Parallel Boosting with Momentum

Indraneel Mukherjee[1], Kevin Canini[1], Rafael Frongillo[2], and Yoram Singer[1]

[1] Google Inc.
indraneelenator@gmail.com, {canini,singer}@google.com
[2] Computer Science Division, University of California Berkeley
raf@cs.berkeley.edu

Abstract. We describe a new, simplified, and general analysis of a fusion of Nesterov's accelerated gradient with parallel coordinate descent. The resulting algorithm, which we call BOOM, for *boo*sting with *m*omentum, enjoys the merits of both techniques. Namely, BOOM retains the momentum and convergence properties of the accelerated gradient method while taking into account the curvature of the objective function. We describe a *distributed* implementation of BOOM which is suitable for massive high dimensional datasets. We show experimentally that BOOM is especially effective in large scale learning problems with rare yet informative features.

Keywords: accelerated gradient, coordinate descent, boosting.

1 Introduction

Large scale supervised machine learning problems are concerned with building accurate prediction mechanisms from massive amounts of high dimensional data. For instance, Bekkerman et al. [15] gave as an example the problem of training a Web search ranker on billions of documents using user-clicks as labels. This vast amount of data also comes with a plethora of user behavior and click patterns. In order to make accurate predictions, different characteristics of the users and the search query are extracted. As a result, each example is typically modeled as a very high-dimensional vector of binary features. Yet, within any particular example only a few features are non-zero. Thus, many features appear relatively rarely in the data. Nonetheless, many of the infrequent features are both highly informative and correlated with each other. Second order methods take into account the local curvature of the parameter space and can potentially cope with the skewed distribution of features. However, even memory-efficient second order methods such as L-BFGS [12] cannot be effective when the parameter space consists of 10^9 dimensions or more. The informativeness of rare features attracted practitioners who crafted domain-specific feature weightings, such as TF-IDF [13], and learning theorists who devised stochastic gradient and proximal methods that adapt the learning rate per feature [4,6]. Alas, our experiments with state-of-the-art stochastic gradient methods, such as AdaGrad [4], underscored the inability of stochastic methods to build very accurate predictors which take into account the highly correlated, informative, yet rare features. The focus of this paper is the design and analysis of a *batch* method that is highly parallelizable, enjoys the fast convergence rate of modern proximal methods, and incorporates a simple and efficient mechanism that copes with what we refer to as the *elliptical* geometry of the feature space.

Our algorithm builds and fuses two seemingly different approaches. Due to the scale of the learning problems, we have to confine ourselves to first order methods whose memory requirements are *linear* in the dimension. One of the most effective approaches among first order optimization techniques is Nesterov's family of accelerated gradient methods. Yuri Nesterov presented the core idea of accelerating gradient-based approaches by introducing a momentum term already in 1983 [8]. The seemingly simple modification to gradient descent obtains optimal performance for the complexity class of first-order algorithms when applied to the problem of minimization of smooth convex functions [7]. Nesterov and colleagues presented several modifications in order to cope with non-smooth functions, in particular composite (sum of smooth and non-smooth) functions. The paper of the late Paul Tseng provides an excellent, albeit technically complex, unified analysis of gradient acceleration techniques [16]. This paper is also the most related to the work presented here as we elaborate in the sequel.

Both Nesterov himself as well as the work of Beck and Teboulle [1], who built on Nesterov's earlier work, studied accelerated gradient methods for composite objectives. Of the two approaches, the latter is considered more efficient to implement as it requires storing parameters from only the last two iterations and a single projection step. Beck and Teboulle termed their approach FISTA for Fast Iterative Shrinkage Thresholding Algorithm. We start our construction with a derivation of an alternative analysis for FISTA and accelerated gradient methods in general. Our analysis provides further theoretical insights and distills to a broad framework within which accelerated methods can be applied. Despite their provably fast convergence rates, in practice accelerated gradient methods often exhibit slow convergence when there are strong correlations between features, amounting to elliptical geometry of the feature space. Putting the projection step aside, first order methods operate in a subspace which conforms with the span of the examples and as such highly correlated rare features can be overlooked. This deficiency is the rationale for incorporating an additional component into our analysis and algorithmic framework.

Coordinate descent methods [17] have proven to be very effective in machine learning problems, and particularly in optimization problems of elliptical geometries, as they can operate on each dimension separately. However, the time complexity of coordinate descent methods scale linearly with the dimension of the parameters and thus renders them inapplicable for high-dimensional problems. Several extensions that perform mitigated coordinate descent steps in parallel for multiple coordinates have been suggested. Our work builds specifically on the parallel boosting algorithm from [2]. However, parallel boosting algorithms on their own are too slow. See for instance [14] for a primal-dual analysis of the rate of convergence of boosting algorithms in the context of loss minimization.

Our approach combines the speed of accelerated gradient methods with the geometric sensitivity of parallel coordinate descent, and enjoys the merits of both approaches. We call the resulting approach BOOM, for *boos*ting with *m*omentum. As our experiments indicate, BOOM clearly outperforms both parallel boosting and FISTA over a range of medium- to large-scale learning problems. The aforementioned paper by Tseng also marries the two seemingly different approaches. It is based on extending the so called 1-memory version of accelerated gradient to inner-product norms, and indeed by employing a matrix-based norm Tseng's algorithm can model the elliptical geometry of the parameter space. However, our analysis and derivation are quite different than Tseng's. We take a more modular and intuitive approach, starting from the simplest form of proximal methods, and which may potentially be used with non-quadratic families of proximal functions.

The rest of the paper is organized as follows. We describe the problem setting in Sec. 2 and review in Sec. 3 proximal methods and parallel coordinate descent for composite objective functions. For concreteness and simplicity of our derivations, we focus on a setting where the non-smooth component of the objective is the 1-norm of the vector of parameters. In Sec. 4 we provide an alternative view and derivation of accelerated gradient methods. Our derivation enables a unified view of Nesterov's original acceleration and the FISTA algorithm as special cases of an abstraction of the accelerated gradient method. This abstraction serves as a stepping stone in the derivation of BOOM in Sec. 5. We provide experiments that underscore the merits of BOOM in Sec. 6. We briefly discuss a loosely-coupled distributed implementation of BOOM which enables its usage on very large scale problems. We comments on potential extensions in Sec. 7.

2 Problem Setting

We start by first establishing the notation used throughout the paper. Scalars are denoted by lower case letters and vectors by boldface letters, e.g. $\mathbf{w}$. We assume that all the parameters and instances are vectors of fixed dimension n. The inner product of two vectors is denoted as $\langle \mathbf{w}, \mathbf{v} \rangle = \sum_{j=1}^{n} w_j v_j$. We assume we are provided a labeled dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$ where the examples have feature vectors in $\mathbb{R}^n$. The unit vector whose j'th coordinate is 1 and the rest of the coordinates are 0 is denoted $\mathbf{e}_j$. For concreteness, we focus on binary classification and linear regression thus the labels are either in $\{-1, +1\}$ or real-valued accordingly. A special case which is nonetheless important and highly applicable is when each feature vector is sparse, that is only a fraction of its coordinates are non-zero. We capture the sparsity via the parameter κ defined as the maximum over the 0-norm of the examples, $\kappa = \max_i |\{j : x_{i,j} \neq 0\}|$. The convex optimization problem we are interested in is to associate an importance weight with each feature, thus find a vector $\mathbf{w} \in \mathbb{R}^n$ which minimizes:

$$\mathcal{L}(\mathbf{w}) = \sum_{i=1}^{m} \ell(\langle \mathbf{w}, \mathbf{x}_i \rangle, y_i) + \lambda_1 \|\mathbf{w}\|_1 \; = \; \mathcal{F}(\mathbf{w}) + \lambda_1 \|\mathbf{w}\|_1 \;,$$

where $\mathcal{F}$ denotes the smooth part of $\mathcal{L}$. As mentioned above, the non-smooth part of $\mathcal{L}$ is fixed to be the 1-norm of $\mathbf{w}$. Here $\ell : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}_+$ is a prediction loss function. In the concrete derivations presented in later sections we focus on two popular loss functions: the squared error $\ell(\hat{y}, y) = \frac{1}{2}(\hat{y} - y)^2$ for real-valued labels, and the logistic loss $\ell(\hat{y}, y) = 1 + e^{-y\hat{y}}$ for binary labels, where $\hat{y}$ is the prediction and y is the true label. Throughout the paper $\mathbf{w}^*$ denotes the point minimizing $\mathcal{L}$.

