Curvature-Exploiting Acceleration of Elastic Net Computations

Vien V. Mai [1] **Mikael Johansson** [1]

Abstract

This paper introduces an efficient second-order method for solving the elastic net problem. Its key innovation is a computationally efficient technique for injecting curvature information in the optimization process which admits a strong theoretical performance guarantee. In particular, we show improved run time over popular first-order methods and quantify the speed-up in terms of statistical measures of the data matrix. The improved time complexity is the result of an extensive exploitation of the problem structure and a careful combination of second-order information, variance reduction techniques, and momentum acceleration. Beside theoretical speed-up, experimental results demonstrate great practical performance benefits of curvature information, especially for ill-conditioned data sets.

1. Introduction

Lasso, ridge and elastic net regression are fundamental problems in statistics and machine learning, with countless applications in science and engineering (Zou & Hastie, 2005). Elastic net regression amounts to solving the following convex optimization problem

$$\underset{\boldsymbol{x}\in\mathbb{R}^d}{\text{minimize}} \left\{ \frac{1}{2n} \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2^2 + \frac{\gamma_2}{2}\|\boldsymbol{x}\|_2^2 + \gamma_1 \|\boldsymbol{x}\|_1 \right\}, \quad (1)$$

for given data matrices $\boldsymbol{A} \in \mathbb{R}^{n\times d}$ and $\boldsymbol{b} \in \mathbb{R}^n$ and regularization parameters γ_1 and γ_2. Setting $\gamma_1 = 0$ results in ridge regression, $\gamma_2 = 0$ yields lasso and letting $\gamma_1 = \gamma_2 = 0$ reduces the problem to the classical least-squares. Lasso promotes sparsity of the optimal solution, which sometimes helps to improve interpretability of the results. Adding the additional l_2-regularizer helps to improve the performance when features are highly correlated (Tibshirani et al., 2015; Zou & Hastie, 2005).

The convergence rates of iterative methods for solving (1) are typically governed by the condition number of the Hessian matrix of the ridge loss, $\boldsymbol{C}+\gamma_2\boldsymbol{I}$, where $\boldsymbol{C} = \frac{1}{n}\boldsymbol{A}^\top\boldsymbol{A}$ is the sample correlation matrix. Real-world data sets often have few dominant features, while the other features are highly correlated with the stronger ones (Gonen et al., 2016; Tibshirani et al., 2015). This translates to a rapidly decaying spectrum of $\boldsymbol{C}$. In this paper, we demonstrate how this property can be exploited to reduce the effect of ill-conditioning and to design faster algorithms for solving the elastic net regression problem (1).

1.1. Related work

Over the past few years, there has been a great attention to developing efficient optimization algorithms for minimizing composite objective functions

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} F(\boldsymbol{x}) \triangleq f(\boldsymbol{x}) + h(\boldsymbol{x}), \quad (2)$$

where $f(\boldsymbol{x}) = \frac{1}{n}\sum_{i=1}^n f_i(\boldsymbol{x})$ is a finite sum of smooth and convex component functions $f_i(\boldsymbol{x})$, and $h(\boldsymbol{x})$ is a possibly non-smooth convex regularizer. In machine learning applications, the function f typically models the empirical data loss and the regularizer h is used to promote desired properties of a solution. For example, the elastic net objective can fit to this form with $f_i(\boldsymbol{x}) = \frac{1}{2}\left(\boldsymbol{a}_i^\top\boldsymbol{x}-\boldsymbol{b}_i\right)^2 + \gamma_2\|\boldsymbol{x}\|_2^2$, and $h(\boldsymbol{x}) = \gamma_1\|\boldsymbol{x}\|_1$.

1.1.1. First-Order methods

Standard deterministic first-order methods for solving (2), such as proximal gradient descent, enjoy linear convergence for strongly convex objective functions and are able to find an ϵ-approximate solution in time $O\left(dn\kappa\log\frac{1}{\epsilon}\right)$, where κ is the condition number of f. This runtime can be improved to $O\left(dn\sqrt{\kappa}\log\frac{1}{\epsilon}\right)$ if it is combined with Nesterov acceleration (Beck & Teboulle, 2009; Nesterov, 2013). However, the main drawback of these methods is that they need to access the whole data set in every iteration, which is too costly in many machine learning tasks.

For large-scale problems, methods based on stochastic gradients have become the standard choice for solv-

[1]Department of Department of Automatic Control, School of Electrical Engineering and Computer Science, Royal Institute of Technology (KTH), Stockholm, Sweden. Correspondence to: V. V. Mai <maivv@kth.se>, M. Johansson <mikaelj@kth.se>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

ing (2). Many linearly convergent proximal methods such as, SAGA (Defazio et al., 2014) and Prox-SVRG (Xiao & Zhang, 2014), have been introduced and shown to outperform standard first-order methods under certain regularity assumptions. These methods improve the time complexity to $O\left(d\,(n+\tilde{\kappa})\log\frac{1}{\epsilon}\right)$, where $\tilde{\kappa}$ is a condition number satisfying $\tilde{\kappa} \geq \kappa$. When the component functions do not vary substantially in smoothness, $\tilde{\kappa} \approx \kappa$, and this complexity is far better than those of deterministic methods above. By exploiting Nesterov momentum in different ways (see, e.g., (Allen-Zhu, 2016; Defazio, 2016; Frostig et al., 2015; Lin et al., 2015)), one can improve the complexity to $O\big(d\big(n+\sqrt{n\tilde{\kappa}}\big)\log\frac{1}{\epsilon}\big)$, which is also optimal for this class of problems (Woodworth & Srebro, 2016).

1.1.2. Second-order methods

Second-order methods are known to have superior performance compared to their first-order counterparts both in theory and practice, especially when the problem at hand is highly nonlinear and/or ill-conditioned. However, such methods often have very high computational cost per iteration. Recently, there has been an intense effort to develop algorithms which use second-order information with a more reasonable computational burden (see, e.g., (Agarwal et al., 2017; Byrd et al., 2016; Erdogdu & Montanari, 2015; Moritz et al., 2016; Roosta-Khorasani & Mahoney, 2016; Xiao & Zhang, 2014; Xu et al., 2016) and references therein). Those methods use techniques such as random sketching, matrix sampling, and iterative estimation to construct an approximate Hessian matrix. Local and global convergence guarantees have been derived under various assumptions. Although many experimental results have shown excellent performance of those methods on many machine learning tasks, current second-order methods for finite-sum optimization tend to have much higher time-complexities than their first-order counterparts (see (Xu et al., 2016) for a detailed comparison).

Apart from having high time complexities, none of the methods cited above have any guarantees in the composite setting since their analyses hinge on differentiability of the objective function. Instead, one has to rely on methods that build on *proximal Newton* updates (see, e.g., (Ghanbari & Scheinberg, 2016; Lee et al., 2012; Liu et al., 2017; Rodomanov & Kropotov, 2016)). However, these methods still inherit the high update and storage costs of conventional second-order methods or require elaborate tuning of several parameters and stopping criteria depending on a phase transition which occurs in the algorithm.

1.1.3. Ridge regression

For the smooth ridge regression problem, the authors in (Gonen et al., 2016) have developed a preconditioning method based on linear sketching which, when coupled with SVRG, yields a significant speed-up over stochastic first-order methods. This is a rare second-order method that has a comparable or even better time complexity than stochastic first-order methods. More precisely, it has a guaranteed running time of $O(d(n+\kappa_{\boldsymbol{H}})\log\frac{1}{\epsilon})$, where $\kappa_{\boldsymbol{H}}$ is a new condition number that can be dramatically smaller than $\tilde{\kappa}$, especially when the spectrum of $\boldsymbol{C}$ decays rapidly. When $d \ll n$, the authors in (Wang & Zhang, 2017) combine sub-sampled Newton methods with the mini-batch SVRG to obtain some further improvements.

