Journal of Machine Learning Research 25 (2024) 1-57 Submitted 9/23; Revised 10/24; Published 11/24

PROMISE: Preconditioned Stochastic Optimization Methods by Incorporating Scalable Curvature Estimates

Zachary Frangella[*] ZFRAN@STANFORD.EDU
Department of Management Science and Engineering
Stanford University

Pratik Rathore[*] PRATIKR@STANFORD.EDU
Department of Electrical Engineering
Stanford University

Shipu Zhao SZ533@CORNELL.EDU
Department of Systems Engineering
Cornell University

Madeleine Udell UDELL@STANFORD.EDU
Department of Management Science and Engineering
Stanford University

Editor: Peter Richtarik

Abstract

Ill-conditioned problems are ubiquitous in large-scale machine learning: as a data set grows to include more and more features correlated with the labels, the condition number increases. Yet traditional stochastic gradient methods converge slowly on these ill-conditioned problems, even with careful hyperparameter tuning. This paper introduces PROMISE (**Pr**econditioned Stochastic **O**ptimization **M**ethods by **I**ncorporating **S**calable Curvature **E**stimates), a suite of sketching-based preconditioned stochastic gradient algorithms that deliver fast convergence on ill-conditioned large-scale convex optimization problems arising in machine learning. PROMISE includes preconditioned versions of SVRG, SAGA, and Katyusha; each algorithm comes with a strong theoretical analysis and effective default hyperparameter values. Empirically, we verify the superiority of the proposed algorithms by showing that, using default hyperparameter values, they outperform or match popular *tuned* stochastic gradient optimizers on a test bed of 51 ridge and logistic regression problems assembled from benchmark machine learning repositories. On the theoretical side, this paper introduces the notion of *quadratic regularity* in order to establish linear convergence of all proposed methods even when the preconditioner is updated infrequently. The speed of linear convergence is determined by the *quadratic regularity ratio*, which often provides a tighter bound on the convergence rate compared to the condition number, both in theory and in practice, and explains the fast global linear convergence of the proposed methods.

Keywords: stochastic optimization, preconditioning, randomized low-rank approximation, lazy Hessians

[*] Equal contribution. A longer, more detailed version of this manuscript is available at `https://arxiv.org/abs/2309.02014v2`.

1. Introduction

Modern machine learning (ML) poses significant challenges for optimization, owing to the sheer scale of the problems. Modern data sets are both enormous and high-dimensional, often with millions of samples and features. As a consequence, classic methods such as gradient descent and L-BFGS, which make a full pass through the data at each iteration, are prohibitively expensive. In this context, stochastic gradient descent (SGD) and its variants, which operate on only a small mini-batch of data at each iteration, have become the dominant optimization methods for modern ML.

When the problem is well-conditioned, SGD quickly finds models that are nearly optimal. Further, although classic SGD converges to a ball around the optimum (with fixed learning rate) or sublinearly (with decaying learning rate) (Moulines and Bach, 2011; Gower et al., 2019b), *variance reduction* techniques like SVRG, SAGA, Katyusha, and L-Katyusha significantly improve performance on convex problems, and converge linearly to the optimum for strongly convex problems (Johnson and Zhang, 2013; Defazio et al., 2014; Allen-Zhu, 2018; Kovalev et al., 2020).

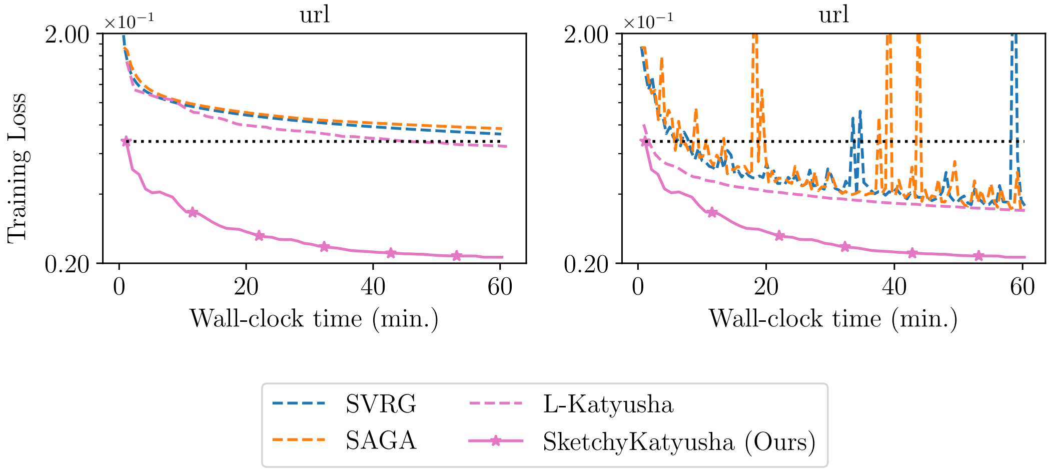

Figure 1: SketchyKatyusha (an algorithm in the PROMISE suite, see Algorithm 5) with its default hyperparameters outperforms standard stochastic gradient optimizers with both default (left) and tuned (right) hyperparameters. The loss curves start after a single epoch of training has been completed; the black dotted line indicates the training loss attained by SketchyKatyusha after a single epoch. Each optimizer is allotted 1 hour of runtime.

Unfortunately SGD and related algorithms are difficult to tune and converge slowly when the data is poorly conditioned. Parameters like the learning rate are difficult to choose and important to get right: slow convergence or divergence loom on either side of the best parameter choice (Nemirovski et al., 2009). Even with the best learning rate, in the worst case, variance-reduced stochastic gradient methods require at least $\mathcal{O}((n+\sqrt{n\kappa})\log(1/\epsilon))$ stochastic gradient evaluations to reach ϵ accuracy (Woodworth and Srebro, 2016), which is problematic as the condition number for many ML problems is typically on the order of 10^4 to 10^8 (see Figure 13 in Frangella et al. (2023a)). Hence the convergence of stochastic gradient methods can be excruciatingly slow (see Fig. 1) and popular stochastic optimizers often provide low-quality solutions even with generous computational budgets.

How should the challenges of ill-conditioning and sensitivity to the learning rate be addressed? Classical optimization wisdom suggests using second-order information based on the Hessian. Second-order methods converge locally at superlinear rates under mild assumptions and beat first-order methods in practice (Nocedal and Wright, 1999; Boyd and Vandenberghe, 2004).

Many authors have attempted to develop second-order methods that use stochastic gradients and Hessians, but major difficulties remain; see Section 4 and especially Table 9 for details. First, no previous stochastic second-order method delivers fast local-linear convergence without vanishing noise in the gradient estimates, as noted in Kovalev et al. (2019); Frangella et al. (2023a). The strategies used to reduce this noise, including exponentially increasing gradient batchsizes and Hessian batchsizes that may depend on the condition number, lead to exceedingly slow iterations as the algorithm converges. Second, even the methods that allow a stochastic Hessian require an expensive (and slow) new estimate of the Hessian at every iteration. Finally, most of these methods are difficult to deploy in real-world ML pipelines, as they introduce new hyperparameters without practical guidelines for choosing them.

In this paper, we introduce PROMISE, a suite of preconditioned stochastic gradient methods that use scalable curvature estimates to directly address each of these problems. PROMISE methods estimate second-order information from minibatch data (i.e., stochastic Hessians) to avoid difficulties with hyperparameter selection and ill-conditioning, and they use infrequent ("lazy") Hessian updates. The resulting algorithms, such as SketchyKatyusha (Algorithm 5), are fast enough per-iteration to compete with first-order methods, yet converge much faster on ill-conditioned problems with minimal or no hyperparameter tuning.

Figure 1 illustrates the benefits of PROMISE by applying SketchyKatyusha to a malicious link detection task using the url data set ($n = 2{,}396{,}130$, $p = 3{,}231{,}961$), which yields a large l^2-regularized logistic regression problem. Popular stochastic optimizers, using the default learning rates, perform poorly. In contrast, with default hyperparameters, the proposed method SketchyKatyusha achieves a loss three times smaller than the best competing method after an hour of training! In fact, even after an hour of training, the other optimizers barely match the training loss that SketchyKatyusha achieves after a single epoch (pass through the data). Even with extensive hyperparameter tuning (which, in practice, increases the cost of optimization by orders of magnitude), the other first-order methods cannot match the performance of SketchyKatyusha with its default hyperparameters.

PROMISE also improves on preexisting theory for stochastic second-order methods. In contrast to prior approaches, PROMISE methods achieve linear convergence with lazy updates to the preconditioner and without large batchsizes for the gradient and Hessian. Significantly, PROMISE methods come with default hyperparameters (including the learning rate) that enable them to work out-of-the-box and outperform or match popular stochastic optimizers *tuned* to achieve their best performance. Numerical experiments on a test bed of 51 ridge and logistic-regression problems verify this claim. Hence our methods avoid the usual theory-practice gap: our theoretical advances yield practical algorithms.

In order to show linear convergence under lazy updates, our analysis introduces a new analytic quantity, the *quadratic regularity ratio*, that controls the convergence rate of all

PROMISE methods. The quadratic regularity ratio generalizes the condition number to the Hessian norm. Unlike the condition number, the quadratic regularity ratio equals one for quadratic objectives and approaches one as the iterates approach the optimum for any objective with a Lipschitz Hessian. Hence the quadratic regularity ratio often gives tighter convergence rates than the condition number and explains why the proposed methods empirically exhibit fast global linear convergence and outperform the competition.

1.1 PROMISE

PROMISE methods solve convex finite-sum minimization (FSM) problems of the form

$$\underset{w\in\mathbb{R}^p}{\text{minimize}}\ F(w) := \frac{1}{n}\sum_{i=1}^{n} f_i(w) + \frac{\nu}{2}\|w\|^2, \tag{FSM}$$

where each f_i is real-valued, smooth, and convex, and $\nu > 0$.

We provide a high-level overview of the PROMISE methods in the Meta-algorithm and Table 1. Each iteration consists of two phases: a (lazy) preconditioner update and a parameter update. By default, PROMISE methods update the preconditioner at a fixed frequency (such as once per epoch) using a stochastic Hessian estimate at the current iterate w_k. The learning rate is then recomputed to adapt to the new preconditioner. For the parameter update, our methods compute a stochastic gradient g_k and preconditioned direction $v_k = P^{-1}g_k$. Our methods then use a parameter update subroutine $\mathcal{S}$ to compute the next iterate w_{k+1}. The $*$ in the call to $\mathcal{S}$ denotes additional arguments to perform variance reduction and acceleration.

Meta-algorithm: PROMISE

```
Require: initial iterate $w_0$, stochastic gradient oracle $\mathcal{O}_g$, stochastic Hessian oracle $\mathcal{O}_H$, gradient and Hessian batchsizes $b_g$ and $b_H$, preconditioner object $\mathcal{P}$, preconditioner update times $\mathcal{U} \subseteq \mathbb{N}$, parameter update subroutine $\mathcal{S}$

for $k = 0, 1, \ldots$ do
    # Preconditioner update
    if $k \in \mathcal{U}$ then
        $\mathcal{P}$.update($\mathcal{O}_H(w_k, b_H)$)        ▷ Update preconditioner $P$ via stochastic Hessian
        $\eta = \mathcal{P}$.get_learning_rate()        ▷ Compute learning rate based on $P$
    end if

    # Parameter update
    $g_k = \mathcal{O}_g(w_k, b_g)$        ▷ Compute stochastic gradient
    $v_k = \mathcal{P}$.direction($g_k$)        ▷ Compute $v_k = P^{-1}g_k$
    $w_{k+1} = \mathcal{S}(w_k, g_k, v_k, *)$        ▷ Compute next iterate
end for
```

The finite-sum structure of the objective (FSM) makes it easy to construct unbiased estimators of the gradient $\nabla F(w)$ and the Hessian $\nabla^2 F(w)$, given batchsizes b_g and b_H, as

$$\widehat{\nabla} F(w) = \frac{1}{b_g}\sum_{i\in\mathcal{B}_g} \nabla f_i(w) + \nu w, \quad \widehat{\nabla}^2 F(w) = \frac{1}{b_H}\sum_{i\in\mathcal{B}_H} \nabla^2 f_i(w) + \nu I,$$

where $\mathcal{B}_g$ and $\mathcal{B}_H$, with size b_g and b_H respectively, are sampled independently and uniformly from $\{1, \ldots, n\}$. As a concrete example, consider a generalized linear model (GLM) with $f_i(w) = \phi_i(a_i^T w)$. Then

$$\nabla f_i(w) = \phi_i'(a_i^T w)a_i, \quad \nabla^2 f_i(w) = \phi_i''(a_i^T w)a_i a_i^T.$$

Input	Description
w_0	Initial iterate, typically set to 0.
$\mathcal{O}_g$	Computes a stochastic gradient.
$\mathcal{O}_H$	Used for computing a stochastic/subsampled Hessian. Does not compute the entire subsampled Hessian in practice.
b_g, b_H	Batchsizes for computing stochastic gradients and Hessians. Used as inputs to $\mathcal{O}_g$ and $\mathcal{O}_H$.
$\mathcal{P}$	Preconditioner object. Examples provided in Section 2.1.
$\mathcal{U}$	Times at which to update the preconditioner.
$\mathcal{S}$	Subroutine that updates the iterate. May include calculations related to variance reduction and acceleration.

Table 1: Inputs to the Meta-algorithm.

The stochastic Hessian as written is a $p \times p$ matrix: rather large! But none of the methods we discuss instantiate such a matrix. Instead, they take advantage of the low-rank structure of the preconditioner to compute the approximate Newton direction $P^{-1}g_k$ efficiently using the Woodbury formula (see Table 4).

1.2 Contributions

We summarize the contributions of this work as follows:

1. We propose preconditioned versions of SVRG, SAGA, and Katyusha, which we call SketchySVRG, SketchySAGA, and SketchyKatyusha. These methods use stochastic approximations to the Hessian to perform preconditioning.

2. We formally describe a wide array of preconditioners that are compatible with our methods. We show that any preconditioner that approximates the Hessian sufficiently well is compatible with our theoretical convergence results.

3. We define the quadratic regularity ratio, which generalizes the condition number globally to the Hessian norm, and use this ratio to prove our methods converge linearly to the optimum despite lazy updates to the preconditioner.

4. We show global linear convergence, independent of the condition number, for SketchySVRG, SketchySAGA, and SketchyKatyusha applied to ridge regression. We also show local linear convergence, independent of the condition number, for SketchySVRG on any strongly convex finite-sum problem with Lipschitz Hessians.

5. We provide default hyperparameters and a heuristic to automatically compute a good learning rate for our proposed methods.

6. We present extensive experiments demonstrating that SketchySVRG, SketchySAGA, and SketchyKatyusha, equipped with their default hyperparameters and learning rate heuristic, outperform popular stochastic optimizers for GLMs.

1.3 Roadmap

Section 2 introduces several scalable preconditioning techniques that are compatible with the PROMISE framework; we provide both implementation details and theoretical results for these preconditioners. Section 3 presents the algorithms that comprise the PROMISE framework, along with default hyperparameters and algorithmic recommendations for various GLMs. Section 4 reviews the literature on preconditioning and stochastic second-order methods and places PROMISE in the context of these existing works. Section 5 establishes linear convergence of all of the proposed methods for strongly convex machine learning problems. Section 6 demonstrates the superior performance of the algorithms in PROMISE over popular *tuned* stochastic optimizers through extensive numerical experiments.

1.4 Notation

Define $[n] \coloneqq \{1, \ldots, n\}$. Throughout the paper, let $\mathcal{B}$ (or $\mathcal{B}_k$) denote subsets of $[n]$ that are sampled independently and uniformly without replacement. The corresponding (unregularized) minibatch gradient and Hessian are given by

$$\begin{aligned} \widehat{\nabla} f(w) &= \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla f_i(w), \\ \widehat{\nabla}^2 f(w) &= \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla^2 f_i(w). \end{aligned}$$

Throughout the paper, we use b_g to refer to the gradient batchsize and b_H to refer to the Hessian batchsize. We abbreviate positive-semidefinite as psd and use $\mathbb{S}_p^+(\mathbb{R})$ to denote the convex cone of psd matrices in $\mathbb{R}^{p \times p}$. The symbol $\preceq$ denotes the Loewner order on the convex cone of psd matrices: $A \preceq B$ means $B - A$ is psd. Given a matrix $A \in \mathbb{S}_p^+(\mathbb{R})$, its eigenvalues in decreasing order are $\lambda_1(A) \geq \lambda_2(A) \geq \cdots \geq \lambda_p(A)$. Moreover, the condition number of $A \in \mathbb{S}_p^+(\mathbb{R})$ is defined to be $\kappa(A) \coloneqq \lambda_1(A)/\lambda_p(A)$. Define $B(w, r)$ to be the closed Euclidean norm ball of radius r, centered at w. We use L_i to denote the smoothness constant of f_i and $L_{\max}$ is defined as $\max_i L_i$.

