FLAG n' FLARE: Fast Linearly-Coupled Adaptive Gradient Methods

Xiang Cheng * Farbod Roosta-Khorasani † Stefan Palombo ‡ Peter L. Bartlett §
Michael W. Mahoney ¶

February 25, 2018

Abstract

We consider first order gradient methods for effectively optimizing a composite objective in the form of a sum of smooth and, potentially, non-smooth functions. We present accelerated and adaptive gradient methods, called FLAG and FLARE, which can offer the best of both worlds. They can achieve the optimal convergence rate by attaining the optimal first-order oracle complexity for smooth convex optimization. Additionally, they can adaptively and non-uniformly re-scale the gradient direction to adapt to the limited curvature available and conform to the geometry of the domain. We show theoretically and empirically that, through the compounding effects of acceleration and adaptivity, FLAG and FLARE can be highly effective for many data fitting and machine learning applications.

*Department of EECS, UC Berkeley. Email: x.cheng@berkeley.edu

†School of Mathematics and Physics, University of Queensland. Email: fred.roosta@uq.edu.au

‡Department of EECS, UC Berkeley. Email: s.palombo@berkeley.edu

§Department of EECS and Department of Statistics, UC Berkeley. Email: bartlett@cs.berkeley.edu

¶Department of Statistics, UC Berkeley. Email: mmahoney@stat.berkeley.edu

Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS) 2018, Lanzarote, Spain. PMLR: Volume 84.

1 Introduction

Optimization problems which exhibit particular structure appear often in many science, engineering, data analysis and machine learning applications [7, 9, 42]. It is, by now, a well-known fact that taking proper advantage of the problem structure can lead to better performance guarantees and more effective algorithms compared to black-box, structure-oblivious methods; see [34, Section 4.1] for a more detailed discussion and [20, 26, 30, 31, 32, 36, 37, 39, 40, 40, 44] for many practical examples.

Here, we consider the optimization problem with the particular form

$$\min_{\mathbf{x}\in\mathcal{C}} F(\mathbf{x}) = f(\mathbf{x}) + h(\mathbf{x}), \tag{1}$$

where $f : \mathbb{R}^d \to \mathbb{R}$ and $h : \mathbb{R}^d \to \mathbb{R}$ are, respectively, smooth and potentially non-smooth, closed proper convex functions and $\mathcal{C}$ is a closed convex set. Optimization problems of the form (1) are often known as *composite optimization* and arise in many applications. Notable examples are those in which h encapsulates an *a priori* assumption on the sought-after parameter $\mathbf{x}$, e.g., sparsity or low-rank structure.

In problems of the form (1) with non-smooth h, sub-gradient methods [3, 5, 9] can result in algorithms with sub-linear convergence rate of order

$$F(\mathbf{x}_k) - \min_{\mathbf{x}\in\mathcal{C}} F(\mathbf{x}) \leq \mathcal{O}\left(\frac{1}{\sqrt{k}}\right),$$

where $\mathbf{x}_k$ is the k-th iterate. However, if h is "simple", then algorithms with superior convergence rates exist. In particular, the class of Iterative Shrinkage-Thresholding Algorithms (ISTA) algorithms, [8, 12, 13, 37], can improve upon the slow rate of sub-gradient methods and, indeed, recover the convergence rate of the standard gradient de-

scent method, i.e.,

$$F(\mathbf{x}_k) - \min_{\mathbf{x}\in\mathcal{C}} F(\mathbf{x}) \leq \mathcal{O}\left(\frac{1}{k}\right).$$

However, ISTA, both empirically and theoretically, has been shown to be very slow, e.g., see [8, section 6]. As a result, there have been many efforts to *accelerate* ISTA by non-trivial modifications, all of which are multi-step methods, i.e., the next iterate is computed from several previous ones. Most notably, the celebrated Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [4] exploits smoothness of f and simple structure of h to improve the convergence rate to

$$F(\mathbf{x}_k) - \min_{\mathbf{x}\in\mathcal{C}} F(\mathbf{x}) \leq \mathcal{O}\left(\frac{1}{k^2}\right),$$

which is known to be optimal for first order (gradient) methods [29] and matches that of Nesterov's accelerated algorithms [33,34] for smooth problems. Similar accelerated multi-step methods have also been investigated for solving non-smooth problems of the form (1), e.g., [6, 18, 35, 46]. The great theoretical properties as well as empirical performance of such accelerated methods have prompted many authors to try to understand the underlying mechanism and the natural scope of the acceleration concept, e.g., physical momentum, relations to other first-order algorithms as well as geometrical and continuous-time dynamics point of view [1, 10, 19, 25, 28, 43, 47]. Most relevant to the present paper is the result of [1] in which an acceleration scheme can was designed by an appropriate *linear coupling* of the gradient and mirror descent steps to draw upon their complementary characteristics. The insightful idea of [1] constitutes the first main ingredient for our proposed algorithms.

In addition to acceleration through a multi-step scheme and employing information from previous iterates, another approach to improve the empirical as well as the theoretical properties of first order methods for (1) is by incorporating previous sub-gradients in the form of adaptively choosing a preconditioner for each gradient (mirror) step. This idea was first pioneered in Adagrad [17], a sub-gradient method designed for online learning, [21]. Through the use of the history of the sub-gradients from previous iterations, Adagrad scales the current sub-gradient to *adapt* to the geometry of the domain. In particular, the coordinates of the search direction are non-uniformly scaled in order to take larger steps along the coordinates with smaller sub-derivatives and, correspondingly, smaller steps along those with larger sub-derivatives. Loosely speaking, this makes the optimization problem better-conditioned. For these reasons, Adagrad has been shown to be highly-suited to data fitting problems with, for example, sparse data [16, 38]. This work has led to many related algorithms that have been widely used in machine learning applications, e.g., RMSProp [45], ESGD [14], Adam [24], and Adadelta [48]. The second critical ingredient for our algorithmic design is based on this successful idea of *adaptivity* for non-uniform scaling of the search direction's coordinates.

In this paper, we present methods which offer the best of both worlds. More precisely, we draw upon the ideas of linear coupling [1] and adaptivity [17], introduce a fast linearly-coupled adaptive gradient method (FLAG) along with its relaxation (FLARE), and demonstrate their theoretical and empirical performance for solving the composite problem (1). We show that FLAG and FLARE can be equivalently regarded as adaptive versions of FISTA or alternatively, as accelerated versions of AdaGrad adopted for problem (1). In other words, like Nesterov's accelerated algorithm and its proximal variant, FISTA, our methods achieve the optimal convergence rate of $1/k^2$ and like AdaGrad our methods adaptively choose a regularizer, in a way that performs almost as well as the best choice of regularizer in hindsight. These two desirable effects contribute to the improved theoretical properties as well as practical performance of FLAG and FLARE.