1.2. Contributions

Recently, the work (Arjevani & Shamir, 2017) shows that under some mild algorithmic assumptions, and if the dimension is sufficiently large, the iteration complexity of second-order methods for *smooth* finite-sum problems composed of quadratics is no better than first-order methods. Therefore, it is natural to ask whether one can develop a second-order method for solving the elastic net problem which has improved practical performance but still enjoys a strong *worst-case* time complexity like the stochastic first-order methods do? It should be emphasized that due to the non-smooth objective, achieving this goal is much more challenging than for ridge regression. The preconditioning approach in (Gonen et al., 2016) is not applicable, and the current theoretical results for second-order methods are not likely to offer the desired running time.

In this paper, we provide a positive answer to this question. Our main contribution is the design and analysis of a simple second-order method for the elastic net problem which has a strong theoretical time complexity and superior practical performance. The convergence bound adapts to the problem structure and is governed by the spectrum and a *statistical measure* of the data matrix. These quantities often yield significantly stronger time complexity guarantees for practical datasets than those of stochastic first-order methods (see Table 1). To achieve this, we first leverage recent advances in randomized low-rank approximation to generate a simple, one-shot approximation of the Hessian matrix. We then exploit the composite and finite-sum structure of the problem to develop a variant of the ProxSVRG method that builds upon Nesterov's momentum acceleration and inexact computations of *scaled proximal* operators, which may be of independent interest. We provide a simple convergence proof based on an explicit Lyapunov function, thus avoiding the use of sophisticated *stochastic estimate sequences*.

2. Preliminaries and Notation

Vectors are indicated by bold lower-case letters, and matrices are denoted by bold upper-case letters. We denote

Table 1. Summary of different algorithms solving the elastic net problem. Here, κ and $\tilde{\kappa}$ are conventional condition numbers satisfying $\kappa \leq \tilde{\kappa}$, while $\kappa_{\boldsymbol{H}}$ is a new condition number defined w.r.t the $\boldsymbol{H}$-norm. When $\boldsymbol{H}$ is an approximate Hessian of the ridge loss, $\kappa_{\boldsymbol{H}}$ is often much smaller than $\tilde{\kappa}$, especially on practical data sets.

Algorithm	Time complexity	2nd-order
PGD	$O(dn\kappa \log \frac{1}{\epsilon})$	no
FISTA	$O\left(dn\sqrt{\kappa} \log \frac{1}{\epsilon}\right)$	no
ProxSVRG	$O\left(d\left(n+\tilde{\kappa}\right) \log \frac{1}{\epsilon}\right)$	no
Katyusha	$O\left(d(n+\sqrt{n\tilde{\kappa}}) \log \frac{1}{\epsilon}\right)$	no
Ours	$O\left(d\left(n+\kappa_{\boldsymbol{H}}\right) \log \frac{1}{\epsilon}\right)$	yes

the dot product between $\boldsymbol{x}$ and $\boldsymbol{y}$ by $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{x}^\top \boldsymbol{y}$, and the Euclidean norm of $\boldsymbol{x}$ by $\|\boldsymbol{x}\|_2 = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle}$. For a symmetric positive definite matrix $\boldsymbol{H}$, $\langle \boldsymbol{x}, \boldsymbol{y} \rangle_{\boldsymbol{H}} = \boldsymbol{x}^\top \boldsymbol{H} \boldsymbol{y}$ is the $\boldsymbol{H}$-inner product of two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ and $\|\boldsymbol{x}\|_{\boldsymbol{H}} = \sqrt{\langle \boldsymbol{x}, \boldsymbol{x} \rangle_{\boldsymbol{H}}}$ is the Mahalanobis norm of $\boldsymbol{x}$. We denote by $\lambda_i(\boldsymbol{A})$ the ith largest eigenvalue of $\boldsymbol{A}$. Finally, λ_i denotes the ith largest eigenvalue of the correlation matrix $\boldsymbol{C}$.

In the paper, we shall frequently use the notions of strong convexity and smoothness in the $\boldsymbol{H}$-norm, introduced in the next two assumptions.

Assumption 1. *The function $h(\boldsymbol{x})$ is lower-semicontinous and convex and $\operatorname{dom} h := \{\boldsymbol{x} \in \mathbb{R}^d \,|\, h(\boldsymbol{x}) < \infty\}$, is closed. Each function f_i is L_i-smooth w.r.t the $\boldsymbol{H}$-norm, i.e, there exists a positive constant L_i such that*

$$\|\nabla f_i(\boldsymbol{x}) - \nabla f_i(\boldsymbol{y})\|_{\boldsymbol{H}^{-1}} \leq L_i \|\boldsymbol{x} - \boldsymbol{y}\|_{\boldsymbol{H}}, \ \forall \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d.$$

Assumption 1 implies that ∇f is L-Lipschitz:

$$\|\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})\|_{\boldsymbol{H}^{-1}} \leq L \|\boldsymbol{x} - \boldsymbol{y}\|_{\boldsymbol{H}}$$

for some $L \leq L_{\text{avg}} = \frac{1}{n}\sum_{i=1}^n L_i$. As a consequence, we have the following bound:

$$f(\boldsymbol{y}) \leq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \frac{L_{\text{avg}}}{2} \|\boldsymbol{y} - \boldsymbol{x}\|_{\boldsymbol{H}}^2.$$

Assumption 2. *The function $f(\boldsymbol{x})$ is μ-strongly convex w.r.t the $\boldsymbol{H}$-norm, i.e, there exists a positive constant μ such that*

$$f(\lambda \boldsymbol{x} + (1-\lambda)\boldsymbol{y}) \leq \lambda f(\boldsymbol{x}) + (1+\lambda) f(\boldsymbol{y}) - \frac{\mu \lambda (1-\lambda)}{2} \|\boldsymbol{x} - \boldsymbol{y}\|_{\boldsymbol{H}}^2$$

holds for all $x, y \in \mathbb{R}^n$ and $\lambda \in [0, 1]$.

Assumption 2 is equivalent to the requirement that

$$f(\boldsymbol{y}) \geq f(\boldsymbol{x}) + \langle \nabla f(\boldsymbol{x}), \boldsymbol{y} - \boldsymbol{x} \rangle + \frac{\mu}{2} \|\boldsymbol{y} - \boldsymbol{x}\|_{\boldsymbol{H}}^2,$$

holds for all $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$. We will use both of these definitions of strong convexity in our proofs.