Throughout the remainder of this paper, we assume we have access to some GLM $\mathcal{M}$. We note our theoretical convergence results do not require the objective to be a GLM, but the implementation of the SASSN preconditioner (Section 2.1) requires this structure. Moreover, most convex machine learning problems arising in practice are GLMs, so we specialize our implementation to GLMs. Given that F is a GLM, we assume access to oracles for obtaining the regularization parameter ν, row subsamples of the data matrix $A \in \mathbb{R}^{n \times p}$, the diagonal of the stochastic Hessian of F (excluding the regularization ν), stochastic gradients of F, and full gradients of F. We present the names, inputs, and outputs of these oracles in Table 2.

Oracle	Output
$\mathcal{M}.\mathtt{get\_reg}()$	ν
$\mathcal{M}.\mathtt{get\_data}(\mathcal{B})$	$A_{\mathcal{B}}$
$\mathcal{M}.\mathtt{get\_hessian\_diagonal}(\mathcal{B}, w)$	$\Phi''(A_{\mathcal{B}}w)$
$\mathcal{M}.\mathtt{get\_stoch\_grad}(\mathcal{B}, w)$	$\frac{1}{\lvert\mathcal{B}\rvert}\sum_{i\in\mathcal{B}} \nabla f_i(w) + \nu w$
$\mathcal{M}.\mathtt{get\_full\_grad}(w)$	$\nabla F(w)$

Table 2: Oracles associated with the GLM $\mathcal{M}$. $\mathcal{B} \subseteq [n]$ is a batch of indices and $w \in \mathbb{R}^p$.

The oracles `get_data` and `get_hessian_diagonal` output $A_{\mathcal{B}}w$ and $\Phi''(A_{\mathcal{B}}w)$, respectively, for minibatch $\mathcal{B} = \{i_1, i_2, \ldots, i_{|\mathcal{B}|}\} \subseteq [n]$, where

$$A_{\mathcal{B}} := \begin{pmatrix} a_{i_1}^T \\ a_{i_2}^T \\ \vdots \\ a_{i_{|\mathcal{B}|}}^T \end{pmatrix}, \qquad \Phi''(A_{\mathcal{B}}w) := \operatorname{diag}\left(\begin{pmatrix} \phi''_{i_1}\left(a_{i_1}^T w\right) \\ \phi''_{i_2}\left(a_{i_2}^T w\right) \\ \vdots \\ \phi''_{i_{|\mathcal{B}|}}\left(a_{i_{|\mathcal{B}|}}^T w\right) \end{pmatrix}\right).$$

The oracles `get_reg`, `get_data` and `get_hessian_diagonal` are used in the preconditioners described in Section 2.1, while the oracles `get_stoch_grad` and `get_full_grad` are used in the optimization algorithms in Section 3. In practice, `get_hessian_diagonal` returns the diagonal as a vector, not a matrix.

2. Scalable Preconditioning Techniques

We present three scalable preconditioning techniques: Subsampled Newton (SSN), Nyström Subsampled Newton (NySSN), and Sketch-and-Solve Subsampled Newton (SASSN), all of which are based on stochastic approximations of the Hessian. The key driver behind the scalability of these methods is that subsampling and randomized low-rank approximation provide cheap, reliable estimates of the curvature.

For general convex, finite-sum objectives, the SSN and NySSN preconditioners require access to a stochastic Hessian, while the SASSN preconditioner requires access to the square root of the stochastic Hessian. Fortunately, the stochastic Hessian in GLMs, $\frac{1}{b_H} A_{\mathcal{B}}^T \Phi''(A_{\mathcal{B}}w) A_{\mathcal{B}}$, has a structure that allows us to compute all of these preconditioners using its square root, $\frac{1}{\sqrt{b_H}} [\Phi''(A_{\mathcal{B}}w)]^{1/2} A_{\mathcal{B}}$.

We present mathematical and algorithmic formulations of the four preconditioning techniques (specialized to GLMs) in Section 2.1. We then compare the computational costs associated with each preconditioner and provide preconditioner recommendations for various problem regimes in Section 2.2. We analyze the approximation quality of the proposed preconditioners in Section 2.3. Finally, we describe how the proposed preconditioners can be extended beyond GLMs in Section 2.4. Any proofs not provided in this section can found in the arxiv report.

2.1 Mathematical and Algorithmic Formulation of Preconditioners

This section provides mathematical formulations for each proposed preconditioner in the GLM setting, and object-oriented pseudocode for SSN. Each preconditioner has `update` and `direction` methods. The `update` method constructs the preconditioner and estimates the preconditioned smoothness constant (i.e., it combines `update` and `get_learning_rate` methods in Meta-algorithm), while the `direction` method applies the preconditioner to a vector (similar to `direction` in Meta-algorithm). These preconditioners play a critical role in the optimization algorithms presented in Section 3.

2.1.1 Subsampled Newton (SSN)

The first preconditioning method we present is SSN (Erdogdu and Montanari, 2015; Roosta-Khorasani and Mahoney, 2019). SSN forms a preconditioner using the Hessian of a random subsample of the terms in the finite-sum objective (FSM). Given a point $w \in \mathbb{R}^p$, SSN constructs the preconditioner

$$P = \frac{1}{b_H} \sum_{i \in \mathcal{B}} \nabla^2 f_i(w) + \rho I, \tag{1}$$

where $\rho \geq \nu$ and $\mathcal{B}$ consists of b_H elements sampled uniformly at random from $[n]$. For GLMs, $f_i = \phi_i(a_i^T w)$, so Eq. (1) simplifies to

$$P = \frac{1}{b_H} \sum_{i \in \mathcal{B}} \phi_i''(a_i^T w) a_i a_i^T + \rho I = \frac{1}{b_H} A_{\mathcal{B}}^T \Phi''(A_{\mathcal{B}} w) A_{\mathcal{B}} + \rho I. \tag{2}$$

If $b_H \geq p$, we may form and factor P (via Cholesky) in $\mathcal{O}(b_H p^2 + p^3)$ time and compute $P^{-1}v$ for $v \in \mathbb{R}^p$ in $\mathcal{O}(p^2)$ time via triangular solves. When $b_H \leq p$ (as is typical), we compute the Cholesky factorization $LL^T = \frac{1}{b_H} \Phi''(A_{\mathcal{B}} w)^{1/2} A_{\mathcal{B}} A_{\mathcal{B}}^T \Phi''(A_{\mathcal{B}} w)^{1/2} + \rho I$ and

$$P^{-1}v = \left(v - \frac{1}{b_H} A_{\mathcal{B}}^T \Phi''(A_{\mathcal{B}} w)^{1/2} L^{-T} L^{-1} \Phi''(A_{\mathcal{B}} w)^{1/2} A_{\mathcal{B}} v \right) / \rho$$

via the Woodbury formula (Higham, 2002). Lemma 1 summarizes the operational costs.

Lemma 1 *Let $v \in \mathbb{R}^p$ and let P be as in (2). If $b_H \leq p$, then the Cholesky factorization can be constructed in $\mathcal{O}(b_H^2 p + b_H^3)$ time and $P^{-1}v$ can be computed in $\mathcal{O}(b_H p)$ time. Furthermore, if the data matrix A is row-sparse with sparsity parameter s, the computational cost of $P^{-1}v$ can be reduced to $\mathcal{O}(b_H s)$ time.*

The $\mathcal{P}_{\text{ssn}}$ class (Table 3 and Algorithms 1 and 2) provides an implementation of the SSN preconditioner. The attributes of the $\mathcal{P}_{\text{ssn}}$ class are given in Table 3 and pseudocode for the `update` and `direction` methods is provided in Algorithms 1 and 2, respectively.

Attribute	Description
ρ	Regularization for preconditioner
b	Size of Hessian batch used for preconditioner construction
X	Square root of subsampled Hessian (excluding l^2-regularization)
L	Lower-triangular Cholesky factor for storing preconditioner
$\lambda_{\mathcal{P}}$	Estimate of preconditioned smoothness constant

Table 3: Attributes of the $\mathcal{P}_{\text{ssn}}$ class.

The `update` method takes a GLM $\mathcal{M}$, Hessian batches $\mathcal{B}_1, \mathcal{B}_2$, and vector $w \in \mathbb{R}^p$ as input. In the first phase, this method constructs the SSN preconditioner P at w by computing the square root of the subsampled Hessian, followed by an appropriate Cholesky

Algorithm 1 Update $\mathcal{P}_{\text{ssn}}$ preconditioner and preconditioned smoothness constant

```
Require: $\mathcal{P}_{\text{ssn}}$ object with attributes $\rho, b, X, L, \lambda_{\mathcal{P}}$
  function $\mathcal{P}_{\text{ssn}}$.update($\mathcal{M}, \mathcal{B}_1, \mathcal{B}_2, w$)
    $\rho \leftarrow \mathcal{P}_{\text{ssn}}.\rho$                                    ▷ Get attributes

    # Phase 1: Update preconditioner
    $A_{\text{sub}} \leftarrow \mathcal{M}$.get_data($\mathcal{B}_1$)
    $d_{\text{sub}} \leftarrow \mathcal{M}$.get_hessian_diagonal($\mathcal{B}_1, w$)
    $X \leftarrow \text{diag}(\sqrt{d_{\text{sub}}})A_{\text{sub}}$                  ▷ Square root of subsampled Hessian
    if $|\mathcal{B}_1| \geq p$ then
      $L \leftarrow$ cholesky($X^T X + \rho I$)
    else
      $L \leftarrow$ cholesky($X X^T + \rho I$)
    end if

    # Phase 2: Update estimated preconditioned smoothness constant
    $A_{\text{sub}} \leftarrow \mathcal{M}$.get_data($\mathcal{B}_2$)
    $d_{\text{sub}} \leftarrow \mathcal{M}$.get_hessian_diagonal($\mathcal{B}_2, w$)
    $Z \leftarrow A_{\text{sub}}^T \text{diag}(d_{\text{sub}}) A_{\text{sub}} + \mathcal{M}$.get_reg()$I$     ▷ Subsampled Hessian
    $\lambda_{\mathcal{P}} \leftarrow$ eig($Z(X^T X + \rho I)^{-1}, k = 1$)        ▷ Compute largest eigenvalue

    $\mathcal{P}_{\text{ssn}}.b \leftarrow |\mathcal{B}_1|, \mathcal{P}_{\text{ssn}}.X \leftarrow X, \mathcal{P}_{\text{ssn}}.L \leftarrow L, \mathcal{P}_{\text{ssn}}.\lambda_{\mathcal{P}} \leftarrow \lambda_{\mathcal{P}}$     ▷ Set attributes
```

Algorithm 2 Compute $\mathcal{P}_{\text{ssn}}$ direction

```
Require: $\mathcal{P}_{\text{ssn}}$ object with attributes $\rho, b, X, L, \lambda_{\mathcal{P}}$
  function $\mathcal{P}_{\text{ssn}}$.direction($g$)
    $b \leftarrow \mathcal{P}_{\text{ssn}}.b, L \leftarrow \mathcal{P}_{\text{ssn}}.L, X \leftarrow \mathcal{P}_{\text{ssn}}.X$       ▷ Get attributes
    if $b \geq p$ then
      $v \leftarrow L^{-1} g$                                        ▷ Triangular solve
      $v \leftarrow L^{-T} v$                                        ▷ Triangular solve
      return $v$
    else
      $v \leftarrow X g$
      $v \leftarrow L^{-1} v$                                        ▷ Triangular solve
      $v \leftarrow L^{-T} v$                                        ▷ Triangular solve
      $v \leftarrow X^T v$
      return $(g - v)/\rho$
    end if
```

factorization. The matrix used in the Cholesky factorization changes depending on the Hessian batchsize in order to obtain the computational costs in Lemma 1. In the second phase, this method estimates the preconditioned smoothness constant by computing $\lambda_1(P^{-1/2}\widehat{\nabla}^2 F(w) P^{-1/2}) = \lambda_1(\widehat{\nabla}^2 F(w) P^{-1})$. We never instantiate the subsampled Hessian to perform this calculation. Instead, we define matrix-vector products with the subsampled Hessian and inverse preconditioner and compute the largest eigenvalue via powering (our implementation uses `scipy.sparse.linalg.eigs`).

The `direction` method takes a vector $g \in \mathbb{R}^p$ (typically a stochastic gradient) as input. This method then computes $P^{-1}g$ using the Cholesky factor L and the square root of the subsampled Hessian X (as necessary). The reason for having two cases is to achieve the computational complexity in Lemma 1 by taking advantage of the Woodbury formula.

2.1.2 Nyström Subsampled Newton (NySSN)

NySSN combines the SSN preconditioner with randomized low-rank approximation, specifically the randomized Nyström approximation (Williams and Seeger, 2000; Gittens and Mahoney, 2016; Tropp et al., 2017). This approach was previously developed by in Frangella et al. (2023a) to precondition stochastic gradient descent. Given $H \in \mathbb{S}_p^+(\mathbb{R})$, the randomized Nyström approximation with respect to a random test matrix $\Omega \in \mathbb{R}^{p\times r}$ is given by

$$\hat{H} = (H\Omega)\left(\Omega^T H\Omega\right)^{\dagger}(H\Omega)^T. \tag{3}$$

Common choices for Ω include standard normal random matrices, subsampled randomized Hadamard transforms, and sparse sign embeddings (Tropp et al., 2017). The benefit of the latter two test matrices is that computation of the sketch $H\Omega$ becomes cheaper.

For a minibatch $\mathcal{B}$ ($|\mathcal{B}| = b_H$) and query point $w \in \mathbb{R}^p$, NySSN takes $H = \frac{1}{b_H}A_{\mathcal{B}}^T\Phi''(A_{\mathcal{B}}w)A_{\mathcal{B}}$ in (3) and produces a randomized low-rank approximation $\hat{H}$ of (the un-regularized portion of) the subsampled Hessian.

A practical algorithm for constructing the randomized low-rank approximation outputs $\hat{H}$ in the factored form $U\hat{\Lambda}U^T$, where $U \in \mathbb{R}^{p\times r}$ is an orthogonal matrix containing approximate eigenvectors and $\hat{\Lambda} \in \mathbb{R}^{r\times r}$ is a diagonal matrix containing approximate eigenvalues. We emphasize that this algorithm never forms $\frac{1}{b_H}A_{\mathcal{B}}^T\Phi''(A_{\mathcal{B}}w)A_{\mathcal{B}}$ explicitly. The resulting preconditioner is

$$P = \hat{H} + \rho I = U\hat{\Lambda}U^T + \rho I. \tag{4}$$

The dominant costs in the (practical) construction of P are computing the sketch $H\Omega$ and a SVD of a $p \times r$ matrix. Furthermore, we can compute $P^{-1}v$ via the Woodbury formula, which yields

$$P^{-1}v = U\left(\hat{\Lambda} + \rho I\right)^{-1}U^T v + \frac{1}{\rho}(v - UU^T v).$$

We summarize the costs of these operations in Lemma 2.

Lemma 2 *Let $v \in \mathbb{R}^p$ and let P be as in* (4)*. Then P can be constructed in $\mathcal{O}(b_H rp + r^2 p)$ time and $P^{-1}v$ can be computed in $\mathcal{O}(rp)$ time. For GLMs, $b_H \geq r$, so the construction cost is reduced to $\mathcal{O}(b_H rp)$.*

The main advantage of NySSN over SSN is in the setting where the data matrix A is dense and has rapid spectral decay. When A has rapid spectral decay, we can use a relatively small value of r ($r \leq 10$) to construct the NySSN preconditioner. With this small value of r, the $\mathcal{O}(rp)$ cost of applying the NySSN preconditioner to a vector is usually cheaper than the $\mathcal{O}(b_H p)$ cost of applying the SSN preconditioner. On the other hand, when A is row-sparse with sparsity parameter s, the cost of applying the SSN preconditioner to a vector is reduced to $\mathcal{O}(b_H s)$, which negates the speedups provided by the NySSN preconditioner.

Our code repository, linked in Section 6, implements the NySSN preconditioner.