3 Proximal Methods

We begin by reviewing gradient descent for smooth objective functions and then show how to incorporate non-smooth 1-norm regularization. We review how the same minimization task can be carried out using parallel coordinate descent.

Gradient Descent. Let us first assume that $\lambda_1 = 0$, hence $\mathcal{L} = \mathcal{F}$. We denote by L the maximum curvature of $\mathcal{F}$ in any direction, so that $\nabla^2 \mathcal{F} \preceq \mathsf{L}I$. This property of $\mathcal{F}$ coincides with having a Lipschitz-continuous gradient of a Lipschitz constant L. We can locally approximate $\mathcal{F}$ around any point $\mathbf{v}$ using the following quadratic upper bound: $\mathcal{F}(\mathbf{w}+\boldsymbol{\delta}) \le \mathcal{F}(\mathbf{w}) + \langle \nabla\mathcal{F}(\mathbf{v}), \boldsymbol{\delta}\rangle + \frac{1}{2}\underbrace{\boldsymbol{\delta}^\dagger(\mathsf{L}I)\boldsymbol{\delta}}_{\mathsf{L}\|\boldsymbol{\delta}\|^2}$.

In each iteration, the new weight $\mathbf{w}_{t+1}$ is chosen to minimize the above bound anchored at the previous iterate $\mathbf{w}_t$, which amounts to,

$$\mathbf{w}_{t+1} = \mathbf{w}_t - (1/\mathsf{L})\nabla\mathcal{F}(\mathbf{w}_t) \ . \tag{1}$$

For this update, recalling $\mathcal{L} = \mathcal{F}$, simple algebra yields the following drop in the loss,

$$\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_{t+1}) \ge \|\nabla\mathcal{L}(\mathbf{w}_t)\|_2^2/(2\mathsf{L}) \ . \tag{2}$$

The guarantee of (2), yields that it takes $O\big(\mathsf{L}\|\mathbf{w}_0 - \mathbf{w}^*\|^2/\epsilon\big)$ iterations to obtain an approximation that is ϵ-close in the objective value.

Incorporating 1*-norm regularization.* When $\lambda_1 > 0$, the local approximation has to explicitly account for the ℓ_1 regularization, we have the following local approximation:

$$\mathcal{L}(\mathbf{w}+\boldsymbol{\delta}) \le \mathcal{F}(\mathbf{w}) + \langle \boldsymbol{\delta}, \nabla\mathcal{F}(\mathbf{w})\rangle + (\mathsf{L}/2)\|\boldsymbol{\delta}\|^2 + \lambda_1\|\mathbf{w}+\boldsymbol{\delta}\|_1 \ .$$

We can decompose the above Taylor expansion in a coordinate-wise fashion

$$\mathcal{L}(\mathbf{w}+\boldsymbol{\delta}) \le \mathcal{L}(\mathbf{w}) + \textstyle\sum_{j=1}^n f_j^{\mathsf{L}}(\delta_j) \ , \tag{3}$$

where

$$f_j^{\mathsf{L}}(\delta) \triangleq g_j\delta + (1/2)\mathsf{L}\delta^2 + \lambda_1|w_j+\delta| - \lambda_1|w_j| \ , \tag{4}$$

where g_j denotes $\nabla\mathcal{F}(\mathbf{w})_j$. Multiple authors (see e.g. [5]) showed that the value δ_j^* minimizing f_j satisfies $w_j + \delta_j^* = \mathcal{P}_{\mathsf{L}}^{g_j}(w_j)$, where

$$\mathcal{P}_{\mathsf{L}}^{g}(w) \triangleq \operatorname{sign}(w - g/\mathsf{L})\left[|w - g/\mathsf{L}| - \lambda_1/\mathsf{L}\right]_+ \ . \tag{5}$$

In fact, with this choice of δ_j^* we can show the following guarantee using standard arguments from calculus.

Lemma 1. *Let f_j^L be defined by (4) and δ_j^* chosen as above, then*

$$f_j^L(0) - f_j^L(\delta_j^*) \geq (\mathsf{L}/2)\delta_j^{*2} \ .$$

Gradient descent with ℓ_1 regularization iteratively chooses $\mathbf{w}_{t+1}$ to minimize the local approximation around $\mathbf{w}_t$, which amounts to

$$\forall j : w_{t+1,j} = \mathcal{P}_{\mathsf{L}}^{g_j}(w_{t,j}) \ , \tag{6}$$

and using (3) and Lemma 1 we obtain the following guarantee:

$$\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_{t+1}) \geq (\mathsf{L}/2)\|\mathbf{w}_t - \mathbf{w}_{t+1}\|^2 \ . \tag{7}$$

This guarantee yields the same convergence rate of $O\big(\mathsf{L}\|\mathbf{w}_0 - \mathbf{w}^*\|^2/\epsilon\big)$ as before.

Coordinate Descent. In coordinate descent algorithms, we take into account the possibility that the curvature could be substantially different for each coordinate. Let L_j denotes the maximum curvature of $\mathcal{F}$ along coordinate j, then as we show in the sequel that parallel coordinate descent achieves a convergence rate of $O(\sum_{j=1}^n \kappa\mathsf{L}_j(w_{0,j} - w_j^*)^2/\epsilon)$, where κ is the aforementioned sparsity parameter. Note that when the dataset is perfectly spherical and all the coordinate wise curvatures are equal to L_0, convergence rate simplifies to $O\big(\kappa\mathsf{L}_0\|\mathbf{w}_0 - \mathbf{w}^*\|^2/\epsilon\big)$, and we approximately recover the rate of gradient descent. In general though, the coordinate-specific rate can yield a significant improvement over the gradient descent bounds, especially for highly elliptical datasets.

Let us describe a toy setting that illustrates the advantage of coordinate descent over gradient descent when the feature space is elliptical. Assume that we have a linear regression problem where all of the labels are 1. The data matrix is of size $2(n–1)\times n$. The first n–1 examples are all the same and equal to the unit vector $\mathbf{e}_1$, that is, the first feature is 1 and the rest of the features are zero. The next $n-1$ examples are the unit vectors $\mathbf{e}_2, \ldots, \mathbf{e}_n$. The matrix $\nabla^2\mathcal{F}$ is diagonal where the first diagonal element is $n-1$ and the rest of the diagonal elements are all 1. The optimal vector $\mathbf{w}^*$ is $(1, \ldots, 1)$ and thus its squared 2-norm is n. The largest eigen value of $\nabla^2\mathcal{F}$ is clearly $n-1$ and thus $\mathsf{L} = n-1$. Therefore, gradient descent converges at a rate of $O(n^2/\epsilon)$. The rest of the eigen values of $\nabla^2\mathcal{F}$ are 1, namely, $\mathsf{L}_j = 1$ for $j \geq 2$. Since exactly one feature is “turned on” in each example $\kappa = 1$. We therefore get that coordinate descent converges at a rate of $O(n/\epsilon)$ which is substantially faster in terms of the dimensionality of the problem. We would like to note in passing that sequential coordinate descent converges to the optimal solution in exactly n steps.