At the core of our method is the concept of *scaled proximal mappings*, defined as follows:

Definition 1 (Scaled Proximal Mapping)**.** *For a convex function h and a symmetric positive definite matrix $\boldsymbol{H}$, the scaled proximal mapping of h at $\boldsymbol{x}$ is*

$$\operatorname{prox}_h^{\boldsymbol{H}}(\boldsymbol{y}) = \operatorname*{argmin}_{\boldsymbol{x} \in \mathbb{R}^d} \left\{ h(\boldsymbol{x}) + \frac{1}{2} \|\boldsymbol{x} - \boldsymbol{y}\|_{\boldsymbol{H}}^2 \right\}. \quad (3)$$

The scaled proximal mappings generalize the conventional ones:

$$\operatorname{prox}_h(\boldsymbol{y}) = \operatorname*{argmin}_{\boldsymbol{x} \in \mathbb{R}^d} \left\{ h(\boldsymbol{x}) + \frac{1}{2} \|\boldsymbol{x} - \boldsymbol{y}\|_2^2 \right\}. \quad (4)$$

However, while many conventional prox-mappings admit analytical solutions, this is almost never the case for scaled proximal mappings. This makes it hard to extend efficient first-order proximal methods to second-order ones. Fortunately, scaled proximal mappings do share some key properties with the conventional ones. We collect a few of them in the following result:

Property 1 ((Lee et al., 2012))**.** *The following properties hold:*

1. *$\operatorname{prox}_h^{\boldsymbol{H}}(\boldsymbol{x})$ exists and is unique for $\boldsymbol{x} \in \operatorname{dom} h$.*
2. *Let $\partial h(\boldsymbol{x})$ be the subdifferential of h at $\boldsymbol{x}$, then*
$$\boldsymbol{H}\left(\boldsymbol{x} - \operatorname{prox}_h^{\boldsymbol{H}}(\boldsymbol{x})\right) \in \partial h\left(\operatorname{prox}_h^{\boldsymbol{H}}(\boldsymbol{x})\right).$$
3. *$\operatorname{prox}_h^{\boldsymbol{H}}(\cdot)$ is non-expansive in the $\boldsymbol{H}$-norm:*

$$\left\|\operatorname{prox}_h^{\boldsymbol{H}}(\boldsymbol{x}) - \operatorname{prox}_h^{\boldsymbol{H}}(\boldsymbol{y})\right\|_{\boldsymbol{H}} \leq \|\boldsymbol{x} - \boldsymbol{y}\|_{\boldsymbol{H}} \ \ \forall \boldsymbol{x}, \boldsymbol{y} \in \operatorname{dom} h.$$

Finally, in our algorithm, it will be enough to solve (3) approximately in the following sense:

Definition 2 (Inexact subproblem solutions)**.** *We say that $\boldsymbol{x}^+ \in \mathbb{R}^d$ is an ϵ-optimal solution to (3) if*

$$h\left(\boldsymbol{x}^+\right) + \frac{1}{2\eta} \left\|\boldsymbol{x}^+ - \boldsymbol{y}\right\|_{\boldsymbol{H}}^2 \leq \min_{\boldsymbol{x} \in \mathbb{R}^d} \left\{ h(\boldsymbol{x}) + \frac{1}{2\eta} \|\boldsymbol{x} - \boldsymbol{y}\|_{\boldsymbol{H}}^2 \right\} + \epsilon. \quad (5)$$

3. Building Block 1: Randomized Low-Rank Approximation

The computational cost of many Newton-type methods is dominated by the time required to compute the update direction $\boldsymbol{d} = \boldsymbol{H}^{-1}\boldsymbol{g}$ for some vector $\boldsymbol{g} \in \mathbb{R}^d$ and approximate Hessian $\boldsymbol{H}$. A naive implementation using SVD would take $O\left(nd^2\right)$ flops, which is prohibitive for large-scale data sets. A natural way to reduce this cost is to use

truncated SVD. However, standard deterministic methods such as the power method and the Lanzcos method have run times that scale inversely with the gap between the eigenvalues of the input matrix. This gap can be arbitrarily small for practical data sets, thereby preventing us from obtaining the desired time complexity. In contrast, randomized sketching schemes usually admit gap-free run times (Halko et al., 2011). However, unlike other methods, the block Lanczos method, detailed in Algorithm 1, admits both fast run times and strong guarantees on the errors between the true and the computed approximate singular vectors. This property turns out to be critical for deriving bounds on the condition number of the elastic net.

Proposition 1 ((Musco & Musco, 2015)). *Assume that $\boldsymbol{U}_r$, $\boldsymbol{\Sigma}_r$, and $\boldsymbol{V}_r$ are matrices generated by Algorithm 1. Let $\boldsymbol{A}_r = \boldsymbol{U}_r\boldsymbol{\Sigma}_r\boldsymbol{V}_r^\top = \sum_{i=1}^r \sigma_i \boldsymbol{u}_i\boldsymbol{v}_i^\top$ and let $\boldsymbol{A} = \sum_{i=1}^d \bar{\sigma}_i\bar{\boldsymbol{u}}_i\bar{\boldsymbol{v}}_i^\top$ be the SVD of $\boldsymbol{A}$. Then, the following bounds hold with probability at least $9/10$:*

$$\|\boldsymbol{A} - \boldsymbol{A}_r\|_2 \leq (1+\epsilon')\,\bar{\sigma}_k$$
$$\left|\boldsymbol{u}_i^\top \boldsymbol{A}\boldsymbol{A}^\top \boldsymbol{u}_i - \bar{\boldsymbol{u}}_i^\top \boldsymbol{A}\boldsymbol{A}^\top \bar{\boldsymbol{u}}_i\right| \leq \epsilon'\bar{\sigma}_{r+1}^2, \quad \forall i \in \{1,\ldots,r\}.$$

The total running time is $O\left(ndr\log d(\epsilon')^{-1/2}\right)$.

Note that we only run Algorithm 1 once and $\epsilon' = 1/2$ is sufficient in our work. Thus, the computational cost of this step is negligible, in theory and in practice.

3.1. Aproximating the Hessian

In this work, we consider the following approximate Hessian matrix of the ridge loss:

$$\boldsymbol{H} = \boldsymbol{V}_r\left(\boldsymbol{\Sigma}_r^2 + \gamma_2\boldsymbol{I}\right)\boldsymbol{V}_r^\top + \left(\sigma_r^2+\gamma_2\right)\left(\boldsymbol{I} - \boldsymbol{V}_r\boldsymbol{V}_r^\top\right). \quad (6)$$

Here, the first term is a natural rank r approximation of the true Hessian, while the second term is used to capture information in the subspace orthogonal to the column space of $\boldsymbol{V}_r$. The inverse of $\boldsymbol{H}$ in (6) admits the explicit expression

$$\boldsymbol{H}^{-1} = \boldsymbol{V}_r\left(\boldsymbol{\Sigma}_r^2 + \gamma_2\boldsymbol{I}\right)^{-1}\boldsymbol{V}_r^\top + \frac{1}{\sigma_r^2+\gamma_2}\left(\boldsymbol{I} - \boldsymbol{V}_r\boldsymbol{V}_r^\top\right),$$

so the evaluation of $\boldsymbol{H}^{-1}\boldsymbol{x}$ has time complexity $O\left(rd\right)$.

3.2. Bounding the Condition Number

We now turn our attention to studying how the approximate Hessian affects the relevant condition number of the elastic net problem. We first introduce a condition number that usually determines the iteration complexity of stochastic first-order methods under non-uniform sampling.