2.1.3 Sketch-and-Solve Subsampled Newton (SASSN)

The next preconditioning technique we discuss is SASSN. Similar to NySSN, the fundamental goal of SASSN is to reduce the cost of the SSN preconditioner by replacing it with a randomized low-rank approximation. However, instead of using the randomized Nyström approximation, SASSN computes an approximation in the style of the Newton Sketch (Pilanci and Wainwright, 2017; Lacotte et al., 2021). To start, observe that the subsampled

Hessian $\widehat{\nabla}^2 F(w)$ has the form

$$\widehat{\nabla}^2 f(w) + \nu I = R^T R + \nu I,$$

where $R = \Phi''(A_{\mathcal{B}} w)^{1/2} A_{\mathcal{B}}$. Hence, given a test matrix $\Omega \in \mathbb{R}^{r \times b_H}$, we construct the preconditioner

$$P = R^T \Omega^T \Omega R + \rho I. \tag{5}$$

The dominant costs in the construction of P are computing the sketch ΩR and a Cholesky factorization of $\Omega R (\Omega R)^T + \rho I$. Taking Ω to be a column-sparse or row-sparse (LESS-uniform) embedding (Derezinski et al., 2021), ΩR can be computed in $\mathcal{O}(b_H p)$ time. A preconditioner generated by a column-sparse embedding is referred to as SASSN-C, while a preconditioner generated by a row-sparse embedding is referred to as SASSN-R. SASSN-R tends to be better than SASSN-C because it is cheaper to apply to vectors when the data matrix A is row-sparse. Similar to the NySSN preconditioner, we can compute $P^{-1}v$ via the Woodbury Formula, which yields

$$P^{-1} v = \frac{1}{\rho} \left(v - (\Omega R)^T \left(\Omega R (\Omega R)^T + \rho I \right)^{-1} (\Omega R) v \right).$$

We summarize the costs of these operations in Lemma 3.

Lemma 3 *Let $v \in \mathbb{R}^p$ and let P be as in (5). Then P can be constructed in $\mathcal{O}(b_H p + r^2 p + r^3)$ time and P^{-1} may be applied to vectors in $\mathcal{O}(rp)$ time. Furthermore, if the data matrix A is row-sparse with sparsity parameter s, the computational cost of $P^{-1}v$ for SASSN-R can be reduced to $\mathcal{O}(rs)$ time.*

Similar to NySSN, the costs of constructing and applying the SASSN preconditioner are lower than the costs incurred by SSN. A potential advantage of SASSN over NySSN is that SASSN requires $\mathcal{O}(b_H p + r^2 p + r^3)$ time to construct the preconditioner, whereas NySSN requires $\mathcal{O}(b_H r p)$. Furthermore, the SASSN-R preconditioner takes $\mathcal{O}(rs)$ time to apply when the data matrix A is row-sparse, whereas NySSN takes $\mathcal{O}(rp)$.

However, our experiments (Section 6) suggest that the SASSN preconditioner tends to be of lower quality than the NySSN preconditioner (i.e., it does not reduce the condition number as much), and the theoretical complexity advantage of SASSN is not always realized as the computations in NySSN benefit from (embarassing) parallelism. Concrete comparisons and recommendations between SSN, NySSN, SASSN-C, and SASSN-R are given in Tables 4 and 5 below. An implementation of the SASSN-C/SASSN-R preconditioners can be found in our code repository, which is linked in Section 6.

2.2 Preconditioner Defaults and Comparisons

All of the proposed preconditioners require a regularization ρ, and the NySSN, SASSN-R, and SASSN-C preconditioners require a rank r for forming the low-rank approximation. We recommend setting $\rho = 10^{-3}$ and $r = 10$. We also summarize the costs to construct and apply each preconditioner in Table 4. These costs assume the Hessian batchsize $b_H = \lfloor \sqrt{n} \rfloor$ in the PROMISE algorithms; we provide motivation for this selection in Section 2.3. We also provide guidelines for which preconditioner to use as the problem size varies in Table 5; we do not recommend SASSN-C in practice.

Preconditioner	Construction cost	Cost to apply	Cost to apply (sparse)
SSN	$\mathcal{O}(np + n^{3/2})$	$\mathcal{O}(\sqrt{n}p)$	$\mathcal{O}(\sqrt{n}s)$
NySSN	$\mathcal{O}(\sqrt{n}rp)$	$\mathcal{O}(rp)$	$\mathcal{O}(rp)$
SASSN-C	$\mathcal{O}(\sqrt{n}p)$	$\mathcal{O}(rp)$	$\mathcal{O}(rp)$
SASSN-R	$\mathcal{O}(\sqrt{n}p)$	$\mathcal{O}(rp)$	$\mathcal{O}(rs)$

Table 4: Summary of costs of proposed preconditioners. s denotes the row sparsity of the data matrix A.

Regime	SSN	NySSN	SASSN-R
$n \gg p$ (dense)	2	**1**	3
$n \gg p$ (sparse)	**1**	2	3
$n \sim p$ (dense)	3	**1**	2
$n \sim p$ (sparse)	**1**	3	2
$n \ll p$ (sparse)	**1**	3	2

Table 5: Guidelines for selecting a preconditioner. The best preconditioner for each regime is assigned a rank of 1. NySSN is effective for dense problems, but SSN generally works better for sparse problems because it preserves the sparsity of the data.

2.3 Quality of the Preconditioners

We now analyze the quality of the SSN, NySSN, and SASSN preconditioners that were introduced in Section 2.1. Our goal is to show that these preconditioners satisfy the following *ζ-spectral approximation property* with high probability.

Definition 4 (ζ-spectral approximation) *Let $w \in \mathbb{R}^p$, $\zeta \in (0,1)$. Then we say P is a* ζ-spectral approximation *of $\nabla^2 F(w)$ if the following relation holds:*

$$(1-\zeta)P \preceq \nabla^2 F(w) \preceq (1+\zeta)P. \tag{6}$$

If P satisfies Definition 4, then

$$\kappa(P^{-1/2}\nabla^2 F(w)P^{-1/2}) \leq \frac{1+\zeta}{1-\zeta}.$$

Hence preconditioning $\nabla^2 F(w)$ by P results in a good (small) condition number for moderate ζ (e.g., $\zeta \leq .9$). Moreover, as $P^{-1/2}\nabla^2 F(w)P^{-1/2}$ is nearly the identity, P^{-1} is close to $\nabla^2 F(w)^{-1}$, which ensures the approximate Newton direction computed with P^{-1} is close to the true Newton direction. As a consequence of this last observation, essentially all works on approximate Newton methods require the Hessian approximation to satisfy the conditions of Definition 4 (Pilanci and Wainwright, 2017; Roosta-Khorasani and Mahoney, 2019; Marteau-Ferey et al., 2019a; Ye et al., 2021).

2.3.1 Preliminaries on sampling

To establish the ζ-approximation property for the preconditioners, we require some fundamental concepts from matrix approximation via random sampling, which we now review. We start with the definition of ridge leverage scores (Cohen et al., 2017; Li et al., 2020).

Definition 5 (Ridge leverage scores) *Let $\nu \geq 0$ and $i \in [n]$. Then the ith* ridge leverage score *of a matrix $A \in \mathbb{R}^{n\times p}$ is given by*

$$l_i^\nu(A) := \frac{1}{n}a_i^T\left(\frac{1}{n}A^TA+\nu I\right)^\dagger a_i.$$

where a_i^T is the ith row of A. The maximum *ridge leverage score is $l_\infty^\nu(A) := \max_{1\leq i\leq n} l_i^\nu(A)$.*

The *ith* ridge leverage score measures the importance of row i in the matrix A. These scores play a crucial role in determining how well the matrix $\frac{1}{n}A^TA+\nu I$ may be approximated via uniform sampling. To understand this relation, we recall the notions of effective dimension and ridge leverage incoherence.

Definition 6 (Effective dimension and ridge leverage coherence) *Given $A \in \mathbb{R}^{n\times p}$ and $\nu \geq 0$, the* effective dimension *of A is given by*

$$d_{\text{eff}}^\nu(A) := \sum_{i=1}^n l_i^\nu(A) = \sum_{j=1}^p \frac{\frac{1}{n}\sigma_j^2(A)}{\frac{1}{n}\sigma_j^2(A)+\nu} = \sum_{j=1}^p \frac{\frac{1}{n}\lambda_j(A^TA)}{\frac{1}{n}\lambda_j(A^TA)+\nu}. \tag{7}$$

If $H \in \mathbb{S}_p^+(\mathbb{R})$ with $H = \frac{1}{n}A^TA$, then we overload notation and define $d_{\text{eff}}^\nu(H) := d_{\text{eff}}^\nu(A)$. The ridge leverage coherence *is given by*

$$\chi^\nu(A) := \frac{n}{d_{\text{eff}}^\nu(A)}l_\infty^\nu(A). \tag{8}$$

Similarly, if $H \in \mathbb{S}_p^+(\mathbb{R})$ with $H = \frac{1}{n}A^TA$, we overload notation and define $\chi^\nu(H) := \chi^\nu(A)$.

Effective dimension: discussion. The effective dimension $d_{\text{eff}}^\nu(A)$ has an intuitive interpretation: it provides a smoothed count of the eigenvalues greater than or equal to the regularization ν. In the regularized setting, only directions associated with eigenvalues larger than ν matter, so $d_{\text{eff}}^\nu(A)$ rather than p is the relevant measure of degrees of freedom for the problem. Consequently, the effective dimension often appears in fields that deal with l^2-regularized problems, including non-parametric learning, RandNLA, and statistical learning (Caponnetto and De Vito, 2007; Hsu et al., 2014; Marteau-Ferey et al., 2019b). As many data matrices have fast spectral decay (Derezinski et al., 2020), or obtain it through some algorithmic transformation, such as the celebrated random features method of Rahimi and Recht (2007), $d_{\text{eff}}^\nu(A)$ is often much smaller than $\min\{n,p\}$.

When the loss function f belongs to the GLM family and A has polynomial spectral decay, the following lemma demonstrates that the effective dimension of the Hessian, $\frac{1}{n}A^T\Phi''(Aw)A$, is much smaller than the ambient dimension p. Various results of this form are well-known in the literature, see for instance Caponnetto and De Vito (2007); Bach (2013); Marteau-Ferey et al. (2019a).

Lemma 7 (Effective dimension under polynomial decay) *Let f be a GLM loss satisfying $\sup_{w\in\mathbb{R}}\phi''(w) \leq B$, with data matrix $A \in \mathbb{R}^{n\times p}$ and regularization ν. Suppose the empirical covariance matrix $\frac{1}{n}A^TA$ has polynomial (or faster) spectral decay:*

$$\frac{1}{n}\lambda_j(A^TA) \leq Cj^{-2\beta} \quad (1 \leq j \leq p),$$

for some $C > 0$ and $\beta \in \mathbb{Z}_+$ satisfying $\beta \geq 1$. Then for any $w \in \mathbb{R}^p$,

$$d_{\text{eff}}^{\nu}\left(\frac{1}{n}A^T\Phi''(Aw)A\right) \leq \frac{\pi/(2\beta)}{\sin(\pi/(2\beta))}\left(\frac{BC}{\nu}\right)^{1/2\beta}.$$

Hence, if $\nu = \mathcal{O}(\frac{1}{n})$ we have

$$d_{\text{eff}}^{\nu}\left(\frac{1}{n}A^T\Phi''(Aw)A\right) = \mathcal{O}\left(\sqrt{n}\right).$$

We provide a proof of Lemma 7 in Appendix B.1. Given mild hypotheses, Lemma 7 shows that the effective dimension of the Hessian for GLMs is $\mathcal{O}(\sqrt{n})$. Thus, we generally expect the effective dimension of the Hessian, $d_{\text{eff}}^{\nu}\left(\frac{1}{n}A^T\Phi''(Aw)A\right)$, to be significantly smaller than the ambient dimension of the problem, p. The "smallness" of the effective dimension has been exploited in numerous works to develop fast algorithms for solving a variety of machine learning problems (Bach, 2013; Alaoui and Mahoney, 2015; Rudi et al., 2017; Marteau-Ferey et al., 2019a; Lacotte et al., 2021; Zhao et al., 2022; Frangella et al., 2023b). Similar to these prior works, we will also exploit the small effective dimension of the Hessian to develop effective preconditioners that can be constructed at negligible cost.

We now establish that the various preconditioning methods introduced in Section 2.1 provide a ζ-spectral approximation with high probability.

2.3.2 Subsampled Newton

SSN yields a ζ-spectral approximation for GLMs with high probability, formalized below.

Proposition 8 *Let $w \in \mathbb{R}^p$, $\zeta_0 \in (0,1)$, and suppose f is a GLM. Construct the subsampled Hessian with batchsize $b_H = \Omega\left(\frac{\chi^{\rho}(\nabla^2 f(w))d_{\text{eff}}^{\rho}(\nabla^2 f(w))\log\left(\frac{d_{\text{eff}}^{\rho}(\nabla^2 f(w))}{\delta}\right)}{\zeta_0^2}\right)$. Then for $\zeta = 1 - (1-\zeta_0)\nu/\rho$, with probability at least $1-\delta$,*

$$(1-\zeta)(\widehat{\nabla}^2 f(w) + \rho I) \preceq \nabla^2 f(w) + \nu I \preceq (1+\zeta)(\widehat{\nabla}^2 f(w) + \rho I). \qquad (9)$$

Proposition 8 is well-known in the literature (Li et al., 2020). It shows that when $\chi^{\rho}(\nabla^2 f(w)) = \mathcal{O}(1)$, a batchsize of $b_H = \widetilde{\mathcal{O}}(d_{\text{eff}}^{\rho}(\nabla^2 f(w)))$ is sufficient to ensure that the subsampled Hessian is a ζ-spectral approximation. Furthermore, when the data matrix exhibits polynomial spectral decay, applying Lemma 7 reduces this requirement to $b_H = \widetilde{\mathcal{O}}(\sqrt{n})$. This latter reduction motivates our default hyperparameter setting $b_H = \lfloor\sqrt{n}\rfloor$ for the PROMISE algorithms in Section 3.

Proposition 8 should be contrasted with the Hessian batchsize requirements of works where the f_i's are taken to be arbitrary convex functions. To facilitate this comparison, we first state the following simple lemma.

Lemma 9 (d_{eff}^{ν} vs. $\kappa_{\max}$ for GLMs) *Let f be a GLM, $\nu > 0$, and $\kappa_{\max} = L_{\max}/\nu$. Then*

$$\chi^{\nu}(\nabla^2 f(w))d_{\text{eff}}^{\nu}(\nabla^2 f(w)) \leq \kappa_{\max}.$$

As stated above, some works (Roosta-Khorasani and Mahoney, 2019; Ye et al., 2021; Dereziński, 2022) assume the f_i's possess no structure aside from convexity, which leads to the batchsize requirement $b_H = \mathcal{O}(\kappa_{\max}\log(p/\delta)/\zeta_0^2)$. Lemma 9 shows that $\kappa_{\max} \geq \chi^\nu(\nabla^2 f(w))d_{\text{eff}}^\nu(\nabla^2 f(w))$, so the needed batchsize is always at least as large as the one prescribed by Proposition 8. Moreover, when the data matrix A is ill-conditioned, the gap in required batchsizes can be significant. As a concrete example, consider the setting of Lemma 7 with ridge leverage incoherent A: $\chi^\nu(\nabla^2 f(w))d_{\text{eff}}^\nu(\nabla^2 f(w)) = \mathcal{O}(\sqrt{n})$, while $\kappa_{\max} = \mathcal{O}(n)$. Hence, Proposition 8 predicts a small Hessian batchsize, while the requirement based on $\kappa_{\max}$ states the full Hessian must be used. Thus, the Hessian batchsize required for convex GLMs is considerably smaller than that for a sum of arbitrary convex functions.

2.3.3 Nyström Subsampled Newton

NySSN yields a ζ-spectral approximation for GLMs with high probability, formalized below.

Proposition 10 *Let $w \in \mathbb{R}^p$, $\zeta_0 \in (0,1)$, and suppose f is a GLM. Construct the subsampled Hessian with batchsize $b_H = \Omega\left(\frac{\chi^\nu(\nabla^2 f(w))d_{\text{eff}}^\nu(\nabla^2 f(w))\log\left(\frac{d_{\text{eff}}^\nu(\nabla^2 f(w))}{\delta}\right)}{\zeta_0^2}\right)$. Further, assume Ω is a Gaussian random matrix with $r = \Omega\left(\frac{d_{\text{eff}}^\rho(\widehat{\nabla}^2 f(w))+\log(\frac{1}{\delta})}{\zeta_0^2}\right)$ columns. Then for $\zeta = 1-(1-\zeta_0)\nu/\rho$, with probability at least $1-\delta$,*

$$(1-\zeta)(\hat{H}+\rho I) \preceq \nabla^2 f(w) + \nu I \preceq (1+\zeta)(\hat{H}+\rho I). \tag{10}$$

The regularization parameter ρ controls how much we may truncate the rank parameter r. As ρ increases, $d_{\text{eff}}^\rho(\widehat{\nabla}^2 f(w))$ decreases, so we can use a smaller value of r to construct the preconditioner; conversely, as ρ approaches ν, we must use a larger value of r.

We observe a trade-off: a smaller rank parameter leads to faster computation and less storage, but potentially a less effective preconditioner and slower convergence. In practice, this tradeoff is not as dramatic as the theory might suggest, and we find a rank of $r = 10$ provides excellent performance in a wide range of applications (Section 6).