To derive the parallel coordinate descent update, we proceed as before via a Taylor approximation, but this time have a separate approximation per coordinate: $\mathcal{F}(\mathbf{w}+\theta\mathbf{e}_j) \leq \mathcal{F}(\mathbf{w})+\theta\nabla\mathcal{F}(\mathbf{w})_j+(\mathsf{L}_j/2)\theta^2$. In order to simultaneously step in each coordinate, we employ the sparsity parameter κ and Jensen's inequality (see e.g. [3]) to show

$$\begin{aligned}\mathcal{F}\left(\mathbf{w} + \boldsymbol{\theta}/\kappa\right) &\leq \mathcal{F}(\mathbf{w}) + (1/\kappa)\textstyle\sum_{j=1}^n \left(\theta_j\nabla\mathcal{F}_j(\mathbf{w}) + (\mathsf{L}_j/2)\theta_j^2\right) \\ &= \mathcal{F}(\mathbf{w}) + \textstyle\sum_{j=1}^n \Big(g_j\theta_j/\kappa + (1/2)\mathsf{L}_j\kappa\left(\theta_j/\kappa\right)^2\Big) \ .\end{aligned}$$

By replacing θ_j/κ by δ_j, we get

$$\mathcal{F}(\mathbf{w}+\boldsymbol{\delta}) \le \mathcal{F}(\mathbf{w}) + \textstyle\sum_{j=1}^{n} \left(g_j\delta_j + (\kappa \mathsf{L}_j/2)\delta_j^2\right). \tag{8}$$

Introducing the ℓ_1 regularization terms on both sides, we have

$$\mathcal{L}(\mathbf{w}+\boldsymbol{\delta}) \le \mathcal{L}(\mathbf{w}) + \textstyle\sum_{j=1}^{n} f_j^{\kappa \mathsf{L}_j}(\delta_j) \ , \tag{9}$$

where $f_j^{\kappa \mathsf{L}_j}$ is as in (4). From our earlier discussions, we know the optimal choice $\boldsymbol{\delta}^*$ minimizing the previous expression satisfies $\delta_j^* = \mathcal{P}_{\kappa \mathsf{L}_j}^{g_j}(w_j)$. Accordingly, the parallel coordinate descent update is

$$w_{t+1,j} = \mathcal{P}_{\kappa \mathsf{L}_j}^{g_j}(w_{t,j}) \ , \tag{10}$$

and using (9) and Lemma 1, we have

$$\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}_{t+1}) \ge \textstyle\sum_{j=1}^{n} (\kappa \mathsf{L}_j/2)\left(w_{t,j} - w_{t+1,j}\right)^2 \ . \tag{11}$$

As before, with this guarantee on the progress of each iteration we can show that the convergence rate is $O(\sum_{j=1}^{n} \kappa \mathsf{L}_j (w_{0,j} - w_j^*)^2/\epsilon)$.

4 Accelerated Gradient Methods

In this section we give a new alternative derivation of the accelerated gradient method (AGM) that underscores and distills the core constituents of the method. Our view serves as a stepping stone towards the extension that incorporates parallel boosting in the next section. Accelerated methods take a more general approach to optimization than gradient descent. Instead of updating the parameters using solely the most recent gradient, AGM creates a sequence of auxiliary functions $\phi_t : \mathbb{R}^n \to \mathbb{R}^n$ that are increasingly accurate approximations to the original loss function. Further, the approximating functions uniformly converge to the original loss function as follows,

$$\phi_{t+1}(\mathbf{w}) - \mathcal{L}(\mathbf{w}) \le (1-\alpha_t)\left(\phi_t(\mathbf{w}) - \mathcal{L}(\mathbf{w})\right) \ , \tag{12}$$

where each $\alpha_t \in (0,1)$ and the entire sequence determines the rate of convergence. At the same time, AGM produces weight vectors $\mathbf{w}_t$ such that the true loss is lower bounded by the minimum of the auxiliary function as follows,

$$\mathcal{L}(\mathbf{w}_t) \le \min_{\mathbf{w}} \phi_t(\mathbf{w}) \ . \tag{13}$$

The above requirement yields directly the following lemma.

Lemma 2. *Assume that* (12) *and* (13) *hold in each iteration, then after t iterations, $\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}^*)$ is be upper bounded by,*

$$\left(\textstyle\prod_{k<t}(1-\alpha_k)\right)\left(\phi_0(\mathbf{w}^*) - \phi_0(\mathbf{w}_0) + \mathcal{L}(\mathbf{w}_0) - \mathcal{L}(\mathbf{w}^*)\right) \ .$$

The inequality (12) is achieved by choosing a linear function $\hat{\mathcal{L}}_t$ that lower bounds the original loss function, $\hat{\mathcal{L}}_t(\mathbf{w}) \leq \mathcal{L}(\mathbf{w})$, and then squashing ϕ_t towards the linear function by a factor α_t,

$$\phi_{t+1} = (1-\alpha_t)\phi_t + \alpha_t \hat{\mathcal{L}}_t. \tag{14}$$

Nesterov chose the initial auxiliary function to be a quadratic function centered at $\mathbf{v}_0 = \mathbf{w}_0$ (an arbitrary point vector), with curvature $\gamma_0 = \mathsf{L}/2$, and intercept $\phi_0^* = \mathcal{L}(\mathbf{w}_0)$, $\phi_0(\mathbf{w}) = \gamma_0\|\mathbf{w}-\mathbf{v}_0\|^2 + \phi_0^* = \frac{\mathsf{L}}{2}\|\mathbf{w}-\mathbf{w}_0\|^2 + \mathcal{L}(\mathbf{w}_0)$. Then, using an inductive argument, each ϕ_t is also a quadratic function with a center $\mathbf{v}_t$, curvature $\gamma_t = \gamma_0 \prod_{t'<t}(1-\alpha_{t'})$, and intercept ϕ_t^*

$$\phi_t(\mathbf{w}) = \gamma_t\|\mathbf{w}-\mathbf{v}_t\|^2 + \phi_t^* \ . \tag{15}$$

Moreover, if the linear function $\hat{\mathcal{L}}_t$ has slope $\boldsymbol{\eta}_t$, algebraic manipulations yield:

$$\begin{aligned}
\mathbf{v}_{t+1} &= \mathbf{v}_t - (\alpha_t/2\gamma_{t+1})\boldsymbol{\eta}_t && (16)\\
\phi_{t+1}^* &= \phi_{t+1}(\mathbf{v}_t) - (\phi_{t+1}(\mathbf{v}_t) - \phi_{t+1}(\mathbf{v}_{t+1}))\\
&= (1-\alpha_t)\phi_t^* + \alpha_t\hat{\mathcal{L}}_t(\mathbf{v}_t) - \gamma_{t+1}\|\mathbf{v}_{t+1}-\mathbf{v}_t\|^2\\
&= (1-\alpha_t)\phi_t^* + \alpha_t\hat{\mathcal{L}}_t(\mathbf{v}_t) - (\alpha_t^2/4\gamma_{t+1})\|\boldsymbol{\eta}_t\|^2 \ . && (17)
\end{aligned}$$

The last two equalities follow from (14), (15), and (16). To complete the algorithm and proof, we need to choose $\hat{\mathcal{L}}_t$, α_t and $\mathbf{w}_{t+1}$ so as to satisfy (13). Namely, $\mathcal{L}(\mathbf{w}_{t+1}) \leq \phi_{t+1}^*$. All acceleration algorithms satisfy these properties by tackling the expression in (17) in two steps. First, an intermediate point $\mathbf{y}_{t+1} = (1-\alpha_t)\mathbf{w}_t + \alpha_t\mathbf{v}_t$ is chosen. Note by linearity of $\hat{\mathcal{L}}_t$ we have,

$$\hat{\mathcal{L}}_t(\mathbf{y}_{t+1}) = (1-\alpha_t)\hat{\mathcal{L}}_t(\mathbf{w}_t) + \alpha_t\hat{\mathcal{L}}_t(\mathbf{v}_t) \ \leq \ (1-\alpha_t)\mathcal{L}(\mathbf{w}_t) + \alpha_t\hat{\mathcal{L}}_t(\mathbf{v}_t) \ .$$

Then, a certain proximal step is taken from $\mathbf{y}_{t+1}$ in order to reach $\mathbf{w}_{t+1}$, making sufficient progress in the process that satisfies,

$$\hat{\mathcal{L}}_t(\mathbf{y}_{t+1}) - \mathcal{L}(\mathbf{w}_{t+1}) \geq (\alpha_t^2/4\gamma_{t+1})\|\boldsymbol{\eta}_t\|^2 \ . \tag{18}$$

Combining the above two inequalities and inductively assuming that $\mathcal{L}(\mathbf{w}_t) \leq \phi_t^*$, it can be shown that $\mathcal{L}(\mathbf{w}_{t+1})$ is at most ϕ_{t+1}^* as given by (17).

