algorithm 4, FS_ε has the following sparsity property:

$$\|\beta^k\|_1 \leq k\varepsilon \quad \text{and} \quad \|\beta^k\|_0 \leq k \ . \tag{24}$$

Different choices of ε lead to different instances; for example a choice of $\varepsilon_k := |(r^k)^T \mathbf{X}_{j_k}|$ yields the Forward Stagewise algorithm (FS) [8], which is a greedy version of best-subset selection.

4.1 FS_ε is a Specific Case of Subgradient Descent

We now show that FS_ε is in fact an instance of the subgradient descent method (algorithm 1) to minimize the largest correlation between the residuals and the predictors, over the space of residuals. Indeed, consider the convex optimization problem:

$$\min_{r \in P_{\text{res}}} \quad f(r) := \|\mathbf{X}^T r\|_\infty \tag{25}$$

where $P_{\text{res}} := \{r \in \mathbb{R}^n : r = \mathbf{y} - \mathbf{X}\beta \text{ for some } \beta \in \mathbb{R}^p\}$ is the the space of residuals. One can also interpret the value of the objective function $f(r)$ in (25) as measuring the ℓ_∞ norm of the gradient of the least-squares loss function $L(\beta) := \frac{1}{2}\|\mathbf{y} - \mathbf{X}\beta\|_2^2$ at some (possibly non-unique) point $\beta \in \mathbb{R}^p$. We establish the following equivalence.

Theorem 4.1. *The FS_ε algorithm is an instance of the subgradient descent method to solve problem (25), initialized at $r^0 = \mathbf{y}$ and with a constant step-size of ε at each iteration.*

Proof. $f(r)$ measures the maximum (over all columns $j \in \{1, \dots, p\}$) absolute value of the correlation between $\mathbf{X}_j$ and r, and so $f(\cdot)$ has the following representation:

$$f(r) := \|\mathbf{X}^T r\|_\infty = \max_{j \in \{1,\dots,p\}} |r^T \mathbf{X}_j| = \max_{\beta \in B_1} \ r^T \mathbf{X} \beta \ , \tag{26}$$

thus by (5) for any $r \in \mathbb{R}^n$ we have:

$$j^* \in \arg\max_{j \in \{1,\dots,p\}} |r^T \mathbf{X}_j| \iff \text{sgn}(r^T \mathbf{X}_{j^*}) \mathbf{X}_{j^*} \in \partial f(r) \ . \tag{27}$$

It therefore follows that Step (1.) of FS_ε is identifying a vector $g^k := \text{sgn}((r^k)^T \mathbf{X}_{j_k}) \mathbf{X}_{j_k} \in \partial f(r^k)$. Furthermore, Step (2.) of FS_ε is taking a subgradient step with step-size ε, namely $r^{k+1} := r^k - \varepsilon g^k$. By an easy induction, the iterates of FS_ε satisfy $r^k = \mathbf{y} - \mathbf{X}\beta^k \in P_{\text{res}}$ whereby $r^{k+1} := r^k - \varepsilon g^k = \Pi_{P_{\text{res}}}(r^k - \varepsilon g^k)$. □

As with the Mirror Descent interpretation of AdaBoost, we use the subgradient descent interpretation of FS_ε to obtain computational guarantees for a variety of step-size sequences.

Theorem 4.2. (Complexity of FS_ε) *Let $\beta_{LS} \in \arg\min_\beta \|\mathbf{y} - \mathbf{X}\beta\|_2^2$ be any least-squares solution of the regression model. With the constant shrinkage factor ε, for any $k \geq 0$ it holds that:*

$$\min_{i \in \{0,\ldots,k\}} \|\mathbf{X}^T r^i\|_\infty \leq \frac{\|\mathbf{X}\beta_{LS}\|_2^2}{2\varepsilon(k+1)} + \frac{\varepsilon \|\mathbf{X}\|_{1,2}^2}{2} \ . \tag{28}$$

If we a priori decide to run FS_ε for k iterations and set $\varepsilon := \frac{\|\mathbf{X}\beta_{LS}\|_2}{\|\mathbf{X}\|_{1,2}\sqrt{k+1}}$ then

$$\min_{i \in \{0,\ldots,k\}} \|\mathbf{X}^T r^i\|_\infty \leq \frac{\|\mathbf{X}\|_{1,2} \|\mathbf{X}\beta_{LS}\|_2}{\sqrt{k+1}} \ . \tag{29}$$

If instead the shrinkage factor is dynamically chosen as $\varepsilon = \varepsilon_k := \frac{|(r^k)^T \mathbf{X}_{j_k}|}{\|\mathbf{X}_{j_k}\|_2^2}$ (this is the Forward Stagewise algorithm (FS) [8]), then the bound (29) holds for all values of k without having to set k a priori.