2.3.4 Sketch-and-solve Subsampled Newton

An analogous result for the SASSN preconditioners can be found in the arxiv report.

2.4 Beyond GLMs?

At this juncture, it is natural to ask whether the preconditioners we present can be extended to settings beyond GLMs. From a theoretical perspective, the answer is yes: the approximation bounds in this section hold for general convex finite-sum objectives, with slight adjustments. For instance, the batchsize for SSN depends upon a more complicated version* of the Hessian dissimilarity parameter presented in Section 5.4.1. From a practical

*. For details, see Frangella et al. (2023a).

perspective, both SSN and NySSN are compatible with general convex finite-sum objectives; SASSN is not compatible with general convex finite-sum objectives, since it requires access to the square root of the stochastic Hessian.

3. Algorithms

In this section, we introduce the PROMISE algorithms SketchySVRG (Section 3.2), SketchySAGA (Section 3.3), and SketchyKatyusha (Section 3.4). A summary of these algorithms in provided in Table 6. Each algorithm is compatible with all four preconditioning methods (SSN, NySSN, SASSN-C, SASSN-R) described in Section 2.1. Each algorithm comes with default hyperparameters, which we use in the empirical evaluation in Section 6. In particular, we describe how to automatically compute the learning rate for each algorithm using the estimated preconditioned smoothness constant. The learning rate is hard to tune in stochastic optimization, and it is remarkable that this automated selection works across a wide range of problems (Section 6). Finally, we recommend the best algorithm to use for two important applications, ridge and l^2-regularized logistic regression (Section 3.5).

Algorithm	Base Algorithm	Variance reduction	Acceleration	Stochastic gradients only?
SketchySVRG (Algorithm 3)	SVRG (Johnson and Zhang, 2013)	✓	✗	✗
SketchySAGA (Algorithm 4)	b-nice SAGA (Gazagnadou et al., 2019)	✓	✗	✓
SketchyKatyusha (Algorithm 5)	Loopless Katyusha (Kovalev et al., 2020)	✓	✓	✗

Table 6: Summary of algorithms in PROMISE. **Ticks** are pros while **crosses** are cons. SVRG and Katyusha require some full gradients rather than stochastic gradients only.

3.1 Notation in Algorithms

Throughout this section, $\mathcal{M}$ denotes a GLM with the oracles defined in Table 2. We use P to denote a preconditioner object, which is a member of one of the four preconditioner classes ($\mathcal{P}_{\text{ssn}}, \mathcal{P}_{\text{nyssn}}, \mathcal{P}_{\text{sassn-c}}, \mathcal{P}_{\text{sassn-r}}$). $\mathcal{U} \subseteq \mathbb{N}$ denotes a (possibly infinite) set of times that indicate when to update the preconditioner. We also use the index j to track the time when the preconditioner is constructed: every time the preconditioner is updated, the index j is updated to the most recently used element of $\mathcal{U}$. This index does not have to be tracked in the implementations of these algorithms, but it plays a key role in the theoretical analysis of the proposed algorithms (Section 5).

3.2 SketchySVRG

We formally introduce SketchySVRG in Algorithm 3.

Explanation of algorithm. SketchySVRG is a preconditioned version of SVRG (Johnson and Zhang, 2013). Similar to SVRG, SketchySVRG consists of an "outer" and "inner" loop indexed by s and k, respectively.

Algorithm 3 SketchySVRG

```
Require: initialization $\hat{w}_0$, gradient and Hessian batchsizes $b_g$ and $b_H$, preconditioner object $P$, model $\mathcal{M}$, preconditioner update times $\mathcal{U}$, learning rate multiplier $\alpha$, snapshot update frequency $m$
Initialize: snapshot $\hat{w} \leftarrow \hat{w}_0$
  for $s = 0, 1, \ldots$ do                                        ▷ Outer loop
    $\bar{g} \leftarrow \mathcal{M}.\texttt{get\_full\_grad}(\hat{w})$      ▷ Full gradient at snapshot
    $w_0 \leftarrow \hat{w}$
    for $k = 0, 1, \ldots, m-1$ do                                 ▷ Inner loop
      if $ms + k \in \mathcal{U}$ then                             ▷ Update preconditioner & learning rate
        Sample independent batches $\mathcal{S}_k^1, \mathcal{S}_k^2$      ▷ $|\mathcal{S}_k^1| = |\mathcal{S}_k^2| = b_H$
        $P.\texttt{update}(\mathcal{M}, \mathcal{S}_k^1, \mathcal{S}_k^2, w_k)$   ▷ Compute preconditioner $P_j$ at $w_k$ & update $P.\lambda_\mathcal{P}$
        $\eta \leftarrow \alpha / P.\lambda_\mathcal{P}$                    ▷ Update learning rate
      end if
      Sample batch $\mathcal{B}_k$                                  ▷ $|\mathcal{B}_k| = b_g$
      $\widehat{\nabla} F(w_k) \leftarrow \mathcal{M}.\texttt{get\_stoch\_grad}(\mathcal{B}_k, w_k)$
      $\widehat{\nabla} F(\hat{w}) \leftarrow \mathcal{M}.\texttt{get\_stoch\_grad}(\mathcal{B}_k, \hat{w})$
      $g_k \leftarrow \widehat{\nabla} F(w_k) - \widehat{\nabla} F(\hat{w}) + \bar{g}$   ▷ Unbiased estimate of $\nabla F(w_k)$
      $v_k \leftarrow P.\texttt{direction}(g_k)$                    ▷ Get approx. Newton step $P_j^{-1} g_k$
      $w_{k+1} \leftarrow w_k - \eta v_k$                            ▷ Update parameters
    end for
    Option I: $\hat{w} \leftarrow w_m$                                ▷ Update snapshot to final inner iterate
    Option II: $\hat{w} \leftarrow w_t$ for $t \sim \mathrm{Unif}(\{0, 1, \ldots, m-1\})$   ▷ Update snapshot randomly
  end for
```

The algorithm starts in the outer loop by computing a full gradient $\bar{g}$ at the snapshot $\hat{w}$, which is critical for performing variance reduction. The algorithm then sets the first iterate in the inner loop, w_0, equal to $\hat{w}$.

The inner loop of the algorithm updates the parameters with a preconditioned, variance-reduced stochastic gradient, v_k. SketchySVRG uses the preconditioner update times $\mathcal{U}$ to determine when the preconditioner and learning rate should be updated.

After m iterations of the inner loop, the algorithm returns to the outer loop and updates the snapshot $\hat{w}$ by either using the final inner iterate w_m (Option I) or sampling the previous m iterates uniformly randomly (Option II). In practice, we use Option I, but the theoretical analysis is conducted using Option II (Section 5.5). This discrepancy also appears in the original SVRG analysis (Johnson and Zhang, 2013) and is therefore not a drawback of the analysis in this paper.

Default hyperparameters. SketchySVRG's key hyperparameters include gradient and Hessian batch sizes b_g and b_H, preconditioner update times $\mathcal{U}$, learning rate multiplier α, and snapshot update frequency m. For gradient batch size b_g, we suggest 256 for medium and 4096 for large data sets, while b_H should be $\lfloor \sqrt{n} \rfloor$, as motivated in Section 2.3. We recommend setting $\mathcal{U} = \{0, u, 2u, \ldots\}$, where $u = \infty$ for problems with a constant Hessian (i.e., we only update the preconditioner once in total), like least squares/ridge regression, and $u = \lceil n/b_g \rceil$ (i.e., update the preconditioner after each pass through the data set) for problems with a non-constant Hessian, such as logistic regression. We recommend $\alpha \in [1/3, 1/2]$; our practical implementation uses the SAGA-inspired update rule $\eta \leftarrow \max\{1/(2(\nu n + P.\lambda_\mathcal{P})), 1/(3P.\lambda_\mathcal{P})\}$. We recommend snapshot update frequency $m \in [n/b_g, 2n/b_g]$ to compute a full gradient every one or two passes through the data set; our experiments set $m = \lceil n/b_g \rceil$.

3.3 SketchySAGA

We formally introduce SketchySAGA in Algorithm 4.

Algorithm 4 SketchySAGA

```
Require: initialization $w_0$, gradient and Hessian batchsizes $b_g$ and $b_H$, preconditioner object $P$, model $\mathcal{M}$, preconditioner update times $\mathcal{U}$, learning rate multiplier $\alpha$
Initialize: gradient table $\psi_0 \leftarrow 0 \in \mathbb{R}^{p \times n}$, table avg. $x_0 \leftarrow \frac{1}{n}\psi_0 \mathbf{1}_n \in \mathbb{R}^p$

for $k = 0, 1, \ldots$ do
    if $k \in \mathcal{U}$ then                                              ▷ Update preconditioner & learning rate
        Sample independent batches $\mathcal{S}_k^1, \mathcal{S}_k^2$        ▷ $|\mathcal{S}_k^1| = |\mathcal{S}_k^2| = b_H$
        $P.\texttt{update}(\mathcal{M}, \mathcal{S}_k^1, \mathcal{S}_k^2, w_k)$   ▷ Compute preconditioner $P_j$ at $w_k$ & update $P.\lambda_{\mathcal{P}}$
        $\eta \leftarrow \alpha / P.\lambda_{\mathcal{P}}$                   ▷ Update learning rate
    end if
    Sample batch $\mathcal{B}_k$                                             ▷ $|\mathcal{B}_k| = b_g$
    aux $\leftarrow \sum_{i \in \mathcal{B}_k} (\mathcal{M}.\texttt{get\_stoch\_grad}(i, w_k) - \psi_k^i)$
    $g_k \leftarrow x_k + \frac{1}{|\mathcal{B}_k|}$aux                      ▷ Unbiased estimate of $\nabla F(w_k)$
    $x_{k+1} \leftarrow x_k + \frac{1}{n}$aux                                ▷ Update table average
    $\psi_{k+1}^i \leftarrow \begin{cases} \psi_k^i, & i \notin \mathcal{B}_k \\ \mathcal{M}.\texttt{get\_stoch\_grad}(i, w_k), & i \in \mathcal{B}_k \end{cases}$   ▷ Update table columns for all $i \in [n]$
    $v_k \leftarrow P.\texttt{direction}(g_k)$                               ▷ Get approx. Newton step $P_j^{-1} g_k$
    $w_{k+1} \leftarrow w_k - \eta v_k$                                      ▷ Update parameters
end for
```

Explanation of algorithm. SketchySAGA, a minibatch variant of SAGA with preconditioning, updates the preconditioner and learning rate at specified times in $\mathcal{U}$. Each iteration involves computing stochastic gradients for each index in batch $\mathcal{B}_k$, which then update an auxiliary vector

$$\text{aux} := \sum_{i \in \mathcal{B}_k} \left(\nabla f_i(w_k) - \psi_k^i \right).$$

This auxiliary vector aids in updating the variance-reduced stochastic gradient g_k and the table average x_{k+1}. The gradient table ψ is updated accordingly; if an index i is in the batch, its row in ψ gets updated with the stochastic gradient (essential for variance reduction), otherwise it remains unchanged.

SketchySAGA then calculates the preconditioned variance-reduced stochastic gradient v_k for parameter updates, eliminating the need for full gradient computations and making it efficient for large-scale GLMs. The memory usage is dominated by the gradient table ψ, which typically requires $\mathcal{O}(np)$ storage. However, this can be reduced to $\mathcal{O}(n)$ for GLMs (Defazio et al., 2014). Implementing this storage optimization involves straightforward modifications to the updates of aux and ψ, and separating the regularization term νw from the stochastic gradient calculation. This improved algorithm is used in our experiments.

Default hyperparameters. The main hyperparameters in SketchySAGA are the gradient and Hessian batchsizes b_g and b_H, preconditioner update times $\mathcal{U}$, and learning rate multiplier α. We recommend setting b_g, b_H, and $\mathcal{U}$ similar to SketchySVRG. Furthermore, we recommend setting $\alpha \in [1/3, 1/2]$, although our practical implementation again uses $\eta \leftarrow \max\{1/(2(\nu n + P.\lambda_{\mathcal{P}})), 1/(3P.\lambda_{\mathcal{P}})\}$.

3.4 SketchyKatyusha

We formally introduce SketchyKatyusha in Algorithm 5.

Algorithm 5 SketchyKatyusha

```
Require: initialization $w_0$, gradient and Hessian batchsizes $b_g$ and $b_H$, preconditioner object $P$, model $\mathcal{M}$, preconditioner update times $\mathcal{U}$, momentum multiplier $\alpha$, momentum parameter $\theta_2$, snapshot update probability $\pi$, strong convexity parameter $\mu$
Initialize: snapshot $y \leftarrow w_0$, $z_0 \leftarrow w_0$, full gradient $\bar{g} \leftarrow \mathcal{M}.\texttt{get\_full\_grad}(w_0)$

  for $k = 0, 1, \ldots$ do
      if $k \in \mathcal{U}$ then                                          ▷ Update preconditioner & learning rate
          Sample independent batches $\mathcal{S}_k^1, \mathcal{S}_k^2$                ▷ $|\mathcal{S}_k^1| = |\mathcal{S}_k^2| = b_H$
          $P.\texttt{update}(\mathcal{M}, \mathcal{S}_k^1, \mathcal{S}_k^2, w_k)$     ▷ Compute preconditioner $P_j$ at $w_k$ & update $P.\lambda_{\mathcal{P}}$
          $L \leftarrow P.\lambda_{\mathcal{P}}$
          $\sigma \leftarrow \mu/L$                                       ▷ Estimate of inverse condition number
          $\theta_1 \leftarrow \min(\sqrt{\alpha n \sigma}, 1/2)$                       ▷ Update momentum parameter
          $\eta \leftarrow \frac{\theta_2}{(1+\theta_2)\theta_1}$                                   ▷ Update learning rate
      end if
      $x_k \leftarrow \theta_1 z_k + \theta_2 y + (1 - \theta_1 - \theta_2) w_k$                ▷ "Negative momentum" step
      Sample batch $\mathcal{B}_k$                                            ▷ $|\mathcal{B}_k| = b_g$
      $\widehat{\nabla} F(x_k) \leftarrow \mathcal{M}.\texttt{get\_stoch\_grad}(\mathcal{B}_k, x_k)$
      $\widehat{\nabla} F(y) \leftarrow \mathcal{M}.\texttt{get\_stoch\_grad}(\mathcal{B}_k, y)$
      $g_k \leftarrow \widehat{\nabla} f(x_k) - \widehat{\nabla} f(y) + \bar{g}$                       ▷ Unbiased estimate of $\nabla F(x_k)$
      $v_k \leftarrow P.\texttt{direction}(g_k)$                               ▷ Get approx. Newton step $P_j^{-1} g_k$
      $z_{k+1} \leftarrow \frac{1}{1+\eta\sigma}(\eta\sigma x_k + z_k - \frac{\eta}{L} v_k)$
      $w_{k+1} \leftarrow x_k + \theta_1(z_{k+1} - z_k)$                                  ▷ Update parameters
      Sample $U \sim \mathrm{Unif}([0, 1])$
      if $U \leq \pi$ then                        ▷ Update snapshot & full gradient with probability $\pi$
          $y \leftarrow w_k$
          $\bar{g} \leftarrow \mathcal{M}.\texttt{get\_full\_grad}(y)$
      end if
  end for
```

Explanation of algorithm. SketchyKatyusha, a preconditioned version of Loopless Katyusha (Kovalev et al., 2020), updates the preconditioner and learning rate at specified times in $\mathcal{U}$. The keys to its acceleration are the vectors x_k and z_k; at each iteration, a "negative momentum" step computes x_k as a convex combination of z_k, snapshot y, and current iterate w_k, which moderates x_k's deviation from y. This approach merges the advantages of variance reduction and acceleration.

Following this, SketchyKatyusha calculates the preconditioned variance-reduced stochastic gradient v_k and then z_{k+1}, and proceeds with a Nesterov momentum-like step to update the parameters w_{k+1}. It sporadically updates the snapshot y and full gradient $\bar{g}$ based on a probability π, enabling a simpler, single-loop implementation instead of Katyusha's original double-loop design (Allen-Zhu, 2018).

Default hyperparameters. The main hyperparameters in SketchyKatyusha are the gradient and Hessian batchsizes b_g and b_H, preconditioner update times $\mathcal{U}$, momentum multiplier α, momentum parameter θ_2, snapshot update probability π, and strong convexity parameter μ. We recommend setting b_g, b_H, and $\mathcal{U}$ similar to SketchySVRG. We recommend setting $\alpha = 2/3$, $\theta_2 = 1/2$, $\pi = b_g/n$, and $\mu = \nu$, where ν is the regularization parameter in the GLM.

3.5 Algorithm Recommendations

We present recommended algorithms for ridge regression and l^2-regularized logistic regression in Tables 7 and 8, respectively.