The rest of this paper in organized as follows. Notation, assumptions and definitions used throughout the paper are introduced in Section 1.1. Our main algorithm, FLAG, and its theoretical properties are presented in Section 2. FLAG can, at times, require more computational effort than FISTA due to the sub-routine involving the linear coupling. As a result, in Section 3, we present a relaxed version of FLAG, dubbed FLARE, which by replacing this potentially expensive step of FLAG, alleviates this problem. Sections 4 contains extensive numerical experiments demonstrating the performance of FLAG and FLARE as compared with FISTA. Conclusions and further thoughts are gathered in Section 5. The details of the proofs are deferred to Appendix A.

1.1 Notation, Assumptions and Definitions

Notation: In what follows, vectors are considered as column vectors and are denoted by bold lower case letters, e.g., $\mathbf{x}$ and matrices are denoted by regular capital letters, e.g., A. We overload the "diag" oper-

ator as follows: for a given matrix A and a vector $\mathbf{x}$, $\mathrm{diag}(A)$ and $\mathrm{diag}(\mathbf{x})$ denote the vector made from the diagonal elements of A and a diagonal matrix made from the elements of $\mathbf{v}$, respectively. Vector norms $\|\mathbf{x}\|_1$, $\|\mathbf{x}\|_2$ and $\|\mathbf{x}\|_\infty$ denote the standard ℓ_1, ℓ_2 and ℓ_∞ respectively. We adopt the `Matlab` notation for accessing the elements of vectors and matrices, i.e., i^{th} components of a vector $\mathbf{x}$ is indicated by $\mathbf{x}(i)$ and $A(i,:)$ denotes the entire i^{th} row of the matrix A. Finally, $A_k \leftarrow [A_{k-1}, \mathbf{v}]$ signifies that A_k is the augmentation of the matrix A_{k-1} with the column vector $\mathbf{v}$. The optimal value of F is denoted by $F^* = \min_{\mathbf{x}\in\mathcal{C}} F(\mathbf{x})$. Finally, the sub-differential of a convex function, h, at a point, $\mathbf{x}$, is denoted by $\partial h(\mathbf{x})$.

Assumptions: Throughout this paper, we make the following assumptions for f and h.

A.1 f is convex and continuously differentiable with L-Lipschitz continuous gradient, i.e.,

$$\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\|_2 \leq L\|\mathbf{x}-\mathbf{y}\|_2, \quad \forall \mathbf{x},\mathbf{y}\in\mathcal{C}, \quad (2)$$

and

A.2 h is convex (but possibly non-smooth)

Definitions: The proximal operator [37] associated with f, h and L is defined as

$$\mathbf{prox}(\mathbf{x}) := \arg\min_{\mathbf{y}\in\mathcal{C}} \; h(\mathbf{y}) + \frac{L}{2}\|\mathbf{y} - \big(\mathbf{x} - \frac{1}{L}\nabla f(\mathbf{x})\big)\|_2^2. \quad (3)$$

For a symmetric positive definite (SPD) matrix S, define $\psi(\mathbf{x}) := \mathbf{x}^T S\mathbf{x}/2 = \|\mathbf{x}\|_S^2$. The Bregman divergence associated with ψ is defined as $B_\psi(\mathbf{x},\mathbf{y}) := \psi(\mathbf{x}) - \psi(\mathbf{y}) - \langle\nabla\psi(\mathbf{y}), \mathbf{x}-\mathbf{y}\rangle = 0.5\|\mathbf{x}-\mathbf{y}\|_S^2$. It is easy to see that the dual of $\psi(\mathbf{x})$ is given by

$$\psi^*(\mathbf{x}) = \sup_{\mathbf{v}\in\mathbb{R}^d} \langle\mathbf{x},\mathbf{v}\rangle - \psi(\mathbf{v}) = \frac{1}{2}\mathbf{x}^T S^{-1}\mathbf{x} = \|\mathbf{x}\|_{S^{-1}}^2. \quad (4)$$

Note that ψ is 1-strongly convex with respect to the norm $\|\mathbf{x}\|_S := \sqrt{\mathbf{x}^T S\mathbf{x}}$, i.e., $\forall \mathbf{x},\mathbf{y}\in\mathcal{C}$, we have $\psi(\mathbf{x}) \geq \psi(\mathbf{y}) + \langle\nabla\psi(\mathbf{y}), \mathbf{x}-\mathbf{y}\rangle + \frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_S^2$. Finally, throughout our analysis, we will use the fact that, for any $\mathbf{z}\in\mathcal{C}$,

$$\begin{aligned}&\langle S(\mathbf{x}-\mathbf{y}), \mathbf{x}-\mathbf{z}\rangle \\ =&\frac{1}{2}\|\mathbf{x}-\mathbf{y}\|_S^2 + \frac{1}{2}\|\mathbf{x}-\mathbf{z}\|_S^2 - \frac{1}{2}\|\mathbf{y}-\mathbf{z}\|_S^2. \end{aligned} \quad (5)$$

2 FLAG

In this section, we present our main algorithm, FLAG (Algorithm 1), and give its main convergence properties in Theorem 1. As mentioned in Section 1, FLAG incorporates techniques from linear coupling of [1] and adaptivity of [17]. At a very high-level, the core of FLAG consists of the following five essential ingredients.

1. A gradient descent step (Step 2 of Algorithm 1),
2. Construction of the adaptive regularization (Steps 3-7 of Algorithm 1),
3. Update of the adaptive stepsize (Step 9 of Algorithm 1),
4. A mirror descent step (Step 10 of Algorithm 1),
5. Linear combination of the gradient and the mirror descent directions (Step 11 of Algorithm 1).