The acceleration method in its abstract and general form is described in Algorithm 1. Further, based on the above derivation, this abstract algorithm ensures that (12) and (13) hold on each iteration. Consequently Lemma 2 holds as well as the following theorem.

Theorem 1. *The optimality gap* $\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}^*)$ *when* $\mathbf{w}_t$ *is constructed according to Algorithm 1 is at most,*

$$\left(\prod_{k<t}(1-\alpha_k)\right) \mathsf{L}\|\mathbf{w}_0 - \mathbf{w}^*\|^2 \ . \tag{19}$$

Proof. It suffices to show that the bound of Lemma 2 can be distilled to the bound of (19). Using the definition of ϕ_0 we have

$$\phi_0(\mathbf{w}^*) - \phi_0(\mathbf{w}_0) = (\mathsf{L}/2)\|\mathbf{w}^* - \mathbf{w}_0\|^2 \ . \tag{20}$$

Further, since the maximum curvature is L, the function has a Lipschitz-continuous gradient with Lipschitz constant L, namely, for any two vectors $\mathbf{x}, \mathbf{x}'$ the following inequality holds $\mathcal{L}(\mathbf{u}) - \mathcal{L}(\mathbf{u}') \leq (\mathsf{L}/2)\|\mathbf{x} - \mathbf{x}'\|^2$ and in particular for $\mathbf{u} = \mathbf{w}_0$ and $\mathbf{u}' = \mathbf{w}^*$. Summing the term from inequality (20) with the Lipschitz-continuity bound completes the proof.

We next present two concrete instantiations of Algorithm 1. Each variant chooses a lower bounding function $\hat{\mathcal{L}}_t$, a proximal step for reaching $\mathbf{w}_{t+1}$ from $\mathbf{y}_{t+1}$, and finally α_t so that (18) holds.

Nesterov's Acceleration [9]. In this setting there is no 1-norm regularization, $\lambda_1 = 0$, so that $\mathcal{L} = \mathcal{F}$. Here $\hat{\mathcal{L}}_t$ is the tangent plane to the surface of $\mathcal{L}$ at $\mathbf{y}_{t+1}$, $\boldsymbol{\eta}_t = \nabla\mathcal{L}(\mathbf{y}_{t+1})$ and $\hat{\mathcal{L}}_t(\mathbf{w}) = \mathcal{L}(\mathbf{y}_{t+1}) + \langle\boldsymbol{\eta}_t, \mathbf{w} - \mathbf{y}_{t+1}\rangle$, which by definition lower bounds $\mathcal{L}$. Further, let $\mathbf{w}_{t+1}$ be obtained from $\mathbf{y}_{t+1}$ using the proximal step in (1),

$$\mathbf{w}_{t+1} = \mathbf{y}_{t+1} - (1/\mathsf{L})\nabla\mathcal{L}(\mathbf{y}_{t+1}) \ . \tag{21}$$

Then, we have the same guarantee as in (2),

$$\mathcal{L}(\mathbf{y}_{t+1}) - \mathcal{L}(\mathbf{w}_{t+1}) \geq 1/(2\mathsf{L})\|\nabla\mathcal{L}(\mathbf{y}_{t+1})\|^2 \ .$$

By definition of $\boldsymbol{\eta}$, (18) holds if we choose α_t to satisfy

$$1/(2\mathsf{L}) = \alpha_t^2/(4\gamma_{t+1}), \tag{22}$$

which by expanding γ_k and using $\gamma_0 = \mathsf{L}/2$, simplifies to

$$\alpha_t^2/(1 - \alpha_t) = \textstyle\prod_{k<t}(1 - \alpha_k) \ . \tag{23}$$

From the above recurrence, the following inverse quadratic upper bound holds.

Lemma 3. *Assume that* (23) *holds, then* $\prod_{s<t}(1 - \alpha_s) \leq \frac{2}{(t+1)^2}$

Lemma 3 with Thm. 1 yields a rate of convergence of $O(\mathsf{L}\|\mathbf{w} - \mathbf{w}^*\|^2/\sqrt{\epsilon})$.

FISTA [1]. In FISTA, $\mathbf{w}_{t+1}$ is set from $\mathbf{y}_{t+1}$ using (6), namely, $w_{t+1,j} = \mathcal{P}_{\mathsf{L}}^{g_j}(y_{t+1,j})$. With this choice of $\mathbf{w}_{t+1}$, $\hat{\mathcal{L}}_t$ is constructed as follows,

$$\boldsymbol{\eta}_t = \mathsf{L}\left(\mathbf{y}_{t+1} - \mathbf{w}_{t+1}\right), \hat{\mathcal{L}}_t(\mathbf{w}) = \mathcal{L}(\mathbf{w}_{t+1}) + (1/2\mathsf{L})\|\boldsymbol{\eta}\|^2 + \langle\boldsymbol{\eta}, \mathbf{w} - \mathbf{y}_{t+1}\rangle \ . \tag{24}$$

The fact that $\hat{\mathcal{L}}_t$ lower bounds $\mathcal{L}$ is not obvious as was shown in [1]. We provide a more general proof later in Lemma 4. Note that the definition (24) implies that, $\hat{\mathcal{L}}_t(\mathbf{y}_{t+1}) - \mathcal{L}(\mathbf{w}_{t+1}) = (1/2\mathsf{L})\|\boldsymbol{\eta}\|^2$. Thus (18) holds when α_t is set as in (22) so as to satisfy the recurrence (23). Once again invoking Lemma 3 and Theorem 1, we obtain the same convergence rate of $O(\mathsf{L}\|\mathbf{w}_0 - \mathbf{w}^*\|^2)/\sqrt{\epsilon})$. The resulting algorithm may appear different than the original FISTA algorithm, but can be shown to be equivalent.

Algorithm 1. Accelerated Gradient

```
1: inputs: loss $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$,
       curvature $\mathsf{L} \in \mathbb{R}_+$.
2: initialize: $\mathbf{w}_0 = \mathbf{v}_0 \in \mathbb{R}^n$,
       $\gamma_0 \leftarrow \mathsf{L}/2$.
3: for $t = 0, 1, \ldots,$ do
4:     Pick $\alpha_t \in (0, 1)$
5:     Set $\gamma_{t+1} = \gamma_0 \prod_{k \leq t}(1 - \alpha_k)$.
6:     $\mathbf{y}_{t+1} \leftarrow (1 - \alpha_t)\mathbf{w}_t + \alpha_t \mathbf{v}_t$.
7:     Choose linear function
           $\hat{\mathcal{L}}_t \leq \mathcal{L}$ with slope $\boldsymbol{\eta}_t$
8:     Choose $\mathbf{w}_{t+1}$ using $\mathbf{y}_{t+1}$
           so that (18) holds
9:     $\mathbf{v}_{t+1} \leftarrow \mathbf{v}_t - (\alpha_t / 2\gamma_{t+1})\boldsymbol{\eta}_t$
10: end for
```