Data Regime	Recommendation (full gradients)	Recommendation (streaming ≤ 10 epochs)	Recommendation (streaming > 10 epochs)	Preconditioner
Dense	SketchyKatyusha	SketchySGD	SketchySAGA	NySSN
Sparse	SketchyKatyusha	SketchySGD	SketchySAGA	SSN

Table 7: Recommended algorithms for ridge regression. We recommend SketchySGD for streaming settings with limited computation (≤ 10 epochs) as SketchySAGA offers no significant advantage over short durations.

Data Regime	Recommendation (full gradients)	Recommendation (streaming ≤ 10 epochs)	Recommendation (streaming > 10 epochs)	Preconditioner
Dense	SketchyKatyusha SketchySAGA	SketchySGD	SketchySAGA	NySSN
Sparse	SketchyKatyusha SketchySAGA	SketchySGD	SketchySAGA	SSN

Table 8: Recommended algorithms for l^2-regularized logistic regression. We recommend SketchySGD for streaming settings with limited computation (≤ 10 epochs) as SketchySAGA offers no significant advantage over short durations.

4. Related Work

We review the literature on stochastic second-order and preconditioned stochastic gradient methods for solving (FSM), with emphasis on work that assumes strong convexity.

The deficiencies of the stochastic first-order methods presented in Section 1 are well-known within the optimization and machine learning communities. Indeed, in the past decade or so, research on stochastic second-order methods and stochastic preconditioning techniques for finite-sum optimization has exploded. Roughly, these methods can be divided into three categories: 1) stochastic second-order methods with full gradients, 2) stochastic second-order methods with stochastic gradients, and 3) preconditioned stochastic gradient methods. The dividing line between stochastic second-order methods and preconditioned methods is not always clear, as many preconditioners use second-order information, including the PROMISE framework. We review the literature on these three approaches in detail below.

4.1 Stochastic Second-order Methods with Full Gradients

We begin with stochastic second-order methods that use full gradients and a stochastic approximation to the Hessian. To the authors' knowledge, the earliest method of this form targeting (FSM) is Byrd et al. (2011). Byrd et al. (2011) subsample the Hessian and use this stochastic approximation in conjunction with an L-BFGS style update. Erdogdu and Montanari (2015); Roosta-Khorasani and Mahoney (2019) independently investigated the application of Newton's method to solve (FSM), where the Hessian is replaced with an approximation constructed through subsampling. The subsampled Newton method, as pioneered in these works, serves as the foundation for many subsequent developments in stochastic second-order methods.

In addition to introducing new algorithms, the works discussed above also provide analysis that lead to various convergence guarantees, which we now review. The analysis of Byrd et al. (2011) is quite coarse, only showing their method converges to the global optimum, provided the objective is strongly convex and the subsampled Hessian is always positive definite. Byrd et al. (2011) provides neither a convergence rate nor a theoretical advantage over first-order methods. The analyses of Erdogdu and Montanari (2015) and Roosta-Khorasani and Mahoney (2019) yield considerably stronger results. Both works establish linear convergence in the strongly convex setting. Furthermore, Roosta-Khorasani and Mahoney (2019) prove local superlinear convergence of Subsampled Newton, albeit under certain unattractive assumptions such as exponentially growing the Hessian batchsize b_H. Despite these assumptions, the results of Roosta-Khorasani and Mahoney (2019) point to potential benefits of stochastic second-order methods over first-order methods. We also note the analyses of these papers have been refined by Ye et al. (2021); Na et al. (2022). In particular, Na et al. (2022) propose a novel averaging scheme for the subsampled Hessian, which achieves local superlinear convergence without requiring a growing Hessian batchsize. Unfortunately, this approach is limited to settings where the dimension p of the feature vectors is modest, as it requires forming the subsampled Hessian for averaging, at a computational cost of $O(b_H p^2)$ and a storage cost of $O(p^2)$.

As an alternative to subsampling, some methods use sketching to construct a stochastic approximation to the full Hessian (Pilanci and Wainwright, 2017; Gower et al., 2019a; Lacotte et al., 2021). Sketching the Hessian has two main benefits over subsampling: (i) it generally produces higher-accuracy approximations to the Hessian (Martinsson and Tropp, 2020), and (ii) it is robust to the origins of the data. By robust, we mean that with an appropriate sketching matrix, the sketch size required to ensure the ζ-spectral approximation property is independent of the ridge leverage coherence. In detail, if the Hessian approximation is constructed from a sketching matrix belonging to an appropriate random ensemble, the sketch size required to ensure the ζ-spectral approximation property holds with high probability, is only $\widetilde{\mathcal{O}}(d_{\text{eff}}^{\nu}(A))$ (Lacotte et al., 2021). In contrast, the Hessian batchsize required by subsampling to ensure the ζ-spectral approximation property depends upon the ridge leverage coherence, which can be quite large when the data contains outliers.

However, sketching has several disadvantages relative to subsampling. A notable disadvantage of existing sketching-based methods is that they require a full pass through the data to approximate the Hessian, while subsampling methods do not. It is desirable to minimize full passes through the data when solving large-scale problems, which limits the usefulness of existing sketching-based methods in this setting. Another limitation of sketching-based methods is that they may only be applicable to problems with certain structure. As a concrete example, the Newton Sketch (Pilanci and Wainwright, 2017; Lacotte et al., 2021) requires access to a matrix R such that $\nabla^2 f(w) = R^T R$. While such a matrix is always available when f is a GLM, this is not the case for more general losses. In contrast, subsampling can always be used to approximate the Hessian of a finite-sum objective, regardless of the form of the loss function. Last, we note that if f is a GLM, then the Newton Sketch with a row-sampling sketching matrix is equivalent to Subsampled Newton.

The analysis guarantees of sketching-based (approximate) second-order methods are similar to their subsampled counterparts. Pilanci and Wainwright (2017); Lacotte et al. (2021) focus on self-concordant functions for their global convergence analysis and show fast linear convergence independent of the condition number. Additionally, Pilanci and Wainwright (2017) show local superlinear convergence for smooth and strongly convex objectives with Lipschitz Hessians, but require a sketch size that depends on the condition number of the problem, which can be larger than n when the problem is ill-conditioned. However, in the setting where p is moderate in size, this issue may be resolved by using the Hessian averaging scheme of Na et al. (2022). Gower et al. (2019a) prove convergence for functions that are relatively smooth and relatively convex, a generalization of smoothness and strong convexity to the local Hessian norm. They establish linear convergence with a rate that depends upon the *relative condition number* and the smallest non-zero eigenvalue of an expected projection matrix. When the objective is quadratic, the relative condition number equals 1 and so the convergence rate is independent of the condition number, which shows an improvement of over first-order methods.

4.2 Stochastic Second-order Methods with Stochastic Gradients

In scenarios where both n and p are large, full gradients become prohibitively expensive. Consequently, a scalable second-order method must use both stochastic gradients and stochastic Hessian approximations. To address this challenge, many methods that employ fully stochastic first- and second-order information have been proposed for solving (FSM). All such proposals employ subsampling-based approximations to the Hessian (Byrd et al., 2016; Moritz et al., 2016; Gower et al., 2016; Bollapragada et al., 2018; Roosta-Khorasani and Mahoney, 2019; Bollapragada et al., 2019; Wang and Zhang, 2019; Dereziński, 2022). These methods can be further categorized based on whether they directly compute the search direction by applying the inverse subsampled Hessian (Roosta-Khorasani and Mahoney, 2019; Bollapragada et al., 2019; Wang and Zhang, 2019; Dereziński, 2022), or by using the subsampled Hessian to stabilize an L-BFGS style update (Byrd et al., 2016; Moritz et al., 2016; Gower et al., 2016; Bollapragada et al., 2018). Hence most stochastic second-order methods that use stochastic gradients have their roots in the full-gradient methods of Byrd et al. (2011); Erdogdu and Montanari (2015); Roosta-Khorasani and Mahoney (2019).

The convergence guarantees of existing proposals vary greatly, often leaving much to be desired. Byrd et al. (2016) established an $\mathcal{O}(1/k)$-rate for their stochastic L-BFGS method assuming strong convexity. However, their analysis relies on two restrictive assumptions: (i) bounded variance of stochastic gradients and (ii) strict positive definiteness of the subsampled Hessian. The first is known to be false for strongly convex functions, unless the iterates lie in a compact set, and the second fails in common applications such as GLMs, where the subsampled Hessian is singular unless $b_H \geq p$. Roosta-Khorasani and Mahoney (2019) present a range of convergence results for a variety of settings, including when F is strongly convex, for which they establish fast local linear convergence. Hence the Subsampled Newton method enjoys the fast local convergence of Newton's method, albeit at a linear rather than quadratic rate. Nevertheless, the fast local convergence rate shows an advantage over stochastic first-order methods, whose convergence depends upon the condition number, regardless of how close the iterates are to the optimum. Unfortunately, in order to achieve their fast local convergence result, Roosta-Khorasani and Mahoney (2019) require an exponentially increasing gradient batchsize, and that the subsampled Hessian batchsize satisfy $b_H = \widetilde{\mathcal{O}}(\kappa/\epsilon^2)$, where $\epsilon \in (0,1)$. Hence the theoretical analysis requires rapidly growing gradient batchsizes and large Hessian batchsizes, which is antithetical to the purpose of stochastic methods.

The first work to obtain a linear rate of convergence for solving (FSM) without requiring large/growing gradient and Hessian batchsizes is Moritz et al. (2016). Moritz et al. (2016) combine the stochastic L-BFGS method of Byrd et al. (2016) with SVRG to reduce the variance of the stochastic gradients without growing the gradient batchsize. Although Moritz et al. (2016) proves global linear convergence, fast local linear convergence is not established, and no theoretical benefit over SVRG is demonstrated. Similar remarks hold for the stochastic L-BFGS methods of Gower et al. (2016); Bollapragada et al. (2018). More recently, Dereziński (2022) proposed Stochastic Variance Reduced Newton (SVRN), which combines Subsampled Newton with SVRG. Dereziński (2022) shows SVRN exhibits fast local linear convergence. However, the analysis requires the gradient batchsize to satisfy $b_g = \widetilde{\mathcal{O}}(\kappa)$, which is still very large, and can easily exceed n for ill-conditioned problems.

4.3 Preconditioned Stochastic Gradient Methods

Taking a general viewpoint, the stochastic second-order methods discussed above are all special cases of preconditioned stochastic gradient methods, where the current preconditioner P_k is based on an approximation to the Hessian matrix. An early notable proposal is the preconditioned SVRG algorithm by Gonen et al. (2016), which employs a preconditioner obtained through low-rank approximation to the Hessian using the randomized block Krylov method (Musco and Musco, 2015). Despite exhibiting improvements over SVRG, this approach requires multiple passes through the data matrix, hindering its suitability for larger problems. Another method, SVRG2 by Gower et al. (2018), combines SVRG with a preconditioner based on a randomized Nyström approximation to the Hessian, but also requires a costly full pass through the data at every outer iteration. Liu et al. (2019) propose preconditioned variants of SVRG and Katyusha using either the covariance matrix or a diagonal approximation. However, the former is impractical for large-scale settings, and the latter, although scalable, may perform poorly. Additionally, their fixed preconditioner approach may lead to suboptimal performance for non-quadratic problems.

4.4 Relation to PROMISE

The methods most closely related to PROMISE are Subsampled Newton (Roosta-Khorasani and Mahoney, 2019) and SVRN (Dereziński, 2022). When a PROMISE method uses the SSN preconditioner, it may be viewed as combining Subsampled Newton with the corresponding stochastic gradient algorithm. Variance reduction stabilizes PROMISE iterations to allow linear convergence without exponentially growing gradient batchsizes, unlike the batchsizes required by Subsampled Newton (Roosta-Khorasani and Mahoney, 2019). As SVRN is simply Subsampled Newton combined with SVRG, it follows that SketchySVRG equipped with the SSN preconditioner is equivalent to SVRN. However, despite this equivalence, the aims of this work and those of Dereziński (2022) are quite different. Dereziński (2022) focuses on proving fast local linear convergence under the hypotheses of large gradient minibatches: their results require $b_g = \widetilde{\mathcal{O}}(\kappa)$. Moreover, Dereziński (2022) only suggests using SVRN to finish off the optimization, and that in the beginning, Subsampled Newton with full gradients should be used to get the iterates sufficiently close to the optimum. This

Algorithm	b_g	b_H/ Sketch size	Lazy preconditioner updates	Fast local-linear convergence
SketchySVRG (Algorithm 3)	$\widetilde{\mathcal{O}}(\tau_\star^\nu)$	$\widetilde{\mathcal{O}}\left(\frac{\chi^\nu\left(\frac{1}{n}A^T\Phi''(Aw)A\right)d_{\text{eff}}^\nu\left(\frac{1}{n}A^T\Phi''(Aw)A\right)}{\zeta_0^2}\right)$	✓	✓
Subsampled Newton (Roosta-Khorasani and Mahoney, 2019)	Exponentially increasing	$\widetilde{\mathcal{O}}(\frac{\kappa_{\max}}{\zeta_0^2})$	✗	✓
Newton Sketch (Lacotte et al., 2021)	Full	$\widetilde{\mathcal{O}}\left(\frac{d_{\text{eff}}^\nu\left(\frac{1}{n}A^T\Phi''(Aw)A\right)}{\zeta_0^2}\right)$	✗	✓
SVRN (Dereziński, 2022)	$\widetilde{\mathcal{O}}(\kappa_{\max})$	$\widetilde{\mathcal{O}}\left(\frac{\kappa_{\max}}{\zeta_0^2}\right)$	✗	✓
SLBFGS (Moritz et al., 2016)	Constant	Constant	✗	✗
Progressive Batching L-BFGS (Bollapragada et al., 2018)	Increasing	Increasing	✗	✗

Table 9: Comparison of preconditioned stochastic gradient methods for solving (FSM) when F is a GLM. Here b_g and b_H are gradient and Hessian batchsizes, $\kappa_{\max} = L_{\max}/\nu$ is the condition number, $\chi^\nu\left(\frac{1}{n}A^T\Phi(Aw)A\right)$ and $d_{\text{eff}}^\nu\left(\frac{1}{n}A^T\Phi''(Aw)A\right)$ are the ridge-leverage coherence and effective dimension of the Hessian, while $\tau_\star^\nu$ denotes the Hessian dissimilarity. Note $\chi^\nu\left(\frac{1}{n}A^T\Phi''(Aw)A\right)d_{\text{eff}}^\nu\left(\frac{1}{n}A^T\Phi''(Aw)A\right)$ and $\tau_\star^\nu$ are never larger than $\kappa_{\max}$. Hence, of all the methods, SketchySVRG has the best required gradient and Hessian batchsizes, and is the only method whose theory accounts for lazy updates.

recommendation contrasts with this work, which shows global linear convergence, allows for lazy updates, and admits variance reduction schemes beyond SVRG. We also prove fast local linear convergence of SketchySVRG with moderate gradient batchsizes, a significant theoretical and practical improvement over the requirements of Dereziński (2022).

To facilitate a straightforward comparison between PROMISE and prior work, we present Table 9. Table 9 compares the properties of various stochastic second-order methods for solving (FSM) when F is a GLM. We select SketchySVRG as a representative for PROMISE. Inspection of Table 9 shows SketchySVRG is the method that enjoys the best batchsize requirements, while still attaining fast local linear convergence. Indeed, $\chi^\nu(\nabla^2 f(w))d_{\text{eff}}^\nu(\nabla^2 f(w))$ and $\tau_\star^\nu$ are always smaller than $\kappa_{\max}$ and n (Lemma 9, Lemma 14). Moreover, it is the only method in Table 9 whose theory accounts for lazy updates to the preconditioner, which is essential for good practical performance.

5. Theory

In this section we establish (global and local) linear convergence results for the PROMISE methods on smooth, strongly convex, finite-sum objectives (including but not restricted to

GLMs). This section begins with our assumptions and then introduces the key concepts of quadratic regularity and the quadratic regularity ratio, which generalize the notions of strong convexity, smoothness, and condition number to the Hessian norm. We follow this by introducing Hessian dissimilarity, which plays a key role in analyzing PROMISE methods with stochastic gradients. Finally, we state the main convergence theorems and provide a convergence proof for SketchySVRG which illustrates the techniques in our analysis. Our first theorem establishes global linear convergence, while our second theorem shows SketchySVRG achieves fast, local convergence independent of the condition number. Any proofs not provided in this section can be found in the arxiv report. Linear convergence results for SketchySAGA and SketchyKatyusha are also available in the arxiv report.

5.1 A Subtlety in Notation

We index the preconditioner in two different ways in this section; sometimes we denote the preconditioner by P_j, and other times we denote the preconditioner by P_k (or in the case of SketchySVRG, $P_k^{(s)}$). In this setting, j indexes the iterate where the preconditioner is constructed, while k (or ${}_k^{(s)}$) indexes the current iterate in the algorithm.