Algorithm 1 FLAG

Input: $\mathbf{x}_1 = \mathbf{y}_1 = \mathbf{z}_1$, $\eta_0 = 0$, $G_0 = [\text{ empty }]$, $\delta > 0$, T, and $\epsilon = 1/(6dT^3)$

1: **for** k=1 to T **do**
2: $\quad\mathbf{y}_{k+1} \leftarrow \mathbf{prox}(\mathbf{x}_k)$
3: $\quad\mathbf{p}_k \leftarrow -L(\mathbf{y}_{k+1} - \mathbf{x}_k)$
4: $\quad\mathbf{g}_k \leftarrow \dfrac{\mathbf{p}_k}{\|\mathbf{p}_k\|_2}$
5: $\quad G_k \leftarrow \big[G_{k-1}, \mathbf{g}_k\big]$
6: $\quad\mathbf{s}_k(i) \leftarrow \|G_k(i,:)\|_2$
7: $\quad S_k \leftarrow \mathrm{diag}(\mathbf{s}_k) + \delta\mathbb{I}$
8: $\quad L_k \leftarrow L\mathbf{g}_k^T S_k^{-1}\mathbf{g}_k$
9: $\quad\eta_k \leftarrow \dfrac{1}{2L_k} + \sqrt{\dfrac{1}{4L_k^2} + \dfrac{\eta_{k-1}^2 L_{k-1}}{L_k}}$
10: $\quad\mathbf{z}_{k+1} \leftarrow \arg\min_{\mathbf{z}\in\mathcal{C}} \langle\eta_k\mathbf{p}_k, \mathbf{z}-\mathbf{z}_k\rangle + \dfrac{1}{2}\|\mathbf{z}-\mathbf{z}_k\|_{S_k}^2$
11: $\quad\mathbf{x}_{k+1} \leftarrow \mathrm{BinarySearch}(\mathbf{z}_{k+1}, \mathbf{y}_{k+1}, \epsilon)$
12: **end for**

Output: $\mathbf{y}_{T+1}$

The subroutine *BinarySearch* is given in Algorithm 2, where $Bisection(r, 0, 1, \epsilon)$ is the usual bisection routine for finding the root of a single variable function $r(t)$ in the interval $(0,1)$ and to the accuracy of ϵ. More specifically, for a root r^* such that $r(t^*) = 0$ and given $r(0)\cdot r(1) < 0$, the subroutine $Bisection(r, 0, 1, \epsilon)$ returns an approximation $t \in (0,1)$ to t^* such that $|t - t^*| \leq \epsilon$ and this is done with only $\log(1/\epsilon)$ function evaluations; see [2, Section 3.2] for details and example `Matlab` code.

Algorithm 2 BinarySearch

Input: $\mathbf{z}$, $\mathbf{y}$, and ϵ

Define the univariate function $r(t) := \langle \mathbf{prox}\left(t\mathbf{y} + (1-t)\mathbf{z}\right) - \left(t\mathbf{y} + (1-t)\mathbf{z}\right), \mathbf{y} - \mathbf{z}\rangle$
if $r(1) \geq 0$ **then**
  Return $\mathbf{y}$
end if
if $r(0) \leq 0$ **then**
  Return $\mathbf{z}$
end if
$t = \text{Bisection}(r, 0, 1, \epsilon)$
Return $\mathbf{x} = t\mathbf{y} + (1-t)\mathbf{z}$

We are now ready to present our main result, Theorem 1, which gives the convergence properties of Algorithm 1.

Theorem 1 (Convergence of FLAG) *Let Assumptions **A.1** and **A.2** hold and define* $D_\infty := \sup_{\mathbf{x},\mathbf{y}\in\mathcal{C}} \|\mathbf{x}-\mathbf{y}\|_\infty$. *For any* $\mathbf{u} \in \mathcal{C}$, *after* T *iterations of Algorithm 1, we get*

$$F(\mathbf{y}_{T+1}) - F(\mathbf{u}) \leq \mathcal{O}\left(\frac{\beta L D_\infty^2}{T^2}\right),$$

for some $\beta \in [1, d]$. *Furthermore, each iteration takes time at most* $\mathcal{O}(\mathcal{T}_{prox} \cdot \log(dT^3))$, *where* $\mathcal{T}_{prox}$ *is the cost of evaluating* ***prox*** *in (3).*

Remark 1: Recall that the convergence rate of FISTA is given by

$$F(y_{T+1}) - F^* \leq \mathcal{O}\left(\frac{L D_2^2}{T^2}\right),$$

where $D_2 := \sup_{\mathbf{x},\mathbf{y}\in\mathcal{C}} \|\mathbf{x}-\mathbf{y}\|_2$ [4, Theorem 4.4]. A quick comparison between FISTA's upper bound with that of FLAG in Theorem 1 implies that the "competitive factor" of FLAG over FISTA is

$$\text{Competitive Factor} = \frac{\beta D_\infty^2}{D_2^2}.$$

For (1), consider a box-constraint of the form $\mathcal{C} = \{\mathbf{x};\ \|\mathbf{x}\|_\infty \leq 1\}$. It is easy to see that since $D_2 = \sqrt{d} D_\infty$, we have Competitive Factor $\in [1/d, 1]$. In such settings, the adaptivity introduced by FLAG can offer a significant improvement in the convergence properties. This is indeed similar to the improvement obtained by Adagrad over proximal subgradient methods[1].

Remark 2: From the proof of Theorem 1, we can see that

$$\beta = \frac{\left(\sum_{i=1}^d \sqrt{\sum_{t=1}^T [\tilde{\mathbf{g}}_t]_i^2}\right)^2}{T},$$

where $\tilde{\mathbf{g}}_t := \mathbf{g}_t / \|\mathbf{g}_t\|$. For illustration purposes only, let us consider $d = 4$, and $T = 3$. Indeed, for the following gradient histories, we can verify that $\beta \in [1, 4]$, e.g.,

$$[\tilde{\mathbf{g}}_1, \tilde{\mathbf{g}}_2, \tilde{\mathbf{g}}_3] = \begin{bmatrix} 1 & -1 & -1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \Longrightarrow \beta = 1,$$

$$[\tilde{\mathbf{g}}_1, \tilde{\mathbf{g}}_2, \tilde{\mathbf{g}}_3] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \Longrightarrow \beta = 3,$$

$$[\tilde{\mathbf{g}}_1, \tilde{\mathbf{g}}_2, \tilde{\mathbf{g}}_3] = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} \end{bmatrix} \Longrightarrow \beta = 4.$$

3 FLARE

The "BinarySearch" in Step 11 of Algorithm 1 can be the bottleneck of the computations. Indeed, from Theorem 1 it can be seen that the running time of FLAG, in the worst case, is dominated by the number of **prox** evaluations involved in the root finding procedure of Algorithm 2. As a result, despite the fact that FLAG achieves the same accelerated convergence rate as FISTA, its per-iteration cost can be much higher than what adaptivity can make up for; see examples of Section 4. In this section, we modify FLAG to obtain a relaxed version, FLARE, whose per-iteration complexity is theoretically similar to FLAG, but empirically is shown to be almost identical to that of FISTA, i.e., $\mathcal{O}(\mathcal{T}_{\mathbf{prox}})$.