Proof. Let $r_{LS} := \mathbf{y} - \mathbf{X}\beta_{LS}$ be the residuals of the least-squares solution $\mathbf{X}\beta_{LS}$, and it follows from orthogonality that $\mathbf{X}^T r_{LS} = 0$ if and only if β_{LS} is a least-squares solution, hence the optimal objective function value of (25) is $f^* = f(r_{LS}) = 0$ and $r^* := r_{LS}$ is an optimal solution of (25). As subgradient descent is simply Mirror Descent using the Euclidean prox function $d(r) = \frac{1}{2} r^T r$ on the space of the residuals $r \in P_{\text{res}}$, we apply Theorem 2.1 with $r = r^* = r_{LS}$. We have:

$$D(r^*, r^0) = D(r_{LS}, r^0) = \frac{1}{2}\|r_{LS} - r^0\|_2^2 = \frac{1}{2}\|r_{LS} - \mathbf{y}\|_2^2 = \frac{1}{2}\|\mathbf{X}\beta_{LS}\|_2^2 \ .$$

By Lemma A.1, $f(\cdot)$ has Lipschitz function values (with respect to the ℓ_2 norm) with Lipschitz constant $L_f = \|\mathbf{X}\|_{1,2} = \max_{j \in \{1,\ldots,p\}} \|\mathbf{X}_j\|_2$. Using these facts and $f(r_{LS}) = f^* = 0$, inequality (8) in Theorem 2.1 implies (28). Setting $\varepsilon := \frac{\|\mathbf{X}\beta_{LS}\|_2}{\|\mathbf{X}\|_{1,2}\sqrt{k+1}}$ and substituting into (28) yields (29). Finally, the step-size $\varepsilon_k := \frac{|(r^k)^T \mathbf{X}_{j_k}|}{\|\mathbf{X}_{j_k}\|_2^2}$ is just the step-size used to yield (14), and in this context $L_f = \|\mathbf{X}\|_{1,2}$ and $\|r^0 - r^*\|_2 = \|\mathbf{X}\beta_{LS}\|_2$ from which (29) follows again. □

The computational complexity bounds in Theorem 4.2 are of a similar spirit to those implied by Theorem 3.2, and can be interpreted as a guarantee on the "closeness" of the coefficient vectors $\{\beta^k\}$ to satisfying the classical optimality condition $\|\mathbf{X}^T r\|_\infty = 0$ for the (unconstrained) least-squares minimization problem.

Note that in the high-dimensional regime with $p > n$ and $\text{rank}(\mathbf{X}) = n$, we have that $\mathbf{y} = \mathbf{X}\beta_{LS}$, thus the selection of ε to obtain (29) does not require knowing (or computing) β_{LS}. Furthermore, we can always bound $\|\mathbf{X}\beta_{LS}\|_2 \leq \|\mathbf{y}\|_2$ and choose ε optimally with respect to the resulting bound

in (28). The interest in the interpretation given by Theorem 4.1 and the consequent complexity results in Theorem 4.2 is due to the sparsity and regularization properties (24) combined with the computational complexity, in contrast to β_{LS} which is not guaranteed to have any such sparsity or regularization properties. Indeed, due to Theorem 4.2, FS_ε now has the specific advantage of balancing the sparsity and regularization properties (24) and the complexity guarantees given by Theorem 4.2 through the choices of the shrinkage parameter ε and the number of iterations k.

A Appendix

Lemma A.1. *Suppose that $f(\cdot) : P \to \mathbb{R}$ is defined by $f(x) := \max_{\lambda \in Q} x^T A\lambda$. Then $f(\cdot)$ has Lipschitz function values with Lipschitz constant $L_f := \max_{\lambda \in Q} \|A\lambda\|_*$. In particular, if $Q \subseteq B^\sharp := \{\lambda : \|\lambda\|_\sharp \leq 1\}$ for some norm $\|\cdot\|_\sharp$, then $L_f \leq \|A\|_{\sharp,*}$ where $\|A\|_{\sharp,*}$ is the operator norm of A.*

Proof. Let $x, x' \in P$ and let $\tilde{\lambda} \in \arg\max_{\lambda \in Q} x^T A\lambda$, $\tilde{\lambda}' \in \arg\max_{\lambda \in Q} (x')^T A\lambda$. Then

$$\begin{aligned} f(x) - f(x') &= x^T A\tilde{\lambda} - (x')^T A\tilde{\lambda}' \\ &\leq x^T A\tilde{\lambda} - (x')^T A\tilde{\lambda} \\ &= (x - x')^T A\tilde{\lambda} \\ &\leq \|A\tilde{\lambda}\|_* \|x - x'\| \\ &\leq L_f \|x - x'\| \ , \end{aligned}$$

and symmetrically we have $f(x') - f(x) \leq L_f \|x' - x\|$. Clearly if $Q \subseteq B^\sharp$, then

$$L_f = \max_{\lambda \in Q} \|A\lambda\|_* \leq \max_{\lambda \in B^\sharp} \|A\lambda\|_* = \|A\|_{\sharp,*} \ .$$

□

Lemma A.2. *Let $e(\cdot) : \Delta_n \to \mathbb{R}$ be the entropy function, defined by $e(x) = \sum_{i=1}^n x_i \ln(x_i) + \ln(n)$, with induced Bregman distance $D(\cdot, \cdot)$, and let $w^0 = (1/n, \ldots, 1/n)$. Then, we have $\max_{w \in \Delta_n} D(w, w^0) = \ln(n)$.*