There is a simple way to map a P_k to the corresponding P_j. If the preconditioner update indices $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$, then a given P_k for $k \in \{u_i, u_i + 1, \ldots, u_{i+1} - 1\}$ is the same as P_j for $j = u_i$. As a concrete example, suppose $\mathcal{U} = \{0, 4, 10\}$. Then, for $k \in \{0, 1, 2, 3\}$, P_k is the same as P_j where $j = 0$; for $k \in \{4, 5, 6, 7, 8, 9\}$, P_k is the same as P_j where $j = 4$; for $k \in \{10, 11, \ldots\}$, P_k is the same as P_j where $j = 10$.

5.2 Assumptions

Here we provide assumptions that will be needed in the convergence analyses of the PROMISE methods.

Assumption 1 (Smoothness and convexity) *For each $i \in [n]$, $f_i(w)$ is L_i-smooth and convex.*

The above assumption is standard in the analysis of stochastic gradient methods for solving (FSM). This assumption is also needed to ensure *quadratic regularity* (a property that we introduce later in this section), which is key to showing convergence of our algorithms.

Assumption 2 (ζ-spectral approximation) *If the preconditioner P_j was constructed at w_j, where $j \in \mathcal{U}$, then*

$$(1 - \zeta)P_j \preceq \nabla^2 f(w_j) + \nu I \preceq (1 + \zeta)P_j,$$

where $\zeta \in (0, 1)$.

Assumption 2 states each preconditioner constructed by the algorithm satisfies the ζ-spectral approximation property, which is reasonable as PROMISE preconditioners satisfy this property with high probability. Assumption 2 can be viewed as conditioning on the event that the preconditioners constructed by the algorithm satisfy the ζ-spectral approximation property. There are two different strategies to make sure this event holds with high probability across the optimization trajectory:

- Fix the number of iterations in advance and apply a union bound.
- Let the failure probability decrease like $1/u^2$.

In practice, this is not necessary: we construct the preconditioner at each update time with the same Hessian batchsize and rank parameter.

5.3 Technical Preliminaries

5.3.1 Quadratic Regularity

We start by introducing the upper and lower quadratic regularity constants for a smooth, convex function $F : \mathcal{C} \mapsto \mathbb{R}$, where $\mathcal{C}$ is a closed convex subset of $\mathbb{R}^p$. These ideas are crucial for establishing linear convergence under infrequent updating of the preconditioner.

Definition 11 (Quadratic regularity) *Let F be a twice differentiable function, and $\mathcal{C}$ a closed convex set. Then we say F is $\mathcal{C}$-upper quadratically regular, if there exists $0 \leq \gamma_u(\mathcal{C}) < \infty$, such that for all $w_0, w_1, w_2 \in \mathcal{C}$,*

$$F(w_2) \leq F(w_1) + \langle \nabla F(w_1), w_2 - w_1 \rangle + \frac{\gamma_u(\mathcal{C})}{2} \|w_2 - w_1\|^2_{\nabla^2 F(w_0)}.$$

Similarly, we say F is $\mathcal{C}$-lower quadratically regular, if there exists $0 < \gamma_\ell(\mathcal{C})$, such that for all $w_0, w_1, w_2 \in \mathcal{C}$

$$F(w_2) \geq F(w_1) + \langle \nabla F(w_1), w_2 - w_1 \rangle + \frac{\gamma_\ell(\mathcal{C})}{2} \|w_2 - w_1\|^2_{\nabla^2 F(w_0)}.$$

We say F is $\mathcal{C}$-quadratically regular if $0 < \gamma_\ell$ and $\gamma_u < \infty$. Further, if F is $\mathcal{C}$-quadratically regular, we define the quadratic regularity ratio *to be*

$$\mathfrak{q}(\mathcal{C}) := \frac{\gamma_u(\mathcal{C})}{\gamma_\ell(\mathcal{C})}.$$

Moreover, if $F(w) = \frac{1}{n}\sum_{i=1}^n F_i(w)$, and each F_i is $\mathcal{C}$-quadratically regular, we denote the corresponding quadratic regularity constants by $\gamma_{u_i}(\mathcal{C})$ and $\gamma_{\ell_i}(\mathcal{C})$, and we define

$$\gamma_u^{\max}(\mathcal{C}) := \max_{i\in[n]} \gamma_{u_i}, \quad \gamma_\ell^{\min}(\mathcal{C}) := \min_{i\in[n]} \gamma_{\ell_i}.$$

Quadratic regularity holds whenever F can be upper- and lower-bounded in terms of the Hessian at any $w_0 \in \mathcal{C}$. Hence the upper and lower quadratic regularity constants may be viewed as global generalizations of the smoothness and strong convexity constants to the Hessian norm $\|\cdot\|_{\nabla^2 F(w)}$. Moreover, the quadratic regularity ratio $\mathfrak{q}$ generalizes the condition number κ to the Hessian norm. There is an explicit formulation of $\gamma_u(\mathcal{C})$ and $\gamma_\ell(\mathcal{C})$ in terms of F and $\mathcal{C}$; see Appendix A for more details.

Upper and lower quadratic regularity expand upon stable Hessians from Karimireddy et al. (2018) and its refinements relative smoothness and relative convexity from Gower et al. (2019a). The relative smoothness and relative convexity parameters from Gower et al. (2019a) are defined similarly to the quadratic regularity constants, except they have $w_0 = w_1$. Unfortunately, relative smoothness and relative convexity are insufficient for our analysis, which incorporates infrequent updating. Under infrequent updating, the preconditioner is constructed at a point $w_0 \neq w_1, w_2$. Relative smoothness only provides bounds in terms of $\nabla^2 F(w_1)$, whereas our analysis requires bounds in terms of $\nabla^2 F(w_0)$, i.e., the Hessian at the iterate where the preconditioner is constructed, which is exactly what quadratic regularity provides.

The additional power given by quadratic regularity could imply it only holds for a restrictive class of functions. However, the following proposition shows this is not the case, as quadratic regularity holds under many standard hypotheses, including smoothness and strong convexity.

Proposition 12 (Sufficient conditions for quadratic regularity) *The following conditions all imply F is $\mathcal{C}$-quadratically regular:*

1. *The function F is L-smooth and μ-strongly convex over $\mathcal{C}$. Then F is $\mathcal{C}$-quadratically regular with*
$$\frac{\mu}{L} \leq \gamma_\ell(\mathcal{C}) \leq \gamma_u(\mathcal{C}) \leq \frac{L}{\mu}.$$
2. *The function F is μ-strongly convex and has a M-Lipschitz Hessian over $\mathcal{C}$, and $\mathcal{C}$ is compact with diameter D. Then F is $\mathcal{C}$-quadratically regular with*
$$\left(1+\frac{MD}{\mu}\right)^{-1} \leq \gamma_\ell(\mathcal{C}) \leq \gamma_u(\mathcal{C}) \leq 1+\frac{MD}{\mu}.$$

Proposition 12 is similar to Theorem 1 in Karimireddy et al. (2018), which establishes analogous sufficient conditions for ensuring a stable Hessian. Moreover, the bounds attained on the quadratic regularity constants are identical to those attained in Karimireddy et al. (2018): this is notable since stable Hessians do not account for lazy updating, but our analysis does.

5.4 When does the Quadratic Regularity Ratio Improve over the Condition Number?

We now provide concrete examples of when quadratic regularity improves upon the condition number.

Convex quadratic functions. Let $F(w) = \frac{1}{2}w^T H w + b^T w + c$, where $H \in \mathbb{S}_p^+(\mathbb{R})$. Since F is quadratic, it has constant Hessian and it equals its own Taylor expansion. It immediately follows that $\gamma_\ell(\mathcal{C}) = \gamma_u(\mathcal{C}) = 1$. Hence $\mathfrak{q}(\mathcal{C}) = 1$, which is a significant improvement over $\kappa(H)$ when H is ill-conditioned.

Quasi-self concordant functions on a bounded domain. A function f is said to be M-quasi-self concordant (M-qsc) over $\mathcal{C}$ if
$$D^3F(x)[u,u,v] \leq M\|u\|^2_{\nabla^2 F(x)}\|v\| \quad \forall x \in \mathcal{C} \text{ and } \forall u,v \in \mathbb{R}^p,$$
where $D^3F(x)$ is the trilinear form representing the third derivative of F (Nesterov, 2018). Let $R > 0$ and suppose that $D = \mathrm{diam}(\mathcal{C}) \leq \log(R)/M$. Then we prove in Appendix B.2 that
$$\mathfrak{q}(\mathcal{C}) \leq R^2.$$

Any GLM (which includes non-quadratic problems like logistic and Poisson regression) with a data matrix A whose rows satisfy $\|a_i\| \leq 1$[*] for all $i \in [n]$ is 1-quasi-self-concordant (Karimireddy et al., 2018; Doikov, 2023). Thus, for $R = \mathrm{e}$, we have $\mathfrak{q}(\mathcal{C}) \leq 8$. In contrast, $\kappa(\mathcal{C}) = \Theta\left(\frac{\sigma^2_{\max}(A)+n\nu}{\sigma^2_{\min}(A)+n\nu}\right)$, which is large for ill-conditioned A. The bound for GLMs should be contrasted with item 2 of Proposition 12, where requires $D = \mathcal{O}(\nu/M)$ to ensure $\mathfrak{q}(\mathcal{C})$ is a small constant. As ν is typically very small, D will be very small, while for GLMs D can be as large as e and $\mathfrak{q}(\mathcal{C})$ will be a small constant. This shows that for objectives of interest, the iterates do not need to be in a tiny set about the optimum (where the objective is nearly quadratic) for the quadratic regularity ratio to be a constant independent of the condition number.

5.4.1 Hessian Dissimilarity

The next important idea is the Hessian dissimilarity, which quantifies how large the gradient batchsize must be to realize the benefits of preconditioning.

Definition 13 *Let $\mathcal{C}$ be a closed convex subset of $\mathbb{R}^p$. The* Hessian dissimilarity *is*

$$\tau_\star^\nu(\mathcal{C}) \coloneqq \sup_{w\in\mathcal{C}} \max_{1\leq i\leq n} \lambda_1\left((\nabla^2 f(w)+\nu I)^{-1/2}(\nabla^2 f_i(w)+\nu I)(\nabla^2 f(w)+\nu I)^{-1/2}\right).$$

The Hessian dissimilarity bounds the worst-case value over $\mathcal{C}$, of the norm ratio:

$$\frac{\|v\|^2_{\nabla^2 F_i(w)}}{\|v\|^2_{\nabla^2 F(w)}},$$

between the full Hessian $\nabla^2 F(w)$ and the Hessian of any term $\nabla^2 F_i(w)$ in the sum. Hessian dissimilarity is analogous to the ridge leverage coherence in Section 2.3: while ridge leverage coherence measures the uniformity of the rows of a matrix, the Hessian dissimilarity measures the uniformity of constituent psd Hessians of a finite-sum convex function $F(w)$. Furthermore, for GLMs, we will see that the Hessian dissimilarity is controlled by a uniform version of the ridge leverage coherence.

In our analysis of PROMISE, the Hessian dissimilarity parameter $\tau_\star^\nu$ appears in the bound on the preconditioned smoothness constant (Proposition 16), where it controls the gradient batchsize needed to ensure a good preconditioned smoothness constant. More precisely, it is the gradient batchsize required to ensure the preconditioned smoothness constant is $\mathcal{O}(1)$ on average, when the gradients are sampled uniformly at random. We believe this issue could be alleviated by importance sampling, but leave it as a direction for future work, as we have found uniform sampling to be sufficient in our experiments.

The Hessian dissimilarity never exceeds n, as shown by the following lemma.

Lemma 14 (Hessian dissimilarity never exceeds n) *Define $\kappa_{\max}$ is as in Lemma 9. The dissimilarity parameter satisfies*

$$1 \leq \tau_\star^\nu(\mathcal{C}) \leq \min\left\{n, 1+\kappa_{\max}\right\}.$$

[*] This is a standard normalization step employed in packages like scikit-learn for stochastic optimizers like SAGA.

Lemma 14 shows the the Hessian dissimilarity never exceeds n, and may be much smaller if the f_i's are well-conditioned. Unfortunately, for ill-conditioned problems, we can easily have $\tau_\star^\nu = n$, in which case there is no improvement in the gradient batchsize needed (in theory) over a full-gradient algorithm. In practice, large gradient batchsizes are unnecessary in any of our numerical experiments, which suggests the bound in Lemma 14 is pessimistic. In particular, this bound does not account for the structure of the objective. For GLMs, we can derive a more informative bound on the Hessian dissimilarity (see Proposition 15 below), which reveals a deep connection to ridge leverage scores and ridge leverage coherence.

Proposition 15 (Hessian dissimilarity for GLMs) *Consider minimizing a regularized GLM*

$$F(w) = \frac{1}{n}\sum_{i=1}^{n}\phi_i(a_i^T w) + \frac{\nu}{2}\|w\|^2.$$

Further suppose that $\sup_{x\in\mathbb{R}} \phi_i^{''}(x) \leq B$, *for some* $B > 0$. *Then*

$$\tau_\star^\nu \leq 1 + \chi_\star^\nu d_{\text{eff}}^{\nu/B}(A),$$

where $\chi_\star^\nu = \sup_{w\in\mathbb{R}^p} \chi^\nu(\Phi''(Aw)^{1/2}A)$ *and* $\Phi''(Aw) = \operatorname{diag}\left([\phi_1''(a_1^T w) \dots \phi_n''(a_n^T w)]\right)$. *In particular, for least squares and logistic regression,*

$$\tau_\star^\nu \leq 1 + \chi_\star^\nu d_{\text{eff}}^{\nu}(A).$$

Proposition 15 shows that for GLMs, the Hessian dissimilarity is controlled by the global ridge leverage coherence of the Hessian, $\chi_\star^\nu$. When $\chi_\star^\nu$ is close to 1, the batchsize required to see the full effects of preconditioning is not much larger than the effective dimension of the data matrix. As $d_{\text{eff}}^\nu(A)$ is much smaller than n under mild assumptions (recall Lemma 7), this implies that a much smaller gradient batchsize suffices to enjoy the effects of preconditioning. This improved theory agrees with our empirical results. Conversely, when the data matrix has high coherence, Proposition 15 suggests that large gradient batchsizes may be necessary to realize the benefits of preconditioning.

5.4.2 The Smoothness of the Preconditioned Stochastic Gradient

Convergence analysis of stochastic gradient methods requires control of the smoothness constant of the minibatch stochastic gradient. To analyze preconditioned methods, we must control the smoothness in the preconditioned norm ($\|\cdot\|_{P^{-1}}$) instead of the Euclidean norm ($\|\cdot\|_2$). The following proposition provides such control in expectation, in terms of the quantity $\mathcal{L}_P$, which we call the preconditioned expected smoothness constant. This proposition extends Proposition 3.8 of Gower et al. (2019b), which handles the case $P = I$.

Proposition 16 (Preconditioned expected smoothness) *Let* F *be* γ_u *upper-quadratically regular,* P *be a* ζ*-spectral approximation, and recall* $\gamma_u^{\max} := \max_{i\in[n]}\gamma_{u_i}$. *Instate Assumption 1 and Assumption 2. Then for any* $w', w \in \mathbb{R}^p$,

$$\mathbb{E}\|\widehat{\nabla}F(w) - \widehat{\nabla}F(w')\|_{P^{-1}}^2 \leq 2\mathcal{L}_P\left(F(w) - F(w') - \langle \nabla F(w'), w - w'\rangle\right),$$

where

$$\mathcal{L}_P := \left(\frac{n(b_g-1)}{b_g(n-1)}\gamma_u + \tau_\star^\nu\frac{n-b_g}{b_g(n-1)}\gamma_u^{\max}\right)(1+\zeta).$$

The proof of Proposition 16 in Appendix B.3.

The preconditioned expected smoothness constant $\mathcal{L}_P$ in Proposition 16 is the preconditioned analogue of the smoothness constant in the stochastic gradient setting. Indeed, if $n = b_g$, then $\mathcal{L}_P = (1+\zeta)\gamma_u$, which is the smoothness constant of F with respect to the preconditioned norm $\|\cdot\|_P$. When $\gamma_u = \mathcal{O}(1)$ we have $\mathcal{L}_P = \mathcal{O}(1)$. However, PROMISE operates in the setting $b_g \ll n$, in which case Proposition 16 yields a new phenomenon not present for full gradients $b_g = n$; namely, that even if $\gamma_u^{\max} = \mathcal{O}(1)$, it is not guaranteed that $\mathcal{L}_P = \mathcal{O}(1)$. To ensure $\mathcal{L}_P = \mathcal{O}(1)$, the gradient batchsize must satisfy $b_g = \mathcal{O}(\tau_\star^\nu)$. Thus, to realize the benefits of preconditioning, b_g must be sufficiently large.