The proposed relaxation in FLARE will be done by "guessing" L_k in FLAG, i.e., Step 8 of Algorithm 1, at iteration k and performing the update without immediately resorting to "BinearySearch". We subsequently verify in the next iteration that the guessed L_k is not too far from the truth; otherwise, we repeat the previous iteration with a better guess. The resulting relaxation is given in Algorithm 3.

[1] For the non-smooth settings considered by Adagrad, the competitive factor is in the form of $\sqrt{\beta} D_\infty / D_2$

Algorithm 3 FLARE

Input: $\mathbf{x}_1 = \mathbf{y}_1 = \mathbf{z}_1$, $\eta_0 = 0$, $G_0 = [\text{ empty }]$, $\delta > 0$, $\gamma > 1$, $\lambda > 1$, T, and $\epsilon \leq 1/(6dT^3)$

1: **while** $k \leq T$ **do**
2: accept $\leftarrow$ FALSE
3: $i \leftarrow 0$
4: **while** accept $\neq$ TRUE **do** $i = i + 1$
5: **if** $i \leq \log \frac{d}{\epsilon}$ **then**
6: $\tilde{L}_k \leftarrow L_{k-1} \cdot \gamma^i$
7: $[\eta_k, \mathbf{x}_k, G_k, S_k, L_k, \mathbf{y}_{k+1}, \mathbf{z}_{k+1}, \text{accept}] \leftarrow$ A&V$(G_{k-1}, S_{k-1}, \eta_{k-1}, \tilde{L}_{k-1}, \mathbf{y}_k, \mathbf{z}_k, \tilde{L}_k, \lambda)$
8: **else**
9: $[\eta_k, \mathbf{x}_k, G_k, S_k, L_k, \mathbf{y}_{k+1}, \mathbf{z}_{k+1}] \leftarrow$ FLAGIteration$(G_{k-1}, S_{k-1}, \eta_{k-1}, \tilde{L}_{k-1}, \mathbf{y}_k, \mathbf{z}_k, \tilde{L}_k)$
10: accept $\leftarrow$ TRUE
11: **end if**
12: **end while**
13: **end while**

Algorithm 3 involves three main steps. Step 6 aims at guessing a viable value for L_k, which can be used at the present iteration. As depicted here and used in our numerical experiments, we have considered guessing L_k with some multiple of the known L_{k-1}. However, Step 6 can be replaced by any reasonable subroutine that tries to guess the valid ratio for $\tilde{L}_k$. Step 7 contains a subroutine, dubbed "A&V" (short for *Advance and Verify*), which computes $\mathbf{x}_k$, $\mathbf{y}_{k+1}$ and $\mathbf{z}_{k+1}$ using the guess $\tilde{L}_k$, and returns "accept=TRUE" if $\tilde{L}_k$ is a sufficiently good guess. Finally, Step 9 involves the "FlagIteration" subroutine, which, by reverting back to using "BinarySearch", computes $\mathbf{x}_k$, $\mathbf{y}_{k+1}$ and $\mathbf{z}_{k+1}$. This step is almost identical to one full iteration of FLAG in (1), though the statements are ordered differently. As shown in the proof of Theorem 3, the resulting updates generated from this step are always acceptable. In all of our numerical simulations, however, we have never observed FLARE performing Step 9. In fact, most often, the very first guess in Step 6 is deemed acceptable by Step 7 leading to FLARE requiring only one "**prox**" evaluation per iteration (as in FISTA).

The following result describes the main convergence properties of FLARE.

Theorem 2 (Convergence of FLARE) *Let Assumptions **A.1** and **A.2** hold and define* $D_\infty := \sup_{\mathbf{x},\mathbf{y} \in \mathcal{C}} \|\mathbf{x} - \mathbf{y}\|_\infty$. *For any* $\mathbf{u} \in \mathcal{C}$, *after* T *iterations of Algorithm 3, we get*

$$F(\mathbf{y}_{T+1}) - F(\mathbf{u}) \leq \mathcal{O}\left(\frac{\lambda \beta L D_\infty^2}{T^2}\right),$$

for some $\beta \in [1, d]$, *where* λ *is a constant specified in the input to Algorithm 3. Furthermore, each iteration takes time at most* $\mathcal{O}(\mathcal{T}_{prox} \cdot \log(dT^3))$, *where* $\mathcal{T}_{prox}$ *is the cost of evaluating* ***prox*** *in (3).*

Algorithm 4 A&V: Advance and Verify

Input: $G_{k-1}, S_{k-1}, \eta_{k-1}, \tilde{L}_{k-1}, \mathbf{y}_k, \mathbf{z}_k, \tilde{L}_k, \lambda$

$\eta_k \leftarrow \frac{1}{2\tilde{L}_k} + \sqrt{\frac{1}{4\tilde{L}_k^2} + \frac{\eta_{k-1}^2 \tilde{L}_{k-1}}{\tilde{L}_k}}$
$\mathbf{x}_k \leftarrow \left(1 - \frac{1}{\eta_k \tilde{L}_k}\right)\mathbf{y}_k + \frac{1}{\eta_k \tilde{L}_k}\mathbf{z}_k$
$\mathbf{y}_{k+1} \leftarrow \mathbf{prox}(\mathbf{x}_k)$
$\mathbf{p}_k \leftarrow -L(\mathbf{y}_{k+1} - \mathbf{x}_k)$
$\mathbf{g}_k \leftarrow \frac{\mathbf{p}_k}{\|\mathbf{p}_k\|_2}$
$G_k \leftarrow [G_{k-1}, \mathbf{g}_k]$
$\mathbf{s}_k(i) \leftarrow \|G_k(i,:)\|_2$
$S_k \leftarrow \text{diag}(\mathbf{s}_k) + \delta \mathbb{I}$
$L_k \leftarrow L\mathbf{g}_k^T S_k^{-1} \mathbf{g}_k$
$\mathbf{z}_{k+1} \leftarrow \arg\min_{\mathbf{z} \in \mathcal{C}} \langle \eta_k \mathbf{p}_k, \mathbf{z} - \mathbf{z}_k \rangle + \frac{1}{2}\|\mathbf{z} - \mathbf{z}_k\|_{S_k}^2$
accept $\leftarrow \tilde{L}_k \in [L_k, \lambda L_k]$