Definition 3. *The average condition number of (1.1) is*

$$\kappa_{\boldsymbol{H}} = \frac{L_{\text{avg}}}{\mu} = \frac{\frac{1}{n}\sum_{i=1}^n L_i}{\mu}.$$

Algorithm 1 Randomized Block Lanczos Method (Musco & Musco, 2015)

Input: Data matrix $\boldsymbol{A} \in \mathbb{R}^{n\times d}$, target rank r, target precision $\epsilon' \in (0,1)$

1: Let $q = O(\log d/\sqrt{\epsilon'})$, and draw $\boldsymbol{\Pi} \sim \mathcal{N}_{d\times r}\left(\boldsymbol{0}, \boldsymbol{I}\right)$
2: Compute $\boldsymbol{K} = \left[\boldsymbol{A}\boldsymbol{\Pi}\ \ \left(\boldsymbol{A}\boldsymbol{A}^\top\right)\boldsymbol{A}\boldsymbol{\Pi}\ \ \ldots\ \ \left(\boldsymbol{A}\boldsymbol{A}^\top\right)^q\boldsymbol{A}\boldsymbol{\Pi}\right]$
3: Orthonormalize columns of $\boldsymbol{K}$ to obtain $\boldsymbol{Q}$
4: Compute truncated r-SVD of $\boldsymbol{Q}^\top\boldsymbol{A}$ as $\boldsymbol{W}_r\boldsymbol{\Sigma}_r\boldsymbol{V}_r^\top$
5: Compute $\boldsymbol{U}_r = \boldsymbol{Q}\boldsymbol{W}_r$

Output: $\boldsymbol{U}_r, \boldsymbol{\Sigma}_r, \boldsymbol{V}_r$

For the elastic net problem (1), the smooth part of the objective is the ridge loss

$$\frac{1}{n}\sum_{i=1}^n \underbrace{\frac{1}{2}\left(\boldsymbol{a}_i^\top\boldsymbol{x} - \boldsymbol{b}_i\right)^2 + \gamma_2\|\boldsymbol{x}\|_2^2}_{f_i(\boldsymbol{x})}.$$

Since we define smoothness and strong convexity of $f_i(\boldsymbol{x})$ in the $\boldsymbol{H}$-norm, the relevant constants are

$$L_i = \|\boldsymbol{H}^{-1}\left(\boldsymbol{a}_i\boldsymbol{a}_i^\top + \gamma_2\boldsymbol{I}\right)\|_2$$
$$\mu = \lambda_d\left(\boldsymbol{H}^{-1/2}\left(\boldsymbol{C} + \gamma_2\boldsymbol{I}\right)\boldsymbol{H}^{-1/2}\right).$$

For comparison, we also define the conventional condition number $\tilde{\kappa}$, which characterizes the smoothness and strong convexity of $f_i(\boldsymbol{x})$ in the Euclidean norm. In this case $\tilde{\kappa} = \sum_i L_i/(n\mu)$, where

$$L_i = \|\boldsymbol{a}_i\boldsymbol{a}_i^\top + \gamma_2\boldsymbol{I}\|_2^2 \quad \text{and} \quad \mu = \lambda_d\left(\boldsymbol{C} + \gamma_2\boldsymbol{I}\right).$$

It will become apparent that $\kappa_{\boldsymbol{H}}$ can be expressed in terms of a statistical measure of the ridge loss and that it may be significantly smaller than $\tilde{\kappa}$. We start by introducing a statistical measure that has been widely used in the analysis of ridge regression (see, e.g., (Hsu et al., 2012) and the references therein).

Definition 4 (Effective Dimension). *For a positive constant λ, the effective dimension of $\boldsymbol{C}$ is defined as*

$$\mathsf{d}_\lambda = \sum_{i=1}^d \frac{\lambda_i}{\lambda_i + \lambda}.$$

The effective dimension generalizes the ordinary dimension and satisfies $\mathsf{d}_\lambda \leq d$ with equality if and only if $\lambda = 0$. It is typical that when $\boldsymbol{C}$ has a rapidly decaying spectrum, most of the λ_i's are dominated by λ, and hence d_λ can be much smaller than d.

The following lemma bounds the eigenvalues of the matrix $\boldsymbol{H}^{-1/2}\left(\boldsymbol{C} + \gamma_2\boldsymbol{I}\right)\boldsymbol{H}^{-1/2}$, which can be seen as the effective Hessian matrix.

Lemma 1 ((Gonen et al., 2016)). *Invoking Algorithm 1 with data matrix $\frac{1}{\sqrt{n}}\boldsymbol{A}$, target rank r, and target precision $\epsilon' = 1/2$, it holds with probability at least $9/10$ that*

$$\lambda_1\left(\boldsymbol{H}^{-1/2}\left(\boldsymbol{C}+\gamma_2\boldsymbol{I}\right)\boldsymbol{H}^{-1/2}\right) \leq 17$$
$$\frac{\gamma_2}{19\left(\lambda_r+\gamma_2\right)} \leq \lambda_d\left(\boldsymbol{H}^{-1/2}\left(\boldsymbol{C}+\gamma_2\boldsymbol{I}\right)\boldsymbol{H}^{-1/2}\right) \leq 2.$$

Equipped with Lemma 1, we can now connect $\kappa_{\boldsymbol{H}}$ with d_λ using the following result.

Theorem 1. *With probability at least $9/10$, the following bound holds up to a multiplicative constant:*

$$\kappa_{\boldsymbol{H}} \leq \min\left(\frac{\mathsf{d}_{\gamma_2}}{\gamma_2}, \frac{r\lambda_r+\sum_{i>r}\lambda_i}{\gamma_2}+d\right).$$

Proof. See Appendix B. □

Since $\tilde{\kappa} = (\sum_i \lambda_i + d\gamma_2)/\gamma_2$, $\kappa_{\boldsymbol{H}}$ is reduced by a factor

$$\frac{\sum_{i\leq r}\lambda_i+\sum_{i>r}\lambda_i}{r\lambda_r+\sum_{i>r}\lambda_i},$$

compared to $\tilde{\kappa}$. If the spectrum of $\boldsymbol{C}$ decays rapidly, then the terms $\sum_{i>r}\lambda_i$ are negligible and the ratio is approximately $\sum_{i\leq r}\lambda_i/(r\lambda_r)$. If the first eigenvalues are much larger than λ_r, this ratio will be large. For example, for the australian data set (Chang & Lin, 2011), this ratio can be as large as 1.34×10^4 and 1.6×10^5 for $r=3$ and $r=4$, respectively. This indicates that it is possible to improve the iteration complexity of stochastic first-order methods if one can capitalize on the notions of strong convexity and smoothness w.r.t the $\boldsymbol{H}$-norm in the optimization algorithm. Of course, this is only meaningful if there is an efficient way to inject curvature information into the optimization process without significantly increasing the computational cost. In the smooth case, i.e., $\gamma_1 = 0$, this task can be done by a preconditioning step (Gonen et al., 2016). However, this approach is not applicable for the elastic net, and we need to make use of another building block.

4. Building Block 2: Inexact Accelerated Scaled Proximal SVRG

In this section, we introduce an inexact scaled accelerated ProxSVRG algorithm for solving the generic finite-sum minimization problem in (2). We then characterize the convergence rate of the proposed algorithm.