The Hessian dissimilarity determines the required size of the b_g. The dependence on $\tau_\star^\nu$ reflects the requirement that b_g must be large enough to ensure that Hessian of the minibatch objective (i.e., the first derivative of the minibatch gradient) is a good approximation to the Hessian of the full objective. If b_g is too small, the corresponding minibatch Hessian may have curvature that is quite different from the full Hessian. In this case, we should not expect a preconditioner built from a good approximation of the full Hessian to help, as it contains information unrelated to that of the minibatch stochastic gradient. Moreover, we have the natural conclusion that the required gradient batchsize is smaller when the $\nabla^2 F_i$ are more similar, and larger when they are more dissimilar.

Overall, Proposition 16 shows that preconditioning is not a panacea. For problems with highly non-uniform data, convergence of PROMISE methods may be slow if the gradient batchsize is smaller than $\mathcal{O}(\tau_\star^\nu)$. We emphasize this limitation is independent of any particular preconditioning technique, and would remain true even if PROMISE used the perfect preconditioner—the Hessian itself. The problem stems from the use of uniform sampling to construct the stochastic gradient. Evidently, given our extensive empirical results in Section 6, problem instances requiring large gradient batches seem to be uncommon. This is unsurprising, as the data in many ML problems is (approximately) i.i.d.; hence we have strong reasons to believe that data is relatively uniform due to statistical similarity.

To our knowledge, the analysis above is the first to demonstrate the necessity of a minimum gradient batchsize to see the benefits of preconditioning. Previously, Dereziński (2022) observed that least squares with highly coherent data matrices requires large gradient batchsizes, and demonstrated the phenomena empirically. Moreover, Dereziński (2022) shows that leverage score sampling to select the gradients can reduce the required gradient batchsize. Our analysis generalizes this observation to arbitrary loss functions, and provides a simple explanation based on the expected preconditioned smoothness constant.

5.5 SketchySVRG

Theorem 17 shows the global linear convergence of SketchySVRG.

Theorem 17 (SketchySVRG convergence) *Instate the hypotheses of Assumption 1-Assumption 2. Run SketchySVRG with fixed learning rate* $\eta = \frac{1}{8\mathcal{L}_P}$ *and* $m = \frac{19}{(1-\zeta)}\frac{\mathcal{L}_P}{\gamma_\ell}$ *inner iterations. Then*

$$\mathbb{E}[F(\hat{w}^{(s)})] - F(w_\star) \leq \epsilon$$

after $s = 10\log(1/\epsilon)$ outer iterations. Hence, the total number of stochastic gradient queries required to reach an ϵ-suboptimal point is bounded by

$$10\left(n + 19\frac{1+\zeta}{1-\zeta}\left(n\frac{b_g-1}{n-1}\mathfrak{q} + \tau_\star^\nu\frac{n-b_g}{n-1}\mathfrak{q}_{\max}\right)\right)\log\left(\frac{1}{\epsilon}\right).$$

The proof of Theorem 17 is given in Section 5.7.

Theorem 17 shows that SketchySVRG converges linearly at rate controlled by the quadratic regularity ratio $\mathfrak{q}$ and the maximum quadratic regularity ratio $\mathfrak{q}_{\max}$. We have seen these quantities are well-behaved locally for structured functions (Section 5.4). In particular, for functions with a Lipschitz continuous Hessian and qsc-functions, the quadratic regularity ratio is bounded by a constant in any appropriately sized neighborhood of the optimum. Unfortunately, it is difficult to show that the quadratic regularity ratio is small over the entire domain. Hence we cannot conclude in general that SketchySVRG is faster than SVRG. This is unsurprising: existing lower bounds for Newton's method and its approximate variants are no better than those for accelerated gradient descent (Arjevani and Shamir, 2017; Arjevani et al., 2019).

Nevertheless, our experiments in Section 6 show that SketchySVRG converges faster than SVRG to the optimum. Moreover, we verify that the quadratic regularity constants and quadratic regularity ratio along the trajectory are much smaller than what worst-case bounds would predict, which helps explain the improved performance of SketchySVRG over SVRG.

Corollary 18 (SketchySVRG: Fast ridge regression) *Instate the hypotheses of Theorem 17, and suppose F is quadratic. Run SketchySVRG with gradient batchsize $b_g \geq 1$ and $m = 19\frac{1+\zeta}{1-\zeta}\frac{n}{b_g}$ inner iterations. Then*

$$\mathbb{E}[F(\hat{w}^{(s)})] - F(w_\star) \leq \epsilon$$

where $\hat{w}^{(s)}$ is the output after running $s = 10\log\left(\frac{1}{\epsilon}\right)$ outer iterations. Hence the total number of stochastic gradient evaluations to reach an ϵ-suboptimal point is bounded by

$$10\left(1 + 19\frac{1+\zeta}{1-\zeta}\right)n\log\left(\frac{1}{\epsilon}\right).$$

5.6 SketchySVRG: Fast Local Convergence

We establish local linear convergence of SketchySVRG independent of the condition number in the neighborhood

$$\mathcal{N}_{\varepsilon_0}(w_\star) = \left\{w \in \mathbb{R}^p : \|w - w_\star\|_{\nabla^2 F(w_\star)} \leq \frac{\varepsilon_0\nu^{3/2}}{2M}\right\},$$

where M is the uniform Lipschitz constant for each $\nabla^2 F_i$. This result is analagous to the fast local convergence of Newton's method in the full gradient setting.

Theorem 19 *Let $\varepsilon_0 \in (0, 1/6]$. Suppose that each F_i has an M-Lipschitz Hessian, and that $w_0 \in \mathcal{N}_{\varepsilon_0}(w_\star)$. Instate Assumption 1 and Assumption 2 with $\zeta = \varepsilon_0$. Run Algorithm 3 using Option I with $\mathcal{U} = \{0\}$, $m = 6$ inner iterations, $s = 2\log(1/\epsilon)$ outer iterations, $\eta = 1$, and $b_g = \widetilde{\mathcal{O}}\left(\tau(\mathcal{N}_{\varepsilon_0}(w_\star))\log(\frac{1}{\delta})\right)$. Then with probability at least $1-\delta$,*

$$F(\hat{w}^{(s)}) - F(w_\star) \leq \epsilon.$$

Hence the total number of stochastic gradient queries required to reach an ϵ-suboptimal point is bounded by

$$3\left[n + 6\widetilde{\mathcal{O}}\left(\tau(\mathcal{N}_{\varepsilon_0}(w_\star))\log\left(\frac{1}{\delta}\right)\right)\right]\log\left(\frac{1}{\epsilon}\right).$$

The proof is given in Appendix B.4.

Theorem 19 shows that once the iterates are close enough to the optimum, SketchySVRG converges linearly at a rate independent of the condition number, provided the gradient batchsize satisfies $b_g = \widetilde{\mathcal{O}}(\tau_\star^\nu(\mathcal{N}_{\varepsilon_0}(w_\star)))$. Recall $\tau_\star^\nu(\mathcal{N}_{\varepsilon_0}(w_\star))$ is always smaller than n, and is significantly smaller when there are no outliers amongst the individual Hessians $\nabla^2 F_i$.

Theorem 19 significantly improves the required gradient batchsize relative to the prior state-of-the-art. Previously, the best-known batch size requirment was due to Dereziński (2022), which proved a result similar to Theorem 19 when P is the SSN preconditioner, but required $b_g = \widetilde{\mathcal{O}}(\kappa_{\max})$. Although this bound on b_g improves upon the results of Roosta-Khorasani and Mahoney (2019), it can easily exceed n for ill-conditioned problems. In contrast, Theorem 19 reduces the required gradient batchsize from $\widetilde{\mathcal{O}}(\kappa_{\max})$ to $\widetilde{\mathcal{O}}(\tau_\star^\nu(\mathcal{N}_{\varepsilon_0}(w_\star)))$. In the ill-conditioned setting, this reduction can be dramatic. As a concrete example, Corollary 21 shows that for GLMs (with some mild hypotheses), the required gradient batchsize is as small as $\widetilde{\mathcal{O}}(\sqrt{n})$, while prior bounds would indicate a required batchsize of $\mathcal{O}(n)$. Hence, Theorem 19 supports modest gradient batchsizes, which agrees with practice, as PROMISE methods provide excellent empirical performance without large gradient batch sizes. The key idea for achieving the improvements in Theorem 19 is quadratic regularity, which enables tighter control over the gradient in the inverse Hessian norm with high probability.

Remark 20 *In the worst case, $\tau_\star^\nu(\mathcal{N}_{\varepsilon_0}(w_\star)) = n$. Thus, Theorem 19 would require $b_g = \widetilde{\mathcal{O}}(n)$ to achieve fast local convergence. By shrinking the size of $\mathcal{N}_{\varepsilon_0}(w_\star)$, we can still achieve fast local convergence with $b_g \ll n$. The downside of shrinking $\mathcal{N}_{\varepsilon_0}(w_\star)$ is that SketchySVRG will take longer to converge.*

To better understand the implications of Theorem 19, we present the following corollary, which addresses the setting where F is a GLM.

Corollary 21 *Instate the hypotheses of Theorem 19 and let F be a bounded GLM. Moreover suppose its data matrix A has polynomially decaying singular values, the regularization satisfies $\nu = \mathcal{O}(1/n)$, and the ridge leverage incoherence satisfies $\chi_\star^\nu(\mathcal{N}_{\varepsilon_0}(w_\star)) = \mathcal{O}(1)$*[*]*. Run Algorithm 3 with $b_g = \widetilde{\mathcal{O}}\left(\sqrt{n}\log(\frac{1}{\delta})\right)$. Then with probability at least $1-\delta$, at most*

$$3\left[n + 6\widetilde{\mathcal{O}}\left(\sqrt{n}\log\left(\frac{1}{\delta}\right)\right)\right]\log\left(\frac{1}{\epsilon}\right)$$

[*]. Equivalently, $\Phi''(Aw_\star)^{1/2}A$ is ridge leverage incoherent.

stochastic gradient queries are required to find an ϵ-suboptimal point.

Corollary 21 shows that under mild hypotheses on A and the Hessian, SketchySVRG achieves fast local convergence with a small gradient batchsize of $\widetilde{\mathcal{O}}(\sqrt{n})$, for common values of the regularization parameter ν. Prior results such as Roosta-Khorasani and Mahoney (2019); Dereziński (2022) would require $b_g = \mathcal{O}(n)$, which indicates full gradients must be used. Thus, the tighter analysis provided here yields a real improvement over prior work, as it shows that large gradient batch sizes are unnecessary to see the benefits of preconditioning for ill-conditioned GLMs.

5.7 Convergence Proof of SketchySVRG

In this section we prove Theorem 17, which establishes linear convergence of SketchySVRG. The proof is divided into a sequence of helper lemmas; taken together these lemmas allow us to easily establish the theorem.

5.7.1 Notation

For clarity in the proof, we explicitly keep track of the outer iteration that a quantity belongs to. Specifically, we write $w_k^{(s)}$ for the kth iterate in outer iteration s, and do the same for other quantities. Under this convention, $v_k^{(s)}$ denotes the variance-reduced gradient at the kth iteration of outer iteration s, and $P_k^{(s)}$ is the current preconditioner at the kth iteration of outer iteration s.

5.7.2 Preliminary Lemmas

Here, we establish helper lemmas needed to prove Theorem 17. We start by bounding the second moment of the preconditioned variance-reduced stochastic gradients.

Lemma 22 (Variance bound) *Let $v_k^{(s)} = \widehat{\nabla} F(w_k^{(s)}) - \widehat{\nabla} F(\hat{w}^{(s)}) + \nabla F(\hat{w}^{(s)})$ be the variance-reduced stochastic gradient at inner iteration k in outer iteration s. Then*

$$\mathbb{E}\|v_k^{(s)}\|^2_{(P_k^{(s)})^{-1}} \leq 4\mathcal{L}_P[F(w_k^{(s)}) - F(w_\star) + F(\hat{w}^{(s)}) - F(w_\star)].$$

Proof We have

$$\begin{aligned}
\mathbb{E}\|v_k^{(s)}\|^2_{(P_k^{(s)})^{-1}} &\overset{(1)}{\leq} 2\mathbb{E}\|\widehat{\nabla} F(w_k^{(s)}) - \widehat{\nabla} F(w_\star)\|^2_{(P_k^{(s)})^{-1}} + 2\mathbb{E}\|[\widehat{\nabla} F(\hat{w}^{(s)}) - \widehat{\nabla} F(w_\star)] - \nabla F(\hat{w}^{(s)})\|^2_{(P_k^{(s)})^{-1}} \\
&= 2\mathbb{E}\|\widehat{\nabla} F(w_k^{(s)}) - \widehat{\nabla} F(w_\star)\|^2_{(P_k^{(s)})^{-1}} \\
&\quad + 2\mathbb{E}\|[\widehat{\nabla} F(\hat{w}^{(s)}) - \widehat{\nabla} F(w_\star)] - \mathbb{E}[\widehat{\nabla} F(\hat{w}^{(s)}) - \widehat{\nabla} F(w_\star)]\|^2_{(P_k^{(s)})^{-1}} \\
&\overset{(2)}{\leq} 2\mathbb{E}\|\widehat{\nabla} F(w_k^{(s)}) - \widehat{\nabla} F(w_\star)\|^2_{(P_k^{(s)})^{-1}} + 2\mathbb{E}\|\widehat{\nabla} F(\hat{w}^{(s)}) - \widehat{\nabla} F(w_\star)\|^2_{(P_k^{(s)})^{-1}} \\
&\overset{(3)}{\leq} 4\mathcal{L}_P[F(w_k^{(s)}) - F(w_\star) + F(\hat{w}^{(s)}) - F(w_\star)].
\end{aligned}$$

Here, (1) uses $\|a+b\|_A^2 \leq 2\left(\|a\|_A^2 + \|b\|_A^2\right)$ and (2) uses $\mathbb{E}\|X - \mathbb{E}X\|_A^2 \leq \mathbb{E}\|X\|_A^2$, which are valid for any random variable X and symmetric positive definite matrix A. Finally, (3) applies Proposition 16 with $w' = w_\star$ twice. ■

Next, we have the following one-step relation.

Lemma 23 (One-step bound) *Suppose we are in outer iteration s at inner iteration k and $w_{k+1}^{(s)} = w_k^{(s)} - \eta(P_k^{(s)})^{-1}v_k^{(s)}$. Then*

$$\mathbb{E}_k\|w_{k+1}^{(s)}-w_\star\|_{P_k^{(s)}}^2 \leq \|w_k^{(s)}-w_\star\|_{P_k^{(s)}}^2+2\eta\left(2\eta\mathcal{L}_P-1\right)[F(w_k^{(s)})-F(w_\star)]+4\eta^2\mathcal{L}_P[F(\hat{w}^{(s-1)})-F(w_\star)].$$

Proof Simply use the definition of the update, expand the square, and invoke Lemma 22. ■

We now come to the key lemma for establishing convergence, which shows a contraction of suboptimalities between consecutive outer iterations.

Lemma 24 (outer iteration contraction) *Suppose we are in outer iteration $s+1$. Then*

$$\mathbb{E}_{0:s}[F(\hat{w}^{(s+1)})] - F(w_\star) \leq \left[\frac{1}{(1-\zeta)\gamma_\ell\eta(1-2\eta\mathcal{L}_P)m} + \frac{2\eta\mathcal{L}_P}{1-2\eta\mathcal{L}_P}\right]\left(F(\hat{w}^{(s)}) - F(w_\star)\right), \tag{11}$$

where $\mathbb{E}_{0:s}$ denotes the expectation conditioned on outer iterations 0 through s.