Algorithm 5 FlagIteration

Input: $G_{k-1}, S_{k-1}, \eta_{k-1}, \tilde{L}_{k-1}, \mathbf{y}_k, \mathbf{z}_k, \tilde{L}_k, \epsilon$

$\mathbf{x}_k \leftarrow \text{BinarySearch}(\mathbf{z}_k, \mathbf{y}_k, \epsilon)$
$\mathbf{y}_{k+1} \leftarrow \mathbf{prox}(\mathbf{x}_k)$
$\mathbf{p}_k \leftarrow -L(\mathbf{y}_{k+1} - \mathbf{x}_k)$
$\mathbf{g}_k \leftarrow \frac{\mathbf{p}_k}{\|\mathbf{p}_k\|_2}$
$G_k \leftarrow [G_{k-1}, \mathbf{g}_k]$
$\mathbf{s}_k(i) \leftarrow \|G_k(i,:)\|_2$
$S_k \leftarrow \text{diag}(\mathbf{s}_k) + \delta \mathbb{I}$
$L_k \leftarrow L\mathbf{g}_k^T S_k^{-1} \mathbf{g}_k$
$\tilde{L}_k \leftarrow L_k$
$\eta_k \leftarrow \frac{1}{2\tilde{L}_k} + \sqrt{\frac{1}{4\tilde{L}_k^2} + \frac{\eta_{k-1}^2 \tilde{L}_{k-1}}{\tilde{L}_k}}$
$\mathbf{z}_{k+1} \leftarrow \arg\min_{\mathbf{z} \in \mathcal{C}} \langle \eta_k \mathbf{p}_k, \mathbf{z} - \mathbf{z}_k \rangle + \frac{1}{2}\|\mathbf{z} - \mathbf{z}_k\|_{S_k}^2$

Note that by Theorem 2, the overall worst-case iteration complexity of FLARE (Algorithm 3), is similar to that of FLAG (Algorithm 1). This is mainly

due to the fact that, in worst case, Algorithm 3 can end up resorting to "BinarySearch" when repeated guessing fails. However, through extensive numerical experiments, we have observed that we rarely require more than one "**prox**" evaluation per iteration. In particular, Step 6 and 7 of Algorithm 3, most often, are only performed once, while Step 9 is never executed.

4 Numerical Experiments

We now numerically illustrate the performance of FLAG and FLARE in comparison to FISTA. We first consider comparing the performance of these algorithms with respect to the *total number of iterations*. Admittedly, "performance vs. iterations" is an unfair measure of comparing these algorithms. Indeed, each iteration of FLAG and FLARE can involve more "**prox**" evaluations than FISTA, and as noted in Section 2, in the worst case such "**prox**" evaluations can dominate the running time. Therefore, we subsequently evaluate these algorithms as measured by *total number of* ***prox*** *evaluations*, which is arguably more indicative of real world performance. In this light, we demonstrate that FLARE and FISTA perform favorably with respect to FLAG, with FLARE consistently outperforming the rest.

We compare FLAG, FLARE and FISTA on both regression and classification tasks. For regression experiments, we utilized squared loss $f(\mathbf{x}) = \frac{1}{2}\|A\mathbf{x} - \mathbf{b}\|_2^2$, where $A \in \mathbb{R}^{n\times d}$ and $b \in \mathbb{R}^n$ are, respectively, the data matrix and the response vector. For classification experiments, we employed a softmax classifier. In such a classifier, given C classes and a data point $\mathbf{a}$, the probability that $\mathbf{a}$ belongs to a class $c \in \{1, 2, \ldots, C\}$ is given as

$$\mathbf{Pr}\left(c|\mathbf{a}, \mathbf{x}_1, \ldots, \mathbf{x}_C\right) = \frac{e^{\langle \mathbf{a}, \mathbf{x}_c\rangle}}{\sum_{c'=1}^{C} e^{\langle \mathbf{a}, \mathbf{x}_{c'}\rangle}},$$

where $\mathbf{x}_c \in \mathbb{R}^p$ is the weight vector corresponding to class c. Recall that here there are only $C-1$ degrees of freedom, i.e., probabilities all must sum to one. Consequently, for a training data $\{\mathbf{a}_i, b_i\}_{i=1}^n \subset \mathbb{R}^p \times \{1, 2, \ldots, C\}$, the cross-entropy loss function for $\mathbf{x} = [\mathbf{x}_1; \mathbf{x}_2; \ldots; \mathbf{x}_{C-1}] \in \mathbb{R}^{(C-1)p}$ can be written as

$$\begin{aligned} f(\mathbf{x}) &\triangleq f(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{C-1}) \\ &= \sum_{i=1}^{n} \left(\log\left(1 + \sum_{c'=1}^{C-1} e^{\langle \mathbf{a}_i, \mathbf{x}_{c'}\rangle}\right) - \sum_{c=1}^{C-1} \mathbf{1}(b_i = c)\langle \mathbf{a}_i, \mathbf{x}_c\rangle \right). \end{aligned}$$

Note that here $d = (C-1)p$. It then follows that the gradient of f with respect to $\mathbf{x}_c$ is

$$\nabla_{\mathbf{x}_c} f(\mathbf{x}) = \sum_{i=1}^{n} \left(\frac{e^{\langle \mathbf{a}_i, \mathbf{x}_c\rangle}}{1 + \sum_{c'=1}^{C-1} e^{\langle \mathbf{a}_i, \mathbf{x}_{c'}\rangle}} - \mathbf{1}(b_i = c) \right) \mathbf{a}_i.$$

For each regression and classification formulation of $f(\mathbf{x})$, we consider two variants for $h(\mathbf{x})$ and $\mathcal{C}$: unconstrained ℓ_1 regularization, i.e., $h(\mathbf{x}) = \lambda\|\mathbf{x}\|_1$, $\mathcal{C} = \mathbb{R}^d$, as well as unregularized box-constrained as $h(\mathbf{x}) = 0$, $\mathcal{C} = \{\mathbf{x};\ \|\mathbf{x}\|_\infty \le c\}$, where λ and c are, respectively, the regularization parameter for ℓ_1 norm and the infinity ball radius. Recall that for regression, the former variant amounts to the celebrated Lasso [44]. In our experiments, we choose $\lambda = 0.1$ and $c = 1$. It is well-known that "**prox**" operator for ℓ_1 regularization is readily given via soft-thresholding [37], while in the case of box constraints, it involves the projection of the gradient step onto the infinity ball of the given radius. We tested regression and classification tasks on multiple real data sets. Tables 1 and 2, respectively, summarize the data sets used for these tasks.