Algorithm 2. Boosting with Momentum

```
1: inputs: loss $\mathcal{L}$,
       sparsity $\kappa$ and $\mathsf{L}_1, \ldots, \mathsf{L}_n$.
2: initialize: $\mathbf{w}_0 = \mathbf{v}_0 \in \mathbb{R}^n$,
       $\forall j : \gamma_{0,j} \leftarrow \kappa \mathsf{L}_j / 2$.
3: for $t = 0, 1, \ldots,$ do
4:     Pick $\alpha_t \in (0, 1)$
5:     Set $\gamma_{t+1,j} = \gamma_{0,j} \prod_{k \leq t}(1 - \alpha_k)$.
6:     $\mathbf{y}_{t+1} \leftarrow (1 - \alpha_t)\mathbf{w}_t + \alpha_t \mathbf{v}_t$.
7:     Choose linear function
           $\hat{\mathcal{L}}_t \leq \mathcal{L}$ with slope $\boldsymbol{\eta}_t$
8:     Choose $\mathbf{w}_{t+1}$ using $\mathbf{y}_{t+1}$
           so that (28) holds
9:     $\forall j : v_{t+1,j} \leftarrow v_{t,j} - (\alpha_t / 2\gamma_{t+1,j})\eta_{t,j}$.
10: end for
```

5 BOOM: A Fusion

In this section we use the derivation of the previous section in a more general setting in which we combine the momentum-based gradient descent with parallel coordinate decent. As mentioned above we term the approach BOOM as it fuses the parallel *boo*sting algorithm from [2] with *m*omentum accelerated gradient methods [9,10,11].

The structure of this section will closely mirror that of Section 4, the only difference being the details for handling per-coordinate curvature. We start by modifying the auxiliary functions to account for the different curvatures L_j of $\mathcal{F}$ in each direction, starting with the initial function,

$$\phi_0(\mathbf{w}) = \textstyle\sum_{j=1}^n \gamma_{0,j}(w_j - v_{0,j})^2 + \phi_0^*.$$

The $\gamma_{0,j}$ are initialized to $\kappa \mathsf{L}_j/2$ for each coordinate j, and ϕ_0^* is set to $\mathcal{L}(\mathbf{w}_0)$:

$$\phi_0(\mathbf{w}) = \textstyle\sum_{j=1}^n (\mathsf{L}_j/2)(w_j - v_{0,j})^2 + \mathcal{L}(\mathbf{w}_0).$$

So instead of a spherical quadratic, the new auxiliary function is elliptical, with curvatures matching those of the smooth part $\mathcal{F}$ of the loss function. As before, we will choose a linear function $\hat{\mathcal{L}}_t \leq \mathcal{L}$ in each round and squash ϕ_t towards it to obtain the new auxiliary function. Therefore (14) continues to hold, and we can again inductively prove that ϕ_t continues to retain an elliptical quadratic form:

$$\phi_t(\mathbf{w}) = \sum_{j=1}^n \gamma_{t,j} \mathsf{L}_j (w_j - v_{t,j})^2 + \phi_t^*, \tag{25}$$

where $\gamma_{t,j} = \gamma_{0,j} \prod_{k<j}(1 - \alpha_k)$. In fact, if $\hat{\mathcal{L}}_t$ has slope $\boldsymbol{\eta}_t$, then we can show as before:

$$\begin{aligned} v_{t+1,j} &= v_{t,j} - (\alpha_t / 2\gamma_{t+1,j})\eta_{t,j} && (26) \\ \phi^*_{t+1} &= \phi_{t+1}(\mathbf{v}_t) - (\phi_{t+1}(\mathbf{v}_t) - \phi_{t+1}(\mathbf{v}_{t+1})) \\ &= (1-\alpha_t)\phi^*_t + \alpha_t \hat{\mathcal{L}}_t(\mathbf{v}_t) - \textstyle\sum_{j=1}^t \gamma_{t+1,j} \left(v_{t+1,j} - v_{t,j}\right)^2 \\ &= (1-\alpha_t)\phi^*_t + \alpha_t \hat{\mathcal{L}}_t(\mathbf{v}_t) - \textstyle\sum_{j=1}^n (\alpha_t^2 / 4\gamma_{t+1,j})\eta_{t,j}^2 . && (27) \end{aligned}$$

By picking $\mathbf{y}_{t+1} = (1-\alpha_t)\mathbf{w}_t + \alpha_t \mathbf{v}_t$ as before and arguing similarly we can inductively show the same invariant $\mathcal{L}(\mathbf{w}_t) \le \phi^*_t$, except $\mathbf{w}_{t+1}$ has to satisfy the following guarantee instead of (18):

$$\hat{\mathcal{L}}_t(\mathbf{y}_{t+1}) - \mathcal{L}(\mathbf{w}_{t+1}) \ge \textstyle\sum_{j=1}^n (\alpha_t^2 / 4\gamma_{t+1,j})\eta_{t,j}^2 . \tag{28}$$

The algorithm in this abstract form is given in Algorithm 2. We have now established that (12) and (13) hold in each iteration, and hence using Lemma 2 and arguing as before, we have the following theorem.

Theorem 2. *If Algorithm 2 outputs $\mathbf{w}_t$ in iteration t, then the suboptimality $\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}^*)$ can be upper bounded by $\left(\prod_{k<t}(1-\alpha_k)\right) \sum_{j=1}^n \kappa \mathsf{L}_{0,j}(w_{t,j} - w_j^*)^2$.*

In order to make Algorithm 2 concrete, we must choose $\{\alpha_t\}$, $\hat{\mathcal{L}}_t$, and $\mathbf{w}_{t+1}$ to satisfy the required constraints. Our selection will be analogous to the choices made by FISTA, but incorporating different curvatures. We first select $\mathbf{w}_{t+1}$ from $\mathbf{y}_{t+1}$ in a way similar to (10), $\forall j : w_{t+1,j} = \mathcal{P}^{g_j}_{\kappa \mathsf{L}_j}(y_{t+1,j})$, where $g_j = \nabla \mathcal{F}(\mathbf{y}_{t+1})_j$. Based on this choice, we select $\hat{\mathcal{L}}_t$ as follows:

$$\eta_{t,j} = \kappa \mathsf{L}_j \left(y_{t+1,j} - w_{t+1,j}\right) \tag{29}$$

$$\hat{\mathcal{L}}_t(\mathbf{w}) = \mathcal{L}(\mathbf{w}_{t+1}) + \textstyle\sum_{j=1}^n (\eta_j^2 / 2\kappa \mathsf{L}_j) + \langle \boldsymbol{\eta}_t, \mathbf{w} - \mathbf{y}_{t+1} \rangle . \tag{30}$$

By extending Lemma 2.3 in [1] we can show $\hat{\mathcal{L}}_t \le \mathcal{L}$.

Lemma 4. *If $\hat{\mathcal{L}}_t$ is defined as in* (30)*, then $\forall \mathbf{w} : \hat{\mathcal{L}}_t(\mathbf{w}) \le \mathcal{L}(\mathbf{w})$.*

The proof relies on optimality properties of the choice $\mathbf{w}_{t+1}$ and involves some subgradient calculus. We defer it to the supplementary materials. In addition to the lower bounding property $\hat{\mathcal{L}}_t \le \mathcal{L}$, from the definition (30), we also have

$$\hat{\mathcal{L}}_t(\mathbf{y}_{t+1}) - \mathcal{L}(\mathbf{w}_{t+1}) = \textstyle\sum_{j=1}^n (1/2\kappa \mathsf{L}_j)\eta_j^2 .$$

Then the constraint (28) will follow if we set α_t to satisfy:

$$\alpha_t^2 / (4\gamma_{t,j}) = 1/(2\kappa \mathsf{L}_j). \tag{31}$$

Expanding out $\gamma_{t,j}$ and using $\gamma_{0,j} = \mathsf{L}_j/2$ we obtain $\frac{\alpha_t^2}{1-\alpha_t} = \frac{2\gamma_{0,j}}{\kappa \mathsf{L}_j} \prod_{k<t}(1-\alpha_k) = \prod_{k<t}(1-\alpha_k)$, which is identical to (23).