Proof. Clearly $e(w^0) = \ln(1/n) + \ln(n) = 0$ and since $\nabla e(w^0)_i = 1 + \ln(1/n) = 1 - \ln(n)$, we have for any $w \in \Delta_n$:

$$\nabla e(w^0)^T (w - w^0) = (1 - \ln(n)) \sum_{i=1}^n (w_i - 1/n) = (1 - \ln(n))(1 - 1) = 0 \ .$$

Thus we have:

$$D(w, w^0) = e(w) - e(w^0) - \nabla e(w^0)^T (w - w^0) = e(w) = \sum_{i=1}^n w_i \ln(w_i) + \ln(n) \leq \ln(n) \ .$$

Furthermore, the maximum is achieved by $e_1 = (1, 0, \ldots, 0)$. □

References

[1] S. Arora, E. Hazan, and S. Kale, *The multiplicative weights update method: a meta-algorithm and applications*, Theory of Computing **8** (2012), no. 1, 121–164.

[2] A. Beck and M. Teboulle, *Mirror descent and nonlinear projected subgradient methods for convex optimization*, Operations Research Letters **31** (2003), no. 3, 167–175.

[3] D. Bertsekas, *Nonlinear programming*, Athena Scientific, Belmont, MA, 1999.

[4] Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani, *Least angle regression (with discussion)*, Annals of Statistics **32** (2004), no. 2, 407–499. MR MR2060166 (2005d:62116)

[5] Y. Freund and R. E. Schapire, *A short introduction to boosting*, Journal of Japanese Society for Artificial Intelligence **14** (1999), no. 5, 771–780.

[6] Jerome H. Friedman, *Greedy function approximation: A gradient boosting machine*, Annals of Statistics **29** (2000), 1189–1232.

[7] T. Hastie, J. Taylor, R. Tibshirani, and G. Walther, *Forward stagewise regression and the monotone lasso*, Electronic Journal of Statistics **1** (2007), 1–29.

[8] T. Hastie, R. Tibshirani, and J. Friedman, *Elements of statistical learning: Data mining, inference, and prediction*, Springer Verlag, New York, 2009.

[9] L. Mason, J. Baxter, P. Bartlett, and M. R. Frean, *Boosting algorithms as gradient descent*, NIPS (Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller, eds.), The MIT Press, 1999, pp. 512–518.

[10] I. Mukherjee, C. Rudin, and R. E. Schapire, *The rate of convergence of AdaBoost*, Journal of Machine Learning Research - Proceedings Track **19** (2011), 537–558.

[11] A. S. Nemirovsky and D. B. Yudin, *Problem complexity and method efficiency in optimization*, Wiley, New York, 1983.

[12] Y. E. Nesterov, *Introductory lectures on convex optimization: a basic course*, Applied Optimization, vol. 87, Kluwer Academic Publishers, Boston, 2003.

[13] ———, *Smooth minimization of non-smooth functions*, Mathematical Programming **103** (2005), no. 1, 127–152.

[14] B. Polyak, *Introduction to optimization*, Optimization Software, Inc., New York, 1987.

[15] G. Rätsch and M. K. Warmuth, *Efficient margin maximizing with boosting*, Journal of Machine Learning Research **6** (2005), 2131–2152.

[16] C. Rudin, I. Daubechies, and R. E. Schapire, *The dynamics of AdaBoost: Cyclic behavior and convergence of margins*, Journal of Machine Learning Research **5** (2004), 1557–1595.

[17] C. Rudin, R.E. Schapire, and I. Daubechies, *Analysis of boosting algorithms using the smooth margin function*, Annals of Statistics (2007), 2723–2768.

[18] R. E. Schapire, *The boosting approach to machine learning: an overview*, Nonlinear Estimation and Classification (D. D. Denison, M. H. Hansen, C. Holmes, B. Mallick, and B. Yu, eds.), Springer, 2003.

[19] R. E. Schapire, Y. Freund, P. Bartlett, and W. S. Lee, *Boosting the margin: a new explanation for the effectiveness of voting methods*, Proc. 14th International Conference on Machine Learning, Morgan Kaufmann, 1997, pp. 322–330.

[20] R.E. Schapire and Y. Freund, *Boosting: Foundations and algorithms*, Adaptive computation and machine learning, Mit Press, 2012.

[21] S. Shalev-Shwartz and Y. Singer, *On the equivalence of weak learnability and linear separability: new relaxations and efficient boosting algorithms*, Machine Learning **80** (2010), no. 2-3, 141–163.

[22] Matus J. Telgarsky, *The fast convergence of boosting*, Advances in Neural Information Processing Systems 24 (J. Shawe-Taylor, R.S. Zemel, P. Bartlett, F.C.N. Pereira, and K.Q. Weinberger, eds.), 2011, pp. 1593–1601.

[23] R. Tibshirani, *Regression shrinkage and selection via the lasso*, Journal of the Royal Statistical Society, Series B **58** (1996), no. 1, 267–288.