4.1. Description of the Algorithm

To motivate our algorithm, we first recall the ProxSVRG method from (Xiao & Zhang, 2014): For the sth outer iteration with the corresponding outer iterate $\tilde{\boldsymbol{x}}_s$, let $\boldsymbol{x}_0 = \tilde{\boldsymbol{x}}_s$ and for $k = 0, 2, \ldots, T-1$ do

$$\boldsymbol{v}_k = \left(\nabla f_{i_k}(\boldsymbol{x}_k) - \nabla f_{i_k}(\tilde{\boldsymbol{x}}_s)\right)/\left(np_{i_k}\right) + \nabla f(\tilde{\boldsymbol{x}}_s) \quad (7)$$
$$\boldsymbol{x}_{k+1} = \mathrm{prox}_{\eta h}\left(\boldsymbol{x}_k - \eta\boldsymbol{v}_k\right), \quad (8)$$

where i_k is drawn randomly from $\{1,\ldots,n\}$ with probability $p_{i_k} = L_{i_k}/(nL_{\mathrm{avg}})$. Since we are provided with an approximate Hessian matrix $\boldsymbol{H}$, it is natural to use the following update:

$$\boldsymbol{x}_{k+1} = \mathrm{prox}^{\boldsymbol{H}}_{\eta h}\left(\boldsymbol{x}_k - \eta\boldsymbol{H}^{-1}\boldsymbol{v}_k\right), \quad (9)$$

which can be seen as a proximal Newton step with the full gradient vector replaced by $\boldsymbol{v}_k$. Note that when $h(\cdot)$ is the ℓ_1-penalty, ProxSVRG can evaluate (8) in time $O(d)$, while evaluating (9) amounts to solving an optimization problem. It is thus is critical to keep the number of such evaluations small, which then translates into making a sufficient progress at each iteration. A natural way to achieve this goal is to reduce the variance of the noisy gradient $\boldsymbol{v}_k$. This suggests to use large mini-batches, i.e., instead of using a single component function f_{i_k}, we use multiple ones to form:

$$\boldsymbol{v}_k = \frac{1}{b}\sum_{i_k\in\mathcal{B}_k}\left(\nabla f_{i_k}(\boldsymbol{x}_k) - \nabla f_{i_k}(\tilde{\boldsymbol{x}}_s)\right)/(np_{i_k}) + \nabla f(\tilde{\boldsymbol{x}}_s),$$

where $\mathcal{B}_k \subset \{1,\ldots,n\}$ is a set of indices with cardinality $|\mathcal{B}_k| = b$. It is easy to verify that $\boldsymbol{v}_k$ is an unbiased estimate of $\nabla f(\boldsymbol{x}_k)$. Notice that naively increasing the batch size makes the algorithm increasingly similar to its deterministic counterpart, hence inheriting a high-time complexity. This makes it hard to retain the runtime of ProxSVRG in the presence of 2nd-order information.

In the absence of second-order information and under the assumption that the proximal step is computed exactly, the work (Nitanda, 2014) introduced a method called AccProxSVRG that enjoys the same time complexity as ProxSVRG but allows for much larger mini-batch sizes. In fact, it can tolerate a mini-batch of size $O\big(\sqrt{\tilde{\kappa}}\big)$ thanks to the use of Nesterov momentum. This indicates that an appropriate use of Nesterov momentum in our algorithm could allow for the larger mini-batches required to balance the computational cost of using scaled proximal mappings. The improved iteration complexity of the scaled proximal mappings will then give an overall acceleration in terms of wall-clock time. As discussed in (Allen-Zhu, 2016), the momentum mechanism in AccProxSVRG fails to accelerate ProxSVRG unless $\tilde{\kappa} \geq n^2$. In contrast, as we will see, our algorithm will be able to accelerate the convergence also in these scenarios. In summary, our algorithm is designed to run in an inner-outer fashion as ProxSVRG with large mini-batch sizes and Nesterov momentum to compensate for the increased computational cost of subproblems. The overall procedure is summarized in Algorithm 2.

Algorithm 2 Inexact Accelerated Scaled Proximal SVRG

Input: $\tilde{\boldsymbol{x}}_0$, $\boldsymbol{H}$, $\{\mathcal{B}_k\}_{k=0}^{T}$, η, τ

1: **for** $s = 0, 1, \ldots, S$ **do**
2: $\quad \nabla f(\tilde{\boldsymbol{x}}_s) \leftarrow \frac{1}{n}\sum_1^n \nabla f_i(\tilde{\boldsymbol{x}}_s)$
3: $\quad \boldsymbol{x}_0 \leftarrow \boldsymbol{z}_0 \leftarrow \tilde{\boldsymbol{x}}_s$
4: $\quad$ **for** $k = 0, 1, \ldots, T-1$ **do**
5: $\quad\quad \boldsymbol{y}_k \leftarrow \frac{1}{1+\tau}\boldsymbol{x}_k + \frac{\tau}{1+\tau}\boldsymbol{z}_k$
6: $\quad\quad \boldsymbol{v}_k \leftarrow \nabla f_{\mathcal{B}_k}(\boldsymbol{y}_k) - \nabla f_{\mathcal{B}_k}(\tilde{\boldsymbol{x}}_s) + \nabla f(\tilde{\boldsymbol{x}}_s)$
7: $\quad\quad \boldsymbol{x}_{k+1} \approx \mathrm{prox}_{\eta h}^{\boldsymbol{H}}\left(\boldsymbol{y}_k - \eta \boldsymbol{H}^{-1}\boldsymbol{v}_k\right)$
8: $\quad\quad \boldsymbol{g}_{k+1} \leftarrow \frac{1}{\eta}(\boldsymbol{y}_k - \boldsymbol{x}_{k+1})$
9: $\quad\quad \boldsymbol{z}_{k+1} \leftarrow \boldsymbol{z}_k + \tau(\boldsymbol{y}_k - \boldsymbol{z}_k) - \frac{\tau}{\mu}\boldsymbol{g}_{k+1}$
10: $\quad$ **end for**
11: $\quad \tilde{\boldsymbol{x}}_{s+1} \leftarrow \boldsymbol{x}_T$
12: **end for**

Output: $\tilde{\boldsymbol{x}}_S$

4.2. Convergence Argument

In this subsection, we will show that as long as the errors in evaluating the scaled proximal mappings are controlled in an appropriate way, the iterates generated by the outer loop of Algorithm 2 converge linearly in expectation to the optimal solution. Recall that in Step 7 of Algorithm 2, we want to find an ϵ_k-optimal solution in the sense of (5) to the following problem:

$$\underset{\boldsymbol{x}\in\mathbb{R}^d}{\text{minimize}}\ \frac{1}{2\eta}\left\|\boldsymbol{x} - \boldsymbol{y}_k + \eta\boldsymbol{H}^{-1}\boldsymbol{v}_k\right\|_{\boldsymbol{H}}^2 + h(\boldsymbol{x}). \tag{10}$$

The next lemma quantifies the progress made by one inner iteration of the algorithm. Our proof builds on a Lyapunov argument using a Lyapunov function on the form:

$$V_k = F(\boldsymbol{x}_k) - F(\boldsymbol{x}^\star) + \frac{\mu}{2}\left\|\boldsymbol{z}_k - \boldsymbol{x}^\star\right\|_{\boldsymbol{H}}^2. \tag{11}$$

Lemma 2. *Let Assumptions 1–2 hold and let* $\boldsymbol{x}^\star = \mathrm{argmin}_{\boldsymbol{x}} F(\boldsymbol{x})$, $\eta = 1/L_{\mathrm{avg}}$ *and* $\tau = \sqrt{\mu/2L_{\mathrm{avg}}}$. *If the mini-batch size is chosen such that* $b \geq 60\sqrt{L_{\mathrm{avg}}/\mu}$, *then for any* $k \in \{0, \ldots, T-1\}$, *there exists a vector* $\boldsymbol{\xi}_k \in \mathbb{R}^d$ *such that* $\|\boldsymbol{\xi}_k\|_{\boldsymbol{H}^{-1}} \leq \sqrt{2\eta\epsilon_k}$ *and*

$$\mathbb{E}\, V_{k+1} \leq (1-\tau)\,\mathbb{E}\, V_k + \tau L_{\mathrm{avg}}\,\mathbb{E}\,\langle \boldsymbol{\xi}_k, \boldsymbol{x}^\star - \boldsymbol{z}_k\rangle + 5\epsilon_k + \frac{\tau}{5}\,\mathbb{E}\left\{F(\boldsymbol{x}_k) - F(\boldsymbol{x}^\star) + F(\tilde{\boldsymbol{x}}_s) - F(\boldsymbol{x}^\star)\right\}. \tag{12}$$

Proof. See Appendix D. □

Remark 1. *Our proof is direct and based on natural Lyapunov functions, thereby avoiding the use of stochastic estimate subsequences as in (Nitanda, 2014) which is already very complicated even when the subproblems are solved exactly and* $\boldsymbol{H} = \boldsymbol{I}$. *We stress that the result in Lemma 2 also holds for smaller mini-batch sizes, namely* $b \in \left\{1, \ldots, O(\sqrt{L_{\mathrm{avg}}/\mu})\right\}$, *provided that the step size* η *is reduced accordingly. In favor of a simple proof, we only report the large mini-batch result here.*

Equipped with Lemma 2, we can now characterize the progress made by one outer iteration of Algorithm 2.