Table 1: Data Sets for Classification Experiments. "#Test" indicates the size of the test set. "Var." refers to variants used for $h(\mathbf{x})$ and $\mathcal{C}$, i.e., "Box" for box-constrained and "ℓ_1" for ℓ_1-regularized variants, as mentioned in Section 4.

Name	`20 Newsgroups`	`HARUS`	`Gisette`	`Forest Covertype`
$\boldsymbol{n}$	10,142	7,767	6,000	435,759
#Test	1,127	3,162	1,000	145,253
$\boldsymbol{p}$	53,975	561	5,000	54
$\boldsymbol{C}$	20	12	2	7
Var.	Box	Box	ℓ_1	ℓ_1
Ref.	[27]	[15]	[22]	[11]

Table 2: Data Sets for Regression Experiments. "#Test" indicates the size of the test set. "Var." refers to variants used for $h(\mathbf{x})$ and $\mathcal{C}$, i.e., "Box" for box-constrained and "ℓ_1" for ℓ_1-regularized variants, as mentioned in Section 4.

Name	`Blog Feedback`	`Facebook CVD`
$\boldsymbol{n}$	47,157	36,854
#Test	5,240	4,095
$\boldsymbol{d}$	280	53
Var.	Box	ℓ_1
Ref.	[23]	[41]

We ran FLAG, FLARE, and FISTA for 1000 iterations each on softmax classification for the data sets enumerated in 1. Both variants of $h(\mathbf{x})$ and $\mathcal{C}$ are represented. The per iteration loss and test accuracy are displayed in Figures 1, 2, 3, and 4.

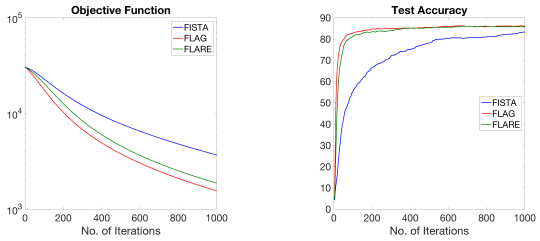

Figure 1: FLAG, FLARE, and FISTA on box-constrained classification for the `20 Newsgroups` data set.

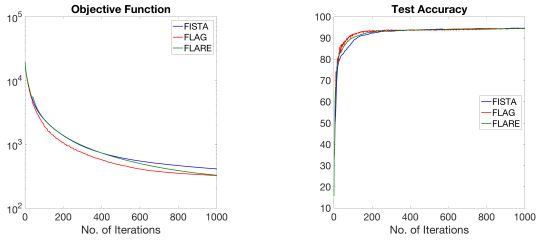

Figure 2: FLAG, FLARE, and FISTA on box-constrained classification for the `HARUS` data set.

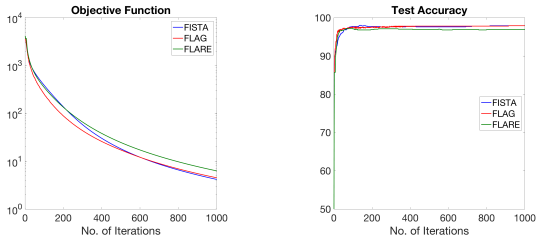

Figure 3: FLAG, FLARE, and FISTA on ℓ_1 regularized classification for the `Gisette` data set.

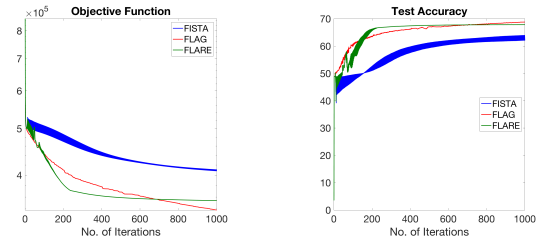

Figure 4: FLAG, FLARE, and FISTA on ℓ_1 regularized classification for the `Forest Covertype` data set.

We note that on the classification tasks, FLAG and FLARE perform as well as, or better than FISTA, as expected from the theoretical analysis. In particular, on the `20 Newsgroups` and `Forest Covertype` data sets, both FLAG and FLARE significantly outperform FISTA.

For the regression task we also ran FLAG, FLARE, and FISTA for 1000 iterations. The data sets used are enumerated in Table 2. The per iteration loss and test error are displayed in Figures 5 and 6.

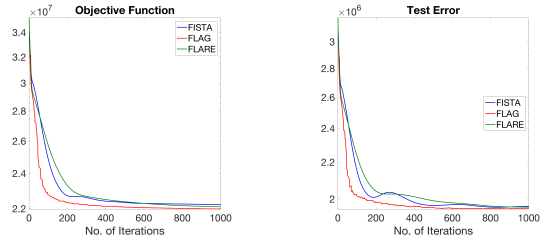

Figure 5: FLAG, FLARE, and FISTA on box-constrained regression for the `BlogFeedback` data set.

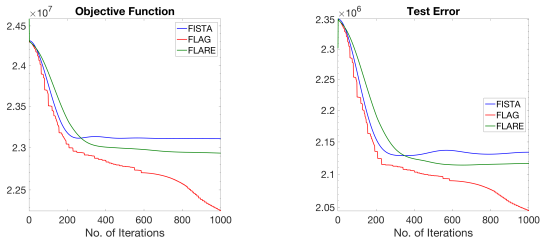

Figure 6: FLAG, FLARE, and FISTA on ℓ_1 regularized classification for the `Facebook CVD` data set.

Similarly to classification tasks, FLAG and FLARE perform as well as or superior to FISTA. Particularly on the `Facebook CVD` data set, FLAG significantly outperforms both FISTA and FLARE.

As previously noted, each iteration of FLAG and FLARE can involve more **prox** evaluations than FISTA, which can dominate the run time. Thus, comparing the performance of these methods as measured by the number of **prox** evaluations is more representative of real world cost than that measured by iterations. We thus repeat the above experiments with the exception that this time we ran each algorithm for 1000 **prox** evaluations and tracked the loss and test accuracy versus the number of **prox** evaluations. The results of these trials are displayed in Figures 7 – 12.