We have now defined a particular instantiation of Algorithm 2, which satisfies the required conditions by the above discussion. We dub this instantiation *BOOM*, and give the full procedure in Algorithm 3 for completeness. Applying Theorem 2 and once again invoking Lemma 3, we have the following theorem, which yields a $O(\sum_{j=1}^n \kappa \mathsf{L}_j (w_{0,j} - w_j^*)^2 / \sqrt{\epsilon})$ convergence rate.

Theorem 3. *The $\mathbf{w}_t$ output by Algorithm 3 satisfies the bound*

$$\mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}^*) \leq (2/(t+1)^2) \textstyle\sum_{j=1}^n \kappa L_j (w_{0,j} - w_j^*)^2 .$$

As examples, consider linear and logistic loss. When $\mathcal{L}$ uses the linear loss, the curvature parameters are given by $\mathsf{L}_j = \sum_i x_{i,j}^2$ where $\mathbf{x}_i \in \mathbb{R}^n$ is the feature vector for the ith example in the training set. With logistic loss, the curvature can be bounded by $\mathsf{L}_j = (1/4) \sum_i x_{i,j}^2$ [5].

6 Experiments

Algorithm 3. BOOM

```
1: inputs: loss $\mathcal{L}$, regularizer $\lambda_1$,
2: parameters: $\kappa$ and $\mathsf{L}_1, \ldots, \mathsf{L}_n$
3: initialize: $\mathbf{w}_0 = \mathbf{v}_0 = \mathbf{0}$, $\gamma_{0,j} \leftarrow \kappa \mathsf{L}_j / 2$
4: for $t = 0, 1, \ldots,$ do
5:     Pick $\alpha_t$ satisfying (23)
6:     Set $\gamma_{t+1,j} \leftarrow \gamma_{0,j}\, \alpha_t^2 / (1 - \alpha_t)$.
7:     $\mathbf{y}_{t+1} \leftarrow (1 - \alpha_t)\mathbf{w}_t + \alpha_t \mathbf{v}_t$.
8:     $\mathbf{g} \leftarrow \nabla \mathcal{L}(\mathbf{y}_{t+1})$
9:     $w_{t+1,j} \leftarrow \mathcal{P}^{g_j}_{\kappa \mathsf{L}_j}(y_{t+1,j})$
10:    $v_{t+1,j} \leftarrow v_{t,j} - \frac{\alpha_t \kappa \mathsf{L}_j}{2\gamma_{t+1,j}}(y_{t+1,j} - w_{t+1,j})$
11: end for
```

We tested the performance of four algorithms: (1) parallel boosting, discussed in Section 3, (2) FISTA, discussed in Section 4, (3) BOOM, shown in Algorithm 3, and (4) sequential boosting. Note that for the first three algorithms, within each iteration, each coordinate in the feature space $\{1, \ldots, n\}$ could be assigned a separate processing thread that could carry out all the computations for that coordinate: e.g., the gradient, the step-size, and the change in weight for that coordinate. Therefore by assigning enough threads, or for the case of massively high dimensional data, implementing these algorithms on a distributed architecture and allocating enough machines, the computation time could remain virtually constant even as the number of dimensions grows. However, in sequential boosting a single iteration consists of choosing n coordinates uniformly at random with replacement, then making optimal steps along these coordinates, one by one in order. Therefore, in terms of computational time, one iteration of the sequential algorithm is actually on the order of n iterations of the parallel algorithms. The goal in including sequential boosting was to get a sense for how well the parallel algorithms can compete with a sequential algorithm, which in some sense has the best performance in terms of number of iterations. In all the experiments, when present, the curve corresponding to parallel boosting is shown in solid red lines, Fista in dashed blue, BOOM in solid lightgreen, and sequential boosting in dashed black.

We next describe the datasets. The synthetic datasets were designed to test how algorithms perform binary classification tasks with varying levels of sparsity, ellipticity, and feature correlations. We generated 9 synthetic datasets for binary classification, and 9 for linear regression. Each dataset has 1000 examples (split into train and test in a 2:1 ratio) and 100 binary features. Each feature is sparse or dense, occurring in 5% or 50%, resp. of the examples. The fraction of sparse features is 0, 0.5, or 1.0. Note that with a 0.5 fraction of sparse features, the ellipticity is higher than when the fraction is 0 or 1. For each of these settings, either 0, 50, or 100 percent of the features are grouped into equal blocks of identical features. The labels were generated by a random linear combination of the features, and contain 10% white label noise for binary classification, or 10% multiplicative Gaussian noise for the linear regression datasets. We ran each of the four algorithms for 100 iterations on each dataset.

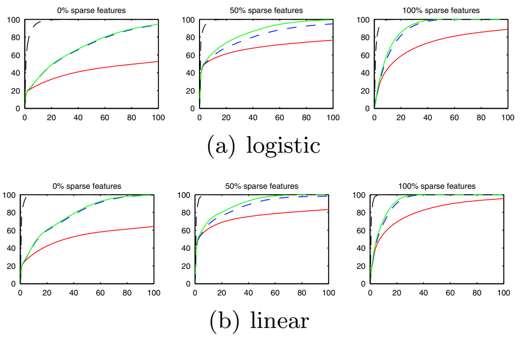

Fig. 1. (Synthetic data) The top and bottom rows represent logistic and linear regression experiments, resp. The columns correspond to partitions of the datasets based on fraction of sparse features. The x-axis is iterations and the y-axis is the progress after each iteration, averaged over the datasets in the partition.

For each algorithm, in each iteration we measure progress as the drop in loss since the first iteration, divided by the best loss possible, expressed as a percentage. For the training set we considered regularized loss, and for the test set unregularized loss. We partition the datasets based on whether the fraction of sparse features, is 0, 0.5 or 1. For each partition, we plot the progress on the training loss of each algorithm, averaged across all the datasets in the partition.

The results are tabulated in Figure 1 separately for logistic and linear regression. BOOM outperforms parallel boosting and FISTA uniformly across all datasets (as mentioned above, the sequential boosting is shown only as a benchmark). Against FISTA, the difference is negligible for the spherical datasets (where the fraction of sparse features is 0 or 1), but more pronounced for elliptical datsets

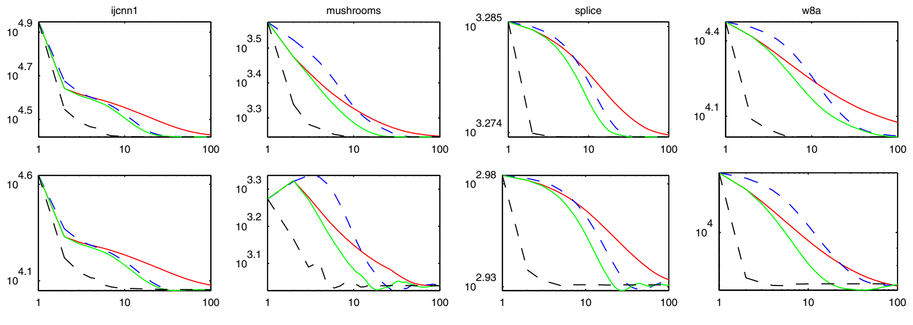

Fig. 2. (Medium-size real data) The top and bottom rows correspond to training and test data, resp. The x-axis measures iterations, and the y-axis measures loss with and without regularization for the training and test sets, resp.

(where the fraction of sparse features is 0.5). The plots on the test loss have the same qualitative properties, so we omit them.