Theorem 2. *Let Assumptions 1–2 hold. Suppose that the parameters* η, b, *and* τ *are chosen according to Lemma 2 and define* $\rho = 9\tau/10$. *Then, if the errors in solving the subproblems satisfy*

$$\epsilon_k \leq (1-\rho)^k V_0$$

for all $k \in \{0, \ldots, T-1\}$ *and* $T \geq (4\log c)/3\rho$, *where* c *is a universal constant, then for every* $s \in \mathbb{N}_+$,

$$\mathbb{E}\left\{F(\tilde{\boldsymbol{x}}_s) - F(\boldsymbol{x}^\star)\right\} \leq \frac{2}{3}\,\mathbb{E}\left\{F(\tilde{\boldsymbol{x}}_{s-1}) - F(\boldsymbol{x}^\star)\right\}.$$

Proof. See Appendix E. □

Remark 2. *The theorem indicates that if the errors in solving the subproblems are controlled appropriately, the outer iterates generated by Algorithm 2 converge linearly in expectation to the optimal solution. Since* V_0 *depends on* $\boldsymbol{x}^\star$, *it is difficult to provide a general closed-form expression for the target precisions* ϵ_k. *However, we will show below that with a certain policy for selecting the initial point, it is sufficient to run the solver a constant number of iterations independently of* $\boldsymbol{x}^\star$. *We stress that the results in this section are valid for minimizing general convex composite functions (2) and not limited to the elastic net problem.*

5. Warm-Start

The overall complexity of Algorithm 2 depends strongly on our ability to solve (10) in a reasonable computational time. If one naively starts the solver at a random point, it may take many iterations to meet the target precision. Thus, it is necessary to have a well-designed warm-start procedure for initializing the solver. Intuitively, the current iterate $\boldsymbol{x}_k$ can be a reasonable starting point since the next iterate $\boldsymbol{x}_{k+1}$ should not be too far away from $\boldsymbol{x}_k$. However, in order to achieve a strong theoretical running time, we use a rather different scheme inspired by (Lin et al., 2015). Let us first define the vector $\boldsymbol{u}_k = \boldsymbol{y}_k - \eta\boldsymbol{H}^{-1}\boldsymbol{v}_k$ for $k \in \{0, 1, \ldots, T-1\}$ and the function

$$p(\boldsymbol{z}, \boldsymbol{u}) = h(\boldsymbol{z}) + \frac{1}{2\eta}\left\|\boldsymbol{z} - \boldsymbol{u}\right\|_{\boldsymbol{H}}^2.$$

Then, the kth subproblem seeks for $\boldsymbol{x}_{k+1}$ such that

$$p(\boldsymbol{x}_{k+1}, \boldsymbol{u}_k) - p\left(\boldsymbol{x}_{k+1}^\star, \boldsymbol{u}_k\right) \leq \epsilon_k, \tag{13}$$

where $\boldsymbol{x}_{k+1}^\star$ is the exact solution. We consider the initialization policy

$$\boldsymbol{z}_0 = \mathrm{prox}_{\gamma h}\left(\boldsymbol{x}_k - \frac{\gamma}{\eta}\boldsymbol{H}(\boldsymbol{x}_k - \boldsymbol{u}_{k-1})\right), \tag{14}$$

which can be seen as one step of the proximal gradient method applied to $p(\boldsymbol{z}, \boldsymbol{u}_{k-1})$ starting at the current $\boldsymbol{x}_k$.

The following proposition characterizes the difference in objective realized by $\boldsymbol{z}_0$ and $\boldsymbol{x}^\star_{k+1}$.

Proposition 2. *Let $\boldsymbol{z}_0$ be defined by (14) with $\gamma = \eta/\lambda_1(\boldsymbol{H})$. Let $\kappa_{\text{sub}} = \lambda_1(\boldsymbol{H})/\lambda_r(\boldsymbol{H})$ be the condition number of the subproblems. Assume that the errors in solving the subproblems satisfy $\epsilon_k \leq (1-\rho)^k V_0$ for all $k \in \{0, 1, \ldots, T-1\}$. Then,*

$$p(\boldsymbol{z}_0, \boldsymbol{u}_k) - p\left(\boldsymbol{x}^\star_{k+1}, \boldsymbol{u}_k\right) \leq \frac{\kappa_{\text{sub}}}{1-\rho}\epsilon_k.$$

Proof. See Appendix F. □

The proposition, together with (13), implies that it suffices to find $\boldsymbol{x}_{k+1}$ such that

$$p(\boldsymbol{x}_{k+1}, \boldsymbol{u}_k) - p\left(\boldsymbol{x}^\star_{k+1}, \boldsymbol{u}_k\right) \leq \frac{1-\rho}{\kappa_{\text{sub}}}\left(p(\boldsymbol{z}_0, \boldsymbol{u}_k) - p\left(\boldsymbol{x}^\star_{k+1}, \boldsymbol{u}_k\right)\right). \quad (15)$$

This is significant since one only needs to reduce the residual error by a constant factor independent of the target precision. Note also that the condition number κ_{sub} is much smaller than $\kappa \approx \lambda_1(\boldsymbol{H})/(\lambda_d+\gamma_2)$, and computing the gradient of the smooth part of $p(\boldsymbol{z}, \boldsymbol{u})$ only takes time $O(rd)$ instead of $O(nd)$ as in the original problem. Those properties imply that the subproblems can be solved efficiently by iterative methods, where only a small (and known) constant number of iterations is needed. The next section develops the final details of convergence proof.

6. Global Time Complexity

We start with the time complexity of Algorithm 2. Let $\mathcal{T}(\alpha)$ be the number of gradient evaluations that a subproblem solver takes to reduce the residual error by a factor α. Then, by Proposition 2, one can find an ϵ_k-optimal solution to the kth subproblem by at most $\mathcal{T}(\kappa_{\text{sub}}/(1-\rho))$ gradient evaluations, where one gradient evaluation is equivalent to $O(d)$ flops. Consider the same setting of Theorem 2 and suppose that the subproblems are initialized by (14). Then, the time complexity of Algorithm 2 is given by:

$$O\left(d\left(n + \kappa_{\boldsymbol{H}} + \sqrt{\kappa_{\boldsymbol{H}}}\,\mathcal{T}\left(\frac{\kappa_{\text{sub}}}{1-\rho}\right)\right)\log(1/\epsilon)\right), \quad (16)$$

where the first summand is due to the full gradient evaluation at each outer loop; the second one comes from the fact that one needs $O(\sqrt{\kappa_{\boldsymbol{H}}})$ inner iterations, each of which uses a mini-batch of size $O(\sqrt{\kappa_{\boldsymbol{H}}})$; and the third one is the result of $O(\sqrt{\kappa_{\boldsymbol{H}}})$ inner iterations, each of which solves a subproblem that needs $\mathcal{T}(\kappa_{\text{sub}}/(1-\rho))$ gradient evaluations. We can now put things together and state our main result.