It can be seen that, as measured by the number of **prox** evaluations, FLARE and FISTA can outper-

form FLAG due to the possibly significant number of **prox** evaluations involved in FLAG's "BinarySearch", i.e., Step 11 of Algorithm 1. For all examples, FLARE performs at least as well as FISTA with FLARE outperforming all other algorithms on certain datasets, e.g., Figures 7 and 10. Empirically, after relaxing the "BinarySearch" in FLAG, FLARE continues to enjoy the performance advantages afforded by leveraging acceleration and adaptivity, while maintaining the low per-iteration cost of FISTA.

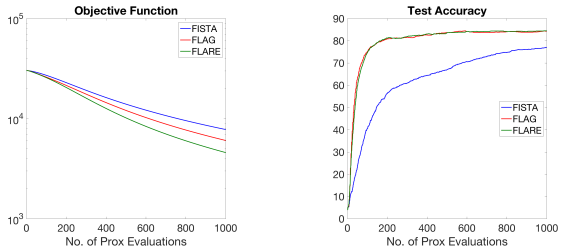

Figure 7: FLAG, FLARE, and FISTA on box-constrained classification for the `20 Newsgroups` data set.

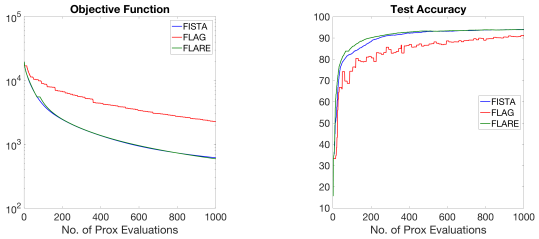

Figure 8: FLAG, FLARE, and FISTA on box-constrained classification for the `HARUS` data set.

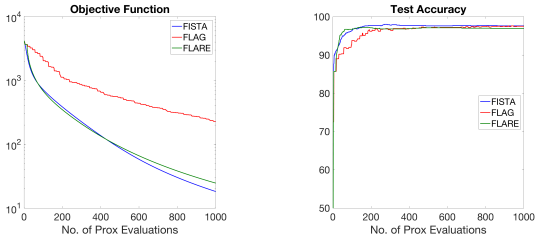

Figure 9: FLAG, FLARE, and FISTA on ℓ_1 regularized classification for the `Gisette` data set.

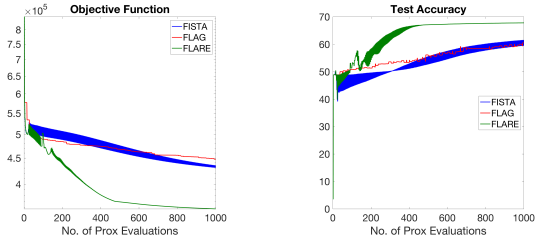

Figure 10: FLAG, FLARE, and FISTA on ℓ_1 regularized classification for the `Forest Covertype` data

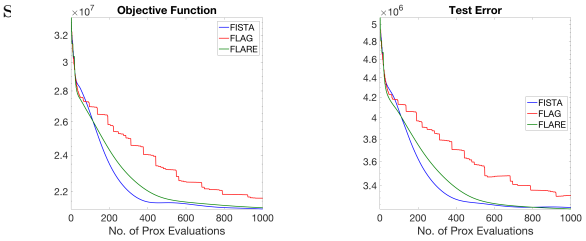

Figure 11: FLAG, FLARE, and FISTA on box-constrained regression for the `BlogFeedback` data

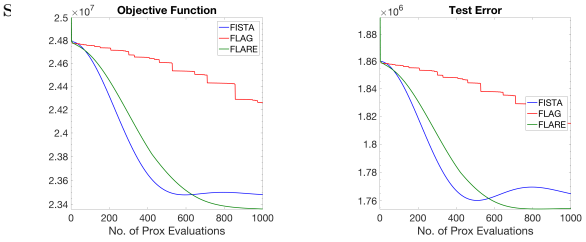

Figure 12: FLAG, FLARE, and FISTA on ℓ_1 regularized regression for the `Facebook CVD` data set.

5 Conclusions

Following the advantages of employing acceleration, e.g., Nesterov's scheme, as well as adaptivity, e.g., Adagrad, here, we considered algorithms that can offer the best of both worlds. Specifically, in the context of composite optimization problem, we theoretically as well as empirically studied FLAG and its relaxation, FLARE, which can achieve this by a particular linear coupling of a simple gradient step with that of a properly scaled mirror update.

We showed that FLAG and FLARE can be equivalently regarded as adaptive versions of FISTA or alternatively, as accelerated versions of AdaGrad. In other words, like Nesterov's accelerated algorithm and its proximal variant, FISTA, our methods achieve the optimal convergence rate of $\mathcal{O}(1/k^2)$ and like AdaGrad our methods adaptively choose a regularizer, in a way that performs almost as well as the best choice of regularizer in hindsight. These two effects contribute to the improved theoretical properties and empirical performance of FLAG and FLARE compared to alternatives, e.g., FISTA.

Acknowledgments

We gratefully acknowledge the support of the NSF through grant IIS-1619362, the Australian Research Council through an Australian Laureate Fellowship (FL110100281) and through the Australian Research Council Centre of Excellence for Mathematical and Statistical Frontiers (ACEMS), as well as grants from DARPA and ARO. The Forest Covertype is Copyrighted 1998 by Jock A. Blackard and Colorado State University.

References

[1] Zeyuan Allen-Zhu and Lorenzo Orecchia. Linear coupling: An ultimate unification of gradient and mirror descent. *arXiv preprint arXiv:1407.1537*, 2014.

[2] Uri M Ascher and Chen Greif. *A First Course on Numerical Methods*. SIAM, 2011.

[3] Adil Bagirov, Napsu Karmitsa, and Marko M Mäkelä. *Introduction to Nonsmooth Optimization: theory, practice and software*. Springer, 2014.

[4] Amir Beck and Marc Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM journal on imaging sciences*, 2(1):183–202, 2009.

[5] Dimitri P Bertsekas and Athena Scientific. *Convex optimization algorithms*. Athena Scientific Belmont, 2015.