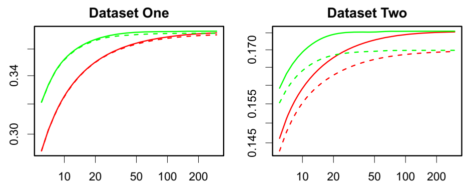

Fig. 3. (Large scale data) The x-axis is iterations and the y-axis is r-squared of the loss. The solid curves correspond to r-squared over training sets, and the dashed curve over the test-sets.

We next ran each algorithm for 100 iterations on four binary classification datasets: ijcnn1, splice, w8a and mushrooms. The criteria for choosing the datasets were that the number of examples exceeded the number of features, and the size of the datasets were reasonable. The continuous features were converted to binary features by discretizing them into ten quantiles. We made random 2:1 train/test splits in each dataset. The datasets can be found at `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/binary.html`. The log-log plot of train and test losses against the number of iterations are shown respectively in the left and right columns of Figure 2. BOOM significantly outperforms both FISTA and parallel boosting on both train and test loss on each dataset, suggesting that most real datasets are elliptical enough for the coordinate-wise approach of BOOM to make a noticeable improvement over FISTA.

Finally, we report large-scale experiments over two anonymized proprietary datasets. Both are linear regression datasets, and we report R-squared errors for the training set in solid lines, and test set in dashed. Dataset One contains 7.964B and 80.435M examples in the train and test sets, and 24.343M features, whereas Dataset Two contains 5.243B and 197.321M examples in the train and test sets, and 712.525M features. These datasets have very long tails, and the sparsity of the features varies drastically. In both datasets 90% of the features occur in less than 7000 examples, whereas the top hundred or so features occur in more than a hundred million examples each. Because of this enormous ellipticity, FISTA barely makes any progress, performing miserably even compared to parallel boosting, and we omit its plot. Sequential boosting is infeasibly slow to implement, and therefore we only show plots for BOOM and parallel boosting on these datasets. The results are shown in Figure 3, where we see that BOOM significantly outperforms parallel boosting on both datasets.

7 Conclusions

We presented in this paper an alternative abstraction of the accelerated gradient method. We believe that our more general view may enable new methods that can be accelerated via a momentum step. Currently the abstract accelerated gradient method (Algorithm 1) requires an update scheme which provides a guarantee on the drop in loss. Just as the choices of step size and slope were be abstracted, we suspect that this gradient-based condition can be relaxed, resulting in a potentially non-quadratic family of proximal functions. Thus, while the present paper focuses on accelerating parallel coordinate descent, in principle the techniques could be applied to other update schemes with a provable guarantee on the drop in loss.

References

1. Beck, A., Teboulle, M.: Fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM Journal of Imaging Sciences 2, 183–202 (2009)
2. Collins, M., Schapire, R.E., Singer, Y.: Logistic regression, AdaBoost and Bregman distances. Machine Learning 47(2/3), 253–285 (2002)
3. Cover, T.M., Thomas, J.A.: Elements of Information Theory. Wiley (1991)
4. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research, 2121–2159 (2011)
5. Duchi, J., Singer, Y.: Boosting with structural sparsity. In: Proceedings of the 26th International Conference on Machine Learning (2009)
6. McMahan, H.B., Streeter, M.: Adaptive bound optimization for online convex optimization. In: Proceedings of the Twenty Third Annual Conference on Computational Learning Theory (2010)
7. Nemirovski, A., Yudin, D.: Problem complexity and method efficiency in optimization. John Wiley and Sons (1983)
8. Nesterov, Y.: A method of solving a convex programming problem with convergence rate $o(1/k^2)$. Soviet Mathematics Doklady 27(2), 372–376 (1983)
9. Nesterov, Y.: Introductory Lectures on Convex Optimization. Kluwer Academic Publishers (2004)
10. Nesterov, Y.: Smooth minimization of nonsmooth functions. Mathematical Programming 103, 127–152 (2005)
11. Nesterov, Y.: Primal-dual subgradient methods for convex problems. Mathematical Programming 120(1), 221–259 (2009)
12. Nocedal, J., Wright, S.: Numerical Optimization, 2nd edn. Springer Series in Operations Research and Financial Engineering (2006)
13. Salton, G., Buckley, C.: Term weighting approaches in automatic text retrieval. Information Processing and Management 24(5) (1988)
14. Shalev-Shwartz, S., Singer, Y.: On the equivalence of weak learnability and linear separability: new relaxations and efficient algorithms. In: Proceedings of the Twenty First Annual Conference on Computational Learning Theory (2008)
15. Svore, K., Burges, C.: Large-scale learning to rank using boosted decision trees. In: Bekkerman, R., Bilenko, M., Langford, J. (eds.) Scaling Up Machine Learning. Cambridge University Press (2012)
16. Tseng, P.: On accelerated proximal gradient methods for convex-concave optimization. Submitted to SIAM Journal on Optimization (2008)
17. Tseng, P., Yun, S.: A coordinate gradient descent method for nonsmooth separable minimization. Mathematical Programming Series B 117, 387–423 (2007)

Appendix

In this appendix we provide technical proofs for theorems and lemmas whose proof were omitted from the body of the manuscript.

Proof of Lemma 1 Let $\partial f_j^{\mathsf{L}}$ denotes a subgradient of f_j^{L}. By convexity of f_j^{L} and optimality of δ_j^*, $\partial f_j^{\mathsf{L}}$ is increasing, in the range $(0, \delta_j^*)$. Further, by optimality of δ_j^*, there f_j^{L} has a zero subgradient at the point δ_j^*. The form of (4) implies that $\partial f_j^{\mathsf{L}}$ increases at the rate of at least L_j. Therefore, we get that

$$f_j^{\mathsf{L}}(0) - f_j^{\mathsf{L}}(\delta_j^*) = \int_{\delta_j^*}^{0} \partial f^{\mathsf{L}}(z)dz \geq \int_0^{\delta_j^*} \mathsf{L}z dz = \tfrac{\mathsf{L}}{2}{\delta_j^*}^2 \ . \ \square$$

Proof of Lemma 2 The proof amounts to application of (13) and (12) as follows,

$$\begin{aligned} & \forall \mathbf{w}^* : \mathcal{L}(\mathbf{w}_t) - \mathcal{L}(\mathbf{w}^*) \leq \phi_t(\mathbf{w}^*) - \mathcal{L}(\mathbf{w}^*) \ \ \text{[using (13)]} \\ \leq \ & \left(\textstyle\prod_{k<t}(1-\alpha_k)\right)(\phi_0(\mathbf{w}^*) - \mathcal{L}(\mathbf{w}^*)) \ \ \text{[recursively applying (12)]} \\ = \ & \left(\textstyle\prod_{k<t}(1-\alpha_k)\right)(\phi_0(\mathbf{w}^*) - \phi_0(\mathbf{w}_0) + \phi_0(\mathbf{w}_0) - \mathcal{L}(\mathbf{w}^*)) \\ = \ & \left(\textstyle\prod_{k<t}(1-\alpha_k)\right)(\phi_0(\mathbf{w}^*) - \phi_0(\mathbf{w}_0) + \mathcal{L}(\mathbf{w}_0) - \mathcal{L}(\mathbf{w}^*)) \ . \ \square \end{aligned}$$

Proof of Lemma 3 We have $\alpha_t^2 = \prod_{s\leq t}(1-\alpha_s)$. Notice that this implies an implicit relation $\alpha_{t+1}^2 = (1-\alpha_{t+1})\alpha_t^2$. To finish the proof, we show that $\alpha_{t-1} \leq 2/t$. We have

$$\frac{1}{\alpha_{t+1}} - \frac{1}{\alpha_t} = \frac{\alpha_t - \alpha_{t+1}}{\alpha_t\,\alpha_{t+1}} = \frac{\alpha_t^2 - \alpha_{t+1}^2}{\alpha_t\,\alpha_{t+1}(\alpha_t + \alpha_{t+1})} = \frac{\alpha_t^2 - \alpha_t^2(1-\alpha_{t+1})}{\alpha_t\,\alpha_{t+1}(\alpha_t + \alpha_{t+1})},$$

where the last equality follows from the implicit relation. Now, using the fact that $\alpha_t > \alpha_{t+1}$ we get that $\frac{1}{\alpha_{t+1}} - \frac{1}{\alpha_t} \geq \frac{\alpha_t^2\,\alpha_{t+1}}{\alpha_t\,\alpha_{t+1}\,2\alpha_t} = \frac{1}{2}$. We also have $\alpha_0^2/(1-\alpha_0) = 1$, and thus $\alpha_0 = (\sqrt{5}-1)/2 < 1$. Therefore, we get $1/\alpha_{t-1} \geq (t-1)/2 + 1/\alpha_0 \geq (t+1)/2$. This in turn implies that $\alpha_{t-1}^2/2 \leq \ 2/(t+1)^2$, and the proof is completed. $\square$

Proof of Lemma 4 The proof is very similar to the proof of Lemma 2.3 in [1]. The proof essentially works by first getting a first order expansion of the loss $\mathcal{L}$ around the point $\mathbf{w}_{t+1}$, and then shifting the point of expansion to $\mathbf{y}_{t+1}$.