Proposition 3. *Suppose that the approximate Hessian matrix $\boldsymbol{H}$ is given by (6) and that Algorithm 2 is invoked with $f(\boldsymbol{x}) = \frac{1}{2n}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2 + \frac{\gamma_2}{2}\|\boldsymbol{x}\|_2^2$ and $h(\boldsymbol{x}) = \|\boldsymbol{x}\|_1$. Assume further that the subproblems are solved by the accelerated proximal gradient descent method (Beck & Teboulle, 2009; Nesterov, 2013). Our method can find an ϵ-optimal solution in time*

$$O\big(d(n + \kappa_{\boldsymbol{H}})\log(1/\epsilon)\big).$$

Proof. The task reduces to evaluating the term $\mathcal{T}(\kappa_{\text{sub}}/(1-\rho))$ in (16). Recall that the iteration complexity of the accelerated proximal gradient descent method for minimizing the function $F(x) = f(x) + h(x)$, where f is a smooth and strongly convex function and h is a possibly non-smooth convex regularizer, initiliazed at $\boldsymbol{x}_0$ is given by $\sqrt{\kappa}\log\frac{F(\boldsymbol{x}_0)-F(\boldsymbol{x}^\star)}{\epsilon}$, where κ is the condition number. By invoking the about result with $F(\boldsymbol{x}) = p(\boldsymbol{x}, \boldsymbol{u}_k)$, $\boldsymbol{x}^\star = \boldsymbol{x}^\star_{k+1}$, $\boldsymbol{x}_0 = \boldsymbol{z}_0$, $\kappa = \kappa_{\text{sub}}$, and ϵ is the right-hand side of (15), it follows that the number of iterations for each subproblem can be bounded by

$$O\left(\sqrt{\kappa_{\text{sub}}}\log\left(\kappa_{\text{sub}}/(1-\rho)\right)\right). \quad (17)$$

In addition, each iteration takes time $O(rd)$ to compute the gradient implying the time complexity

$$O\left(d\big(n + \kappa_{\boldsymbol{H}} + r\sqrt{\kappa_{\text{sub}}}\sqrt{\kappa_{\boldsymbol{H}}}\log\frac{\kappa_{\text{sub}}}{1-\rho}\big)\log\frac{1}{\epsilon}\right).$$

Selecting r such that $\kappa_{\boldsymbol{H}} \geq \kappa_{\text{sub}}$ completes the proof. □

We can easily recognize that this time complexity has the same form as the stochastic first-order methods discussed in Section 1.2.1 with the condition number $\tilde{\kappa}$ replaced by $\kappa_{\boldsymbol{H}}$. It has been shown in Theorem 1 that $\kappa_{\boldsymbol{H}}$ can be much smaller than $\tilde{\kappa}$, especially, when $\boldsymbol{C}$ has a rapidly decaying spectrum. Note also that κ_{sub} is available for free to us after having approximated the Hessian matrix.

7. Experimental Results

In this section, we perform numerical experiments to verify the efficacy of the proposed method on real world data sets (Chang & Lin, 2011; Guyon et al., 2008). We compare our method with the coordinate descent algorithm (`BCD`) (Tibshirani et al., 2015) and several first-order methods: `FISTA` (Beck & Teboulle, 2009) with optimal step-size; `Prox-SVRG` (Xiao & Zhang, 2014) with epoch length $2n/b$ as suggested by the authors; `Katyusha1` (Allen-Zhu, 2016) with epoch length $2n/b$, Katyusha momentum $\tau_2 = 0.5/b$ as suggested by the author; and our method with epoch length $2n/b$. Since `Katyusha1` can use a mini-batch of size $\sqrt{n}$ without slowing down the convergence, we set $b = \sqrt{n}$ for all

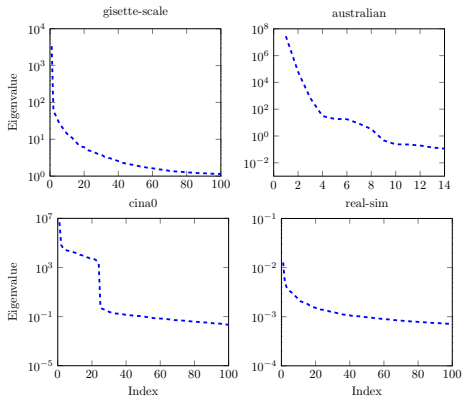

Figure 1. Spectrum of the matrix $\boldsymbol{C}$ for different data sets.

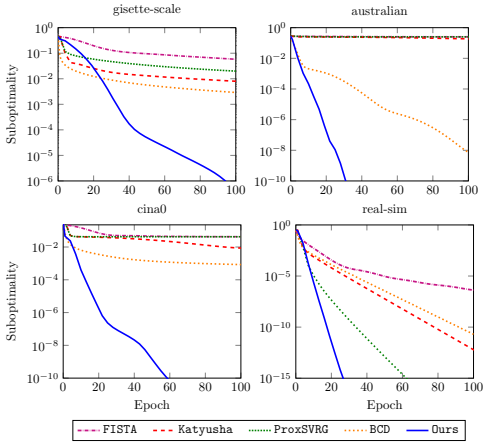

Figure 2. Suboptimality versus the number of epochs.

stochastic methods unless otherwise stated. Finally, to make a fair comparison, for each algorithm above, we tune only the step size, from the set $\eta \times \{10^k, 2 \times 10^k, 5 \times 10^k | k \in \{0, \pm 1, \pm 2\}\}$, where η is the theoretical step size, and report the one having smallest objective value. Other hyper-parameters are set to their theory-predicted values. All methods are initialized at $\mathbf{0}$. For the subproblems in Algorithm 2, we just simply run `FISTA` with $\sqrt{\kappa_{\text{sub}}} \log \kappa_{\text{sub}}$ iterations as discussed previously, without any further tunning steps. The value of r is chosen as a small fraction of d so that the preprocessing time of Algorithm 1 is negligible. Note that the available spectrum of $\boldsymbol{C}$ after running Algorithm 1 also provides an insightful way to choose r.

Figure 2 shows the suboptimality in objective versus the number of epochs for different algorithms solving the elastic net problem. We can see that our method systematically outperforms the others in all settings, and that there is a clear correspondence between the spectrum of $\boldsymbol{C}$ in Fig. 1 and the potential speed-up. Notably, for ill-conditioned data sets such as `australian` and `cina0`, all the first-order methods make almost no progress in the first 100

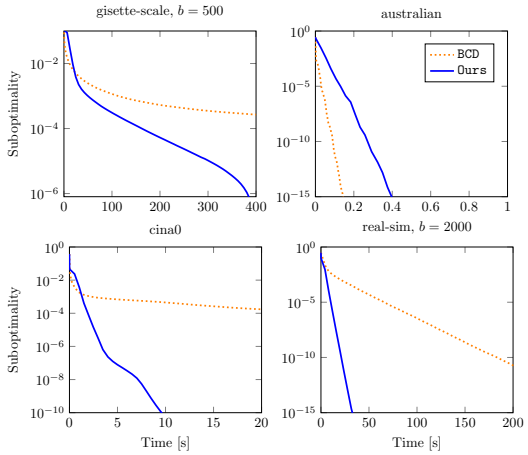

Figure 3. Suboptimality versus runtime.

epochs, while our method can find a high-accuracy solution within tens of epochs, demonstrating a great benefit of second-order information. On the other hand, for a well-conditioned data set that does not exhibit high curvature such as `real-sim`, `ProxSVRG` is comparable to our method and even outperforms `Katyusha1`. This agrees with the time complexities summarized in Table 1.