[6] José M Bioucas-Dias and Mário AT Figueiredo. A new TwIST: Two-step iterative shrinkage/thresholding algorithms for image restoration. *IEEE Transactions on Image processing*, 16(12):2992–3004, 2007.

[7] Léon Bottou, Frank E Curtis, and Jorge Nocedal. Optimization methods for large-scale machine learning. *arXiv preprint arXiv:1606.04838*, 2016.

[8] Kristian Bredies and Dirk A Lorenz. Linear convergence of iterative soft-thresholding. *Journal of Fourier Analysis and Applications*, 14(5):813–837, 2008.

[9] Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

[10] Sébastien Bubeck, Yin Tat Lee, and Mohit Singh. A geometric alternative to Nesterov's accelerated gradient descent. *arXiv preprint arXiv:1506.08187*, 2015.

[11] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[12] Patrick L Combettes and Valérie R Wajs. Signal recovery by proximal forward-backward splitting. *Multiscale Modeling & Simulation*, 4(4):1168–1200, 2005.

[13] Ingrid Daubechies, Michel Defrise, and Christine De Mol. An iterative thresholding algorithm for linear inverse problems with a sparsity constraint. *Communications on pure and applied mathematics*, 57(11):1413–1457, 2004.

[14] Yann Dauphin, Harm de Vries, and Yoshua Bengio. Equilibrated adaptive learning rates for non-convex optimization. In *Advances in Neural Information Processing Systems*, pages 1504–1512, 2015.

[15] Luca Oneto Xavier Parra Davide Anguita, Alessandro Ghio and Jorge L. Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2013*, Bruges, Belgium, apr 2013.

[16] Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V Le, et al. Large scale distributed deep networks. In *Advances in neural information processing systems*, pages 1223–1231, 2012.

[17] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research*, 12:2121–2159, 2011.

[18] Michael Elad, Boaz Matalon, and Michael Zibulevsky. Coordinate and subspace optimization methods for linear least squares with non-quadratic regularization. *Applied and Computational Harmonic Analysis*, 23(3):346–367, 2007.

[19] Nicolas Flammarion and Francis Bach. From averaging to acceleration, there is only a step-size. In *Conference on Learning Theory*, pages 658–695, 2015.

[20] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The Elements of Statistical Learning*, volume 1. Springer series in statistics Springer, Berlin, 2001.

[21] Elad Hazan. Introduction to online convex optimization. *Foundations and Trends® in Optimization*, 2(3-4):157–325, 2016.

[22] Asa Ben-Hur Gideon Dror Isabelle Guyon, Steve R. Gunn. Result analysis of the nips 2003 feature selection challenge. In *NIPS*, Bruges, Belgium, 2004.

[23] Buza K. Feedback prediction for blogs. In *In Data Analysis, Machine Learning and Knowledge Discovery*, volume 14, pages 145–152. Springer International Publishing, 2014.

[24] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[25] Walid Krichene, Alexandre Bayen, and Peter L Bartlett. Accelerated mirror descent in continuous and discrete time. In *Advances in Neural Information Processing Systems*, pages 2827–2835, 2015.

[26] Brian Kulis. Metric learning: a survey. *Foundations and Trends in Machine Learning*, 5(4):287–364, 2012.

[27] Ken Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th international conference on machine learning*, volume 10, pages 331–339, 1995. Available at http://qwone.com/~jason/20Newsgroups/.

[28] Laurent Lessard, Benjamin Recht, and Andrew Packard. Analysis and design of optimization algorithms via integral quadratic constraints. *SIAM Journal on Optimization*, 26(1):57–95, 2016.

[29] A.S. Nemirovskiĭ and D.B. IUdin. *Problem Complexity and Method Efficiency in Optimization*. A Wiley-Interscience publication. Wiley, 1983.

[30] Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics, 1994.

[31] Yu Nesterov. Smooth minimization of non-smooth functions. *Mathematical programming*, 103(1):127–152, 2005.

[32] Yu Nesterov. Rounding of convex sets and efficient gradient methods for linear programming problems. *Optimisation Methods and Software*, 23(1):109–128, 2008.

[33] Yurii Nesterov. A method of solving a convex programming problem with convergence rate $\mathcal{O}(1/k^2)$. In *Soviet Mathematics Doklady*, volume 27, pages 372–376, 1983.

[34] Yurii Nesterov. *Introductory lectures on convex optimization*, volume 87. Springer Science & Business Media, 2004.

[35] Yurii Nesterov. Gradient methods for minimizing composite objective function, 2007. CORE report, available at http://www.ecore.be/DPs/dp 1191313936.pdf.

[36] Daniel P Palomar and Yonina C Eldar. *Convex optimization in signal processing and communications*. Cambridge university press, 2010.

[37] Neal Parikh and Stephen P Boyd. Proximal algorithms. *Foundations and Trends in optimization*, 1(3):127–239, 2014.

[38] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543, 2014.

[39] Farbod Roosta-Khorasani, Gábor J. Székely, and Uri Ascher. Assessing stochastic algorithms for large scale nonlinear least squares problems using extremal probabilities of linear combinations of gamma random variables. *SIAM/ASA Journal on Uncertainty Quantification*, 3(1):61–90, 2015.

[40] Farbod Roosta-Khorasani, Kees van den Doel, and Uri Ascher. Stochastic algorithms for inverse problems involving PDEs and many measurements. *SIAM J. Scientific Computing*, 36(5):S3–S22, 2014.

[41] Kamaljot Singh, Ranjeet Kaur Sandhu, and Dinesh Kumar. Comment volume prediction using neural networks and decision trees. In *IEEE UKSim-AMSS 17th International Conference on Computer Modelling and Simulation, UKSim2015 (UKSim2015)*, Cambridge, United Kingdom, mar 2015.

[42] Suvrit Sra, Sebastian Nowozin, and Stephen J Wright. *Optimization for Machine Learning*. Mit Press, 2012.

[43] Weijie Su, Stephen Boyd, and Emmanuel Candes. A differential equation for modeling Nesterovs accelerated gradient method: Theory and insights. In *Advances in Neural Information Processing Systems*, pages 2510–2518, 2014.

[44] Robert Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[45] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4, 2012.

[46] Paul Tseng. On accelerated proximal gradient methods for convex-concave optimization. *SIAM Journal on Optimization*, 2008.

[47] Andre Wibisono, Ashia C Wilson, and Michael I Jordan. A variational perspective on accelerated methods in optimization. *arXiv preprint arXiv:1603.04245*, 2016.

[48] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.