In order to get the expansion around $\mathbf{w}_{t+1}$, we will separately get expansions for the smooth part $\mathcal{F}$ and the 1-norm parts of the loss. For the smooth part, we first take the first order expansion around $\mathbf{y}_{t+1}$:

$$\mathcal{F}(\mathbf{w}) \geq \mathcal{F}(\mathbf{y}_{t+1}) + \langle \nabla\mathcal{F}(\mathbf{y}_{t+1}), \mathbf{w} - \mathbf{y}_{t+1}\rangle \ . \tag{32}$$

We will combine this with the expansion in (8) around the point $\mathbf{y}_{t+1}$ to get an upper bound for the point $\mathbf{w}_{t+1}$. We have:

$$\mathcal{F}(\mathbf{y}_{t+1} + \boldsymbol{\delta}) \leq \mathcal{F}(\mathbf{y}_{t+1}) + \textstyle\sum_{j=1}^n \left(g_j\delta_j + (\kappa\mathsf{L}_j/2)\delta_j^2\right),$$

where $g_j = \nabla\mathcal{F}(\mathbf{y}_{t+1})_j$. Substituting $\mathbf{w}_{t+1}$ for $\mathbf{y}_{t+1} + \boldsymbol{\delta}$ we get

$$\mathcal{F}(\mathbf{w}_{t+1}) \leq \mathcal{F}(\mathbf{y}_{t+1}) + \langle \nabla\mathcal{F}(\mathbf{y}_{t+1}), \mathbf{w}_{t+1} - \mathbf{y}_{t+1}\rangle + \textstyle\sum_{j=1}^n (\kappa\mathsf{L}_j/2)(w_{t+1,j} - y_{t+1,j})^2.$$

Subtracting the previous equation from (32) and rearranging

$$\mathcal{F}(\mathbf{w}) \geq \mathcal{F}(\mathbf{w}_{t+1}) + \langle \nabla\mathcal{F}(\mathbf{y}_{t+1}), \mathbf{w} - \mathbf{w}_{t+1}\rangle - \textstyle\sum_{j=1}^n (\kappa\mathsf{L}_j/2)(w_{t+1,j} - y_{t+1,j})^2.$$

Next we get an expansion for the non-smooth 1-norm part of the loss. If $\boldsymbol{\nu}$ is a subgradient for the $\lambda_1\|\cdot\|_1$ function at the point $\mathbf{w}_{t+1}$, then we have the following expansion: $\lambda_1\|\mathbf{w}\|_1 \geq \lambda_1\|\mathbf{w}_{t+1}\|_1 + \langle \boldsymbol{\nu}, \mathbf{w} - \mathbf{w}_{t+1}\rangle$. Combining with the expansion of the smooth part we get

$$\mathcal{L}(\mathbf{w}) \geq \mathcal{L}(\mathbf{w}_{t+1}) + \langle \nabla\mathcal{F}(\mathbf{y}_{t+1}) + \boldsymbol{\nu}, \mathbf{w} - \mathbf{w}_{t+1}\rangle - \textstyle\sum_{j=1}^n (\kappa\mathsf{L}_j/2)(w_{t+1,j} - y_{t+1,j})^2.$$

We will carefully choose the subgradient vector $\boldsymbol{\nu}$ so that the jth coordinate of $\boldsymbol{\nu} + \nabla\mathcal{F}(\mathbf{y}_{t+1})$, i.e., $\nu_j + g_j$ satisfies

$$\nu_j + g_j = \kappa\mathsf{L}_j(w_{t+1,j} - y_{t+1,j}), \tag{33}$$

matching the definition of $\eta_{t,j}$ in (29). We will show how to satisfy (33) later, but first we show how this is sufficient to complete the proof. Using this we can write the above expansion as

$$\mathcal{L}(\mathbf{w}) \geq \mathcal{L}(\mathbf{w}_{t+1}) + \langle \boldsymbol{\eta}_t, \mathbf{w} - \mathbf{w}_{t+1}\rangle - \textstyle\sum_{j=1}^n (\kappa\mathsf{L}_j/2)(w_{t+1,j} - y_{t+1,j})^2.$$

By shifting the base in the inner product term to $\mathbf{y}_{t+1}$ we can write it as

$$\begin{aligned}\langle \boldsymbol{\eta}_t, \mathbf{w} - \mathbf{w}_{t+1}\rangle &= \langle \boldsymbol{\eta}_t, \mathbf{w} - \mathbf{y}_{t+1}\rangle + \langle \boldsymbol{\eta}_t, \mathbf{y}_{t+1} - \mathbf{w}_{t+1}\rangle \\ &= \langle \boldsymbol{\eta}_t, \mathbf{w} - \mathbf{y}_{t+1}\rangle + \textstyle\sum_{j=1}^n \kappa\mathsf{L}_j(w_{t+1,j} - y_{t+1,j})^2.\end{aligned}$$

Substituting this in the above, we get

$$\mathcal{L}(\mathbf{w}) \geq \mathcal{L}(\mathbf{w}_{t+1}) + \langle \boldsymbol{\eta}_t, \mathbf{w} - \mathbf{y}_{t+1}\rangle + \textstyle\sum_{j=1}^n (\kappa\mathsf{L}_j/2)(w_{t+1,j} - y_{t+1,j})^2.$$

Notice that the right side of the above expression is $\hat{\mathcal{L}}_t(\mathbf{w})$.

To complete the proof we need to show how to select $\boldsymbol{\nu}$ so as to satisfy (33). We will do so based on the optimality properties of $\mathbf{w}_{t+1}$. Recall that by choice $w_{t+1,j} = \mathcal{P}^{g_j}_{\kappa\mathsf{L}_j}(y_{t+1,j}) = y_{t+1,j} + \delta_j^*$, where δ_j^* minimizes the function $f_j^{\kappa\mathsf{L}_j}$ defined as in (4): $f_j^{\kappa\mathsf{L}_j}(\delta) = g_j\delta + \frac{1}{2}\kappa\mathsf{L}_j\delta^2 + \lambda_1\left|y_{t+1,j} + \delta\right| - \lambda_1\left|y_{t+1,j}\right|$, By optimality conditions for the convex function $f_j^{\kappa\mathsf{L}_j}$, we have $g_j + \kappa\mathsf{L}_j\delta_j^* + \nu_j = 0$, for some ν_j that is a subgradient of the $\lambda_1\left|y_{t+1,j} + \cdot\right|$ function at the point δ_j^*, or in other words, a subgradient of the function $\lambda_1\left|\cdot\right|$ at the point $\mathbf{w}_{t+1,j}$. Therefore there exists a subgradient $\boldsymbol{\nu}$ of the $\lambda_1\|\cdot\|_1$ function at the point satisfying (33), completing the proof.