For runtime comparisons, we only report the performance of `BCD` and our method since they outperform the others by a large margin. We observe that for problems with small or medium dimensions, it is useful to have `BCD` as subproblem solver in our method since `BCD` is very effective in solving LASSO-type problems. However, for high-dimensional problems, `BCD` needs to loop over d coordinates, which can be excessive since we only need to reduce the error of the subproblem by a small constant factor. For this reason, we use `BCD` as subproblem solver for `australian` and `cina0`, and `FISTA` for `gisette` and `real-sim`. The runtime plots in Fig 3 demonstrate that curvature information offers significant acceleration over one of the state-of-the-art methods for solving the elastic net problem.

8. Conclusions

We have proposed and analyzed a novel second-order method for solving the elastic net problem. By carefully exploiting the problem structure, we demonstrated that it is possible to deal with the non-smooth objective and to efficiently inject curvature information into the optimization process without (significantly) increasing the computational cost per iteration. The combination of second-order information, fast iterative solvers, and a well-designed warm-start procedure results in a significant improvement of the total runtime complexity over popular first-order methods. An interesting direction for future research would be to go beyond the quadratic loss.

Acknowledgements

This work was supported in part by the Knut and Alice Wallenberg Foundation, the Swedish Research Council and the Swedish Foundation for Strategic Research. We thank the anonymous reviewers for their useful comments and suggestions.

References

Agarwal, N., Bullins, B., and Hazan, E. Second-order stochastic optimization for machine learning in linear time. *Journal of Machine Learning Research*, 18(1): 4148–4187, 2017.

Allen-Zhu, Z. Katyusha: The first direct acceleration of stochastic gradient methods. *arXiv preprint arXiv:1603.05953*, 2016.

Arjevani, Y. and Shamir, O. Oracle complexity of second-order methods for finite-sum problems. In *International Conference on Machine Learning*, pp. 205–213, 2017.

Beck, A. and Teboulle, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

Bertsekas, D. P., Nedić, A., and Ozdaglar, A. E. *Convex analysis and optimization*. Athena Scientific, 2003.

Byrd, R. H., Hansen, S. L., Nocedal, J., and Singer, Y. A stochastic quasi-Newton method for large-scale optimization. *SIAM Journal on Optimization*, 26(2):1008–1031, 2016.

Chang, C.-C. and Lin, C.-J. LIBSVM: A library for support vector machines. *ACM transactions on intelligent systems and technology (TIST)*, 2(3):27, 2011.

Defazio, A. A simple practical accelerated method for finite sums. In *Advances in Neural Information Processing Systems*, pp. 676–684, 2016.

Defazio, A., Bach, F., and Lacoste-Julien, S. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, pp. 1646–1654, 2014.

Erdogdu, M. A. and Montanari, A. Convergence rates of sub-sampled Newton methods. In *Advances in Neural Information Processing Systems*, pp. 3052–3060. MIT Press, 2015.

Frostig, R., Ge, R., Kakade, S., and Sidford, A. Unregularizing: Approximate proximal point and faster stochastic algorithms for empirical risk minimization. In *International Conference on Machine Learning*, pp. 2540–2548, 2015.

Ghanbari, H. and Scheinberg, K. Proximal quasi-Newton methods for convex optimization. *arXiv preprint arXiv:1607.03081*, 2016.

Gonen, A., Orabona, F., and Shalev-Shwartz, S. Solving ridge regression using sketched preconditioned SVRG. In *International Conference on Machine Learning*, pp. 1397–1405, 2016.

Guyon, I., Aliferis, C., Cooper, G., Elisseeff, A., Pellet, J.-P., Spirtes, P., and Statnikov, A. Design and analysis of the causation and prediction challenge. In *Causation and Prediction Challenge*, pp. 1–33, 2008.

Halko, N., Martinsson, P.-G., and Tropp, J. A. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288, May 2011.

Hsu, D., Kakade, S. M., and Zhang, T. Random design analysis of ridge regression. In *Conference on Learning Theory*, pp. 9–1, 2012.

Hu, C., Kwok, J. T., and Pan, W. Accelerated gradient methods for stochastic optimization and online learning. In *Advances in Neural Information Processing Systems*, pp. 781–789, Vancouver, Canada, Dec 2009.

Lee, J. D., Sun, Y., and Saunders, M. Proximal Newton-type methods for convex optimization. In *Advances in Neural Information Processing Systems*, pp. 827–835, 2012.

Lin, H., Mairal, J., and Harchaoui, Z. A universal catalyst for first-order optimization. In *Advances in Neural Information Processing Systems*, pp. 3384–3392, 2015.

Liu, X., Hsieh, C.-J., Lee, J. D., and Sun, Y. An inexact subsampled proximal Newton-type method for large-scale machine learning. *arXiv preprint arXiv:1708.08552*, 2017.

Moritz, P., Nishihara, R., and Jordan, M. A linearly-convergent stochastic L-BFGS algorithm. In *Artificial Intelligence and Statistics*, pp. 249–258, 2016.

Musco, C. and Musco, C. Randomized block Krylov methods for stronger and faster approximate singular value decomposition. In *Advances in Neural Information Processing Systems*, pp. 1396–1404, 2015.

Nesterov, Y. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

Nitanda, A. Stochastic proximal gradient descent with acceleration techniques. In *Advances in Neural Information Processing Systems*, pp. 1574–1582, Montréal, Canada, Dec 2014.

Rodomanov, A. and Kropotov, D. A superlinearly-convergent proximal Newton-type method for the optimization of finite sums. In *International Conference on Machine Learning*, pp. 2597–2605, 2016.

Roosta-Khorasani, F. and Mahoney, M. W. Sub-sampled Newton methods I: Globally convergent algorithms. *arXiv preprint arXiv:1601.04737*, 2016.

Schmidt, M., Roux, N. L., and Bach, F. Convergence rates of inexact proximal-gradient methods for convex optimization. In *Advances in Neural Information Processing Systems*, pp. 1458–1466, Granada, Spain, Dec 2011.

Tibshirani, R., Wainwright, M., and Hastie, T. *Statistical learning with sparsity: The lasso and generalizations*. Chapman and Hall/CRC, 2015.

Wang, J. and Zhang, T. Improved optimization of finite sums with minibatch stochastic variance reduced proximal iterations. *arXiv preprint arXiv:1706.07001*, 2017.

Woodworth, B. E. and Srebro, N. Tight complexity bounds for optimizing composite objectives. In *Advances in Neural Information Processing Systems*, pp. 3639–3647, 2016.

Xiao, L. and Zhang, T. A proximal stochastic gradient method with progressive variance reduction. *SIAM Journal on Optimization*, 24(4):2057–2075, Dec 2014.

Xu, P., Yang, J., Roosta-Khorasani, F., Ré, C., and Mahoney, M. W. Sub-sampled Newton methods with non-uniform sampling. In *Advances in Neural Information Processing Systems*, pp. 3000–3008, 2016.

Zou, H. and Hastie, T. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67(2):301–320, 2005.