Proof Summing the bound in Lemma 23 over $k = 0, \ldots m-1$, we reach

$$\sum_{k=0}^{m-1}\mathbb{E}_k\|w_{k+1}^{(s)} - w_\star\|_{P_k^{(s)}}^2 \leq \sum_{k=0}^{m-1}\|w_k^{(s)} - w_\star\|_{P_k^{(s)}}^2 + 2\eta m\left(2\eta\mathcal{L}_P - 1\right)\frac{1}{m}\sum_{k=0}^{m-1}[F(w_k^{(s)}) - F(w_\star)]$$
$$+ 4m\eta^2\mathcal{L}_P[F(\hat{w}^{(s)}) - F(w_\star)].$$

Now, taking the expectation over all the inner iterations conditioned on outer iterations 0 through s, we find

$$\mathbb{E}_{0:s}\|w_m^{(s)} - w_\star\|_{P_k^{(s)}}^2 \leq \|\hat{w}^{(s)} - w_\star\|_{P_0^{(s)}}^2 + 2\eta m\left(2\eta\mathcal{L}_P - 1\right)\left(\mathbb{E}_{0:s}\left[F(\hat{w}^{(s+1)})\right] - F(w_\star)\right)$$
$$+ 4m\eta^2\mathcal{L}_P[F(\hat{w}^{(s)}) - F(w_\star)].$$

Rearranging and invoking quadratic regularity of f, we reach

$$\mathbb{E}_{0:s}\|w_m^{(s)} - w_\star\|_{P_k^{(s)}}^2 + 2\eta m\left(1 - 2\eta\mathcal{L}_P\right)\left(\mathbb{E}_{0:s}\left[F(\hat{w}^{(s+1)})\right] - F(w_\star)\right)$$
$$\leq 2\left(\frac{1}{(1-\zeta)\gamma_\ell} + 2m\eta^2\mathcal{L}_P\right)[F(\hat{w}^{(s)}) - F(w_\star)].$$

Hence we conclude

$$\mathbb{E}_{0:s}[F(\hat{w}^{(s+1)})] - F(w_\star) \leq \left[\frac{1}{(1-\zeta)\gamma_\ell\eta(1-2\eta\mathcal{L}_P)m} + \frac{2\eta\mathcal{L}_P}{1-2\eta\mathcal{L}_P}\right]\left(F(\hat{w}^{(s)}) - F(w_\star)\right).$$

■

5.7.3 SketchySVRG Convergence: Proof of Theorem 17

Proof From Lemma 24 we have,

$$\mathbb{E}_{0:s-1}[F(\hat{w}^{(s)})] - F(w_\star) \leq \left[\frac{1}{(1-\zeta)\gamma_\ell \eta(1-2\eta\mathcal{L}_P)m} + \frac{2\eta\mathcal{L}_P}{1-2\eta\mathcal{L}_P}\right]\left(F(\hat{w}^{(s-1)}) - F(w_\star)\right).$$

Setting $\eta = \frac{1}{8\mathcal{L}_P}$ and $m = \frac{19}{1-\zeta}\bar{\mathfrak{q}}$, we obtain

$$\mathbb{E}_{0:s-1}[F(\hat{w}^{(s)})] - F(w_\star) \leq \frac{9}{10}\left(F(\hat{w}^{(s-1)}) - F(w_\star)\right).$$

Taking the total expectation over all outer iterations, and recursing, we reach

$$\mathbb{E}[F(\hat{w}^{(s)})] - F(w_\star) \leq \left(\frac{9}{10}\right)^s (F(w_0) - F(w_\star)).$$

Hence after $s = 10\log\left(\frac{F(w_0)-F(w_\star)}{\varepsilon}\right)$ outer iterations we have

$$\mathbb{E}[F(\hat{w}^{(s)})] - F(w_\star) \leq \varepsilon.$$

■

6. Numerical Experiments

In this section, we provide four sets of experiments to demonstrate the effectiveness of the PROMISE methods for l^2-regularized least squares and logistic regression problems. We also investigate the quadratic regularity ratio. We present the following results:

- Performance Experiments (Section 6.1): We compare PROMISE methods to SVRG, b-nice SAGA (henceforth referred to as SAGA), Loopless Katyusha (L-Katyusha), and stochastic L-BFGS (SLBFGS), whose learning rates are tuned. We find that our methods outperform the competition on a testbed of 51 medium-sized least squares and logistic regression problems.
- Suboptimality experiments (Section 6.2): We show that PROMISE methods achieve global linear convergence on several least squares and logistic regression problems, which matches the global linear convergence guarantees in Section 5. Furthermore, our methods converge faster than the competition.
- Showcase experiments (Section 6.3): We evaluate PROMISE methods against the competition on the url, yelp, and acsincome data sets, which originate in real-world applications and lead to large-scale problems. We again find that our methods outperform the competition.
- Streaming experiments (Section 6.4): We test PROMISE methods on performing logistic regression with a large-scale transformation of the HIGGS data set. This transformed data set is so large that it does not fit in the memory of most computers, putting these experiments in a streaming setting where the computation of full gradients is prohibitive. Our methods continue to outperform the competition.

- Regularity study (Section 6.5): We demonstrate that the quadratic regularity ratio, γ_u/γ_ℓ, is well-behaved over the optimization trajectory, which provides empirical support for our claims in Section 6.5.

The experiments in Sections 6.1 to 6.5 run PROMISE methods with the default hyperparameters given in Section 3. Throughout the experiments, we set the l^2-regularization parameter $\nu = 10^{-2}/n_{\mathrm{tr}}$, where n_{tr} is the number of samples in the training set, which typically results in an ill-conditioned problem. All preconditioners use the default values of r and ρ in Section 2.2. Additional details appear in Appendix D of https://arxiv.org/abs/2309.02014v2 and code for our experiments can be found at https://github.com/udellgroup/PROMISE.

6.1 Performance Experiments

Our first set of experiments compares the performance of SketchySVRG, SketchySAGA, and SketchyKatyusha, with their *default* hyperparameters, to SVRG, SAGA, L-Katyusha, and SLBFGS, with *tuned* hyperparameters, on solving ridge and l^2-regularized logistic regression problems. These experiments therefore *understate* the performance improvement that can be expected by using PROMISE methods. Moreover, we modify SLBFGS to compute the preconditioner once per epoch rather than at every iteration for a fair comparison.

SAGA/SketchySAGA require one full pass through the data per epoch, while SVRG/L-Katyusha/SLBFGS/SketchySVRG/SketchyKatyusha use two full passes through the data per epoch since they compute full gradients[*]. By using the number of full data passes we (roughly) equate the computation required for computing gradients, making for a fair comparison. We compute the minimum $F(w^\star)$ for all ridge and logistic regression problems via `scikit-learn` (Pedregosa et al., 2011). We run neither SketchySGD nor SGD because these algorithms do not converge linearly.

Our primary metrics for comparing the performance of these methods are the wall-clock time and number of full data passes to reach suboptimality within 10^{-4} of the minimum, $F(w^\star)$. Each optimizer is run either until this suboptimality condition is met (i.e., the problem is solved), or for 200 full data passes (100 epochs for SVRG, L-Katyusha, SLBFGS, SketchySVRG, and SketchyKatyusha, 200 epochs for SAGA and SketchySAGA).

6.1.1 Ridge Regression

We solve ridge regression problems of the form

$$\mathrm{minimize}_{w\in\mathbb{R}^p} \ \frac{1}{n_{\mathrm{tr}}}\sum_{i=1}^{n_{\mathrm{tr}}}\frac{1}{2}(a_i^T w - b_i)^2 + \frac{\nu}{2}\|w\|_2^2,$$

where $a_i \in \mathbb{R}^p$ is a datapoint, $b_i \in \mathbb{R}$ is a label, and $\nu > 0$ is the regularization parameter.

Our experiments in this setting are performed on a testbed of 17 data sets from OpenML (Vanschoren et al., 2013) and LIBSVM (Chang and Lin, 2011). We apply random features (Rahimi and Recht, 2007; Mei and Montanari, 2022) to most, but not all, data sets; further details regarding preprocessing may be found in the arxiv report.

[*]. L-Katyusha and SketchyKatyusha compute full gradients with random probability, and our hyperparameter settings result in one full gradient computation per epoch, in expectation.

Results appear in Fig. 2, which shows the proportion of problems solved by both our methods and the competitor methods as a function of wall-clock time and full data passes. When combined with any of the SSN, NySSN, SASSN-C, and SASSN-R preconditioners, SketchySVRG, SketchySAGA, and SketchyKatyusha uniformly outperform competitor methods. SketchyKatyusha and SketchySVRG perform slightly better than SketchySAGA, supporting our recommendation to use SketchyKatyusha for ridge regression.

6.1.2 l^2-regularized Logistic Regression

We solve l^2-regularized logistic regression problems of the form

$$\text{minimize}_{w\in\mathbb{R}^p}\ \frac{1}{n_{\text{tr}}}\sum_{i=1}^{n_{\text{tr}}}\log(1+\exp(-b_i a_i^T w))+\frac{\nu}{2}\|w\|_2^2,$$

where $a_i \in \mathbb{R}^p$ is a datapoint, $b_i \in \{-1, 1\}$ a label, and $\nu > 0$ the regularization parameter.

These experiments use a testbed of 34 data sets from LIBSVM. We apply random features to a few of the data sets; further details regarding preprocessing appear in the arxiv report.

The results of these experiments appear in Fig. 3, which shows the proportion of problems solved by both our methods and the competitor methods as a function of wall-clock time and full data passes. When combined with any one of the SSN, NySSN, SASSN-C, and SASSN-R preconditioners, SketchySVRG, SketchySAGA, and SketchyKatyusha uniformly outperform SVRG, SAGA and L-Katyusha. In addition, SketchySAGA and SketchyKatyusha outperform SLBFGS, which also employs preconditioning.

Overall, SketchySAGA and SketchyKatyusha perform much better than SketchySVRG here, supporting our recommendation in Section 3.5 to use SketchyKatyusha (assuming we can compute full gradients) or SketchySAGA for logistic regression.

6.2 Suboptimality Experiments

We examine the objective suboptimality (with respect to the lowest attained training loss for all methods) for SketchySVRG, SketchySAGA, and SketchyKatyusha, with their *default* hyperparameters, and the competitor methods, with *tuned* hyperparameters. For simplicity, we only show PROMISE methods with the NySSN and SSN preconditioner. Each optimizer is run for 200 full data passes (100 epochs for SVRG, L-Katyusha, SLBFGS, SketchySVRG, and SketchyKatyusha, 200 epochs for SAGA and SketchySAGA).

Figs. 4 and 5 display objective suboptimality (with respect to the lowest attained training loss) for selected data sets on ridge and l^2-regularized logistic regression. The objective suboptimality for PROMISE methods decreases linearly for ridge and logistic regression, which matches the theoretical convergence guarantees in Section 5. On ridge regression, PROMISE methods uniformly outperform the competition, even reaching machine precision on the yolanda data set! On logistic regression, PROMISE methods generally outpeform SVRG, SAGA, and L-Katyusha. Interestingly, SLBFGS outperforms PROMISE methods on ijcnn1. However, SLBFGS can be unstable; for example, SLBFGS initially outperforms PROMISE methods on SUSY, but the training loss suddenly spikes and then diverges.

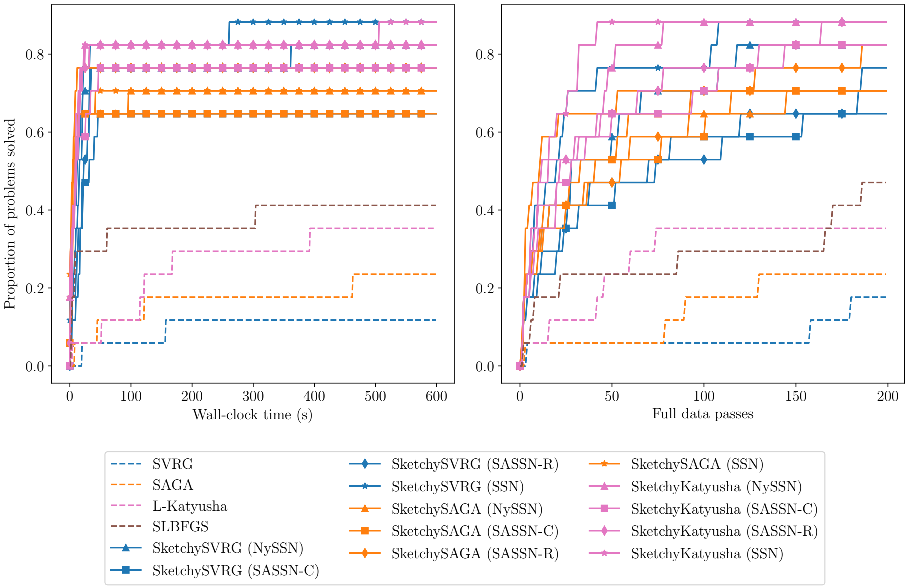

Figure 2: PROMISE methods solve ridge regression problems faster than competitors.

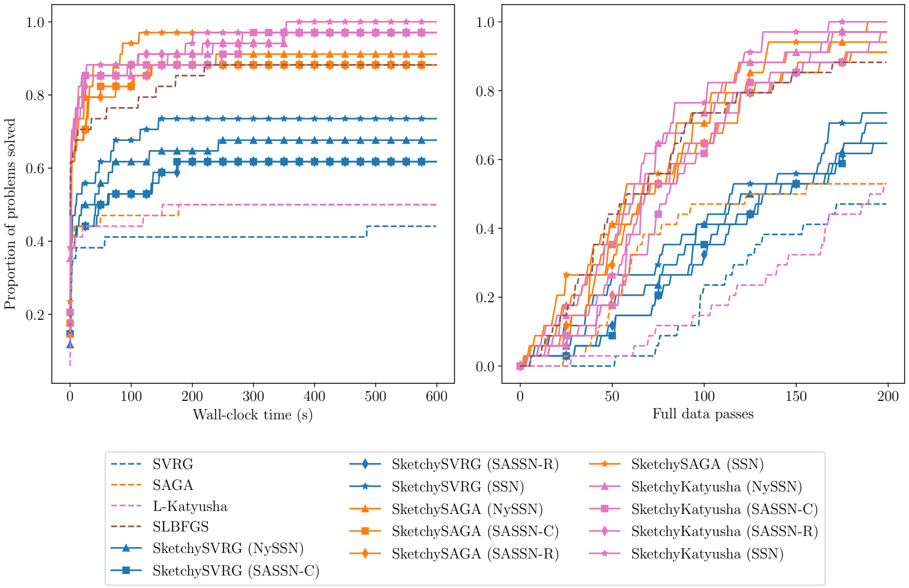

Figure 3: PROMISE methods (and SLBFGS) solve l^2-regularized logistic regression problems faster than competitors.

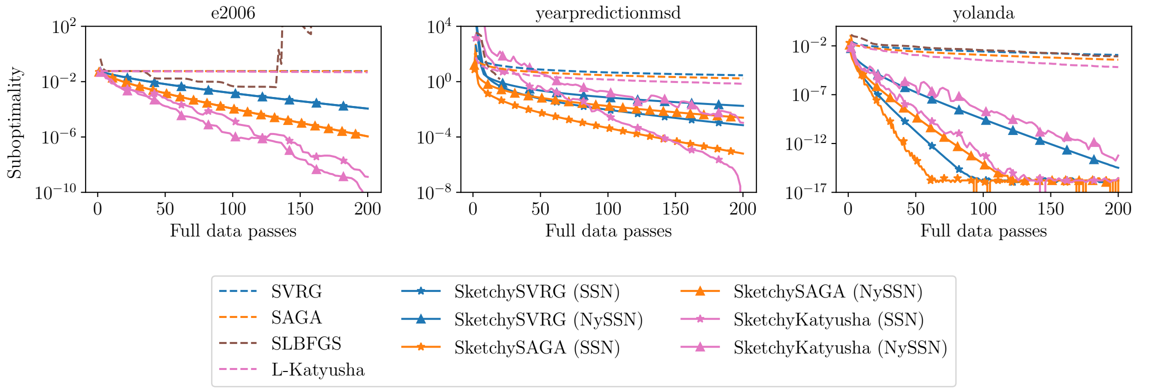

Figure 4: Suboptimality comparisons between our proposed methods and tuned competitor methods for selected data sets on ridge regression.

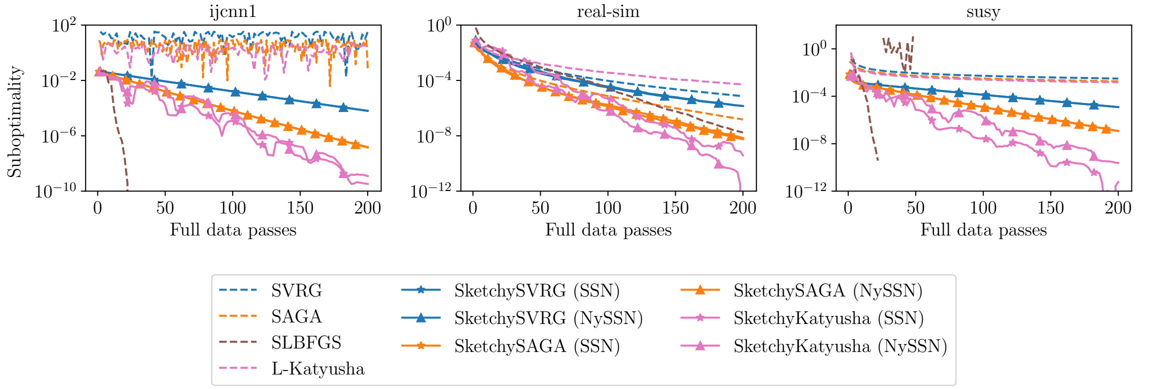

Figure 5: Suboptimality comparisons between our proposed methods and tuned competitor methods for selected data sets on l^2-regularized logistic regression.

6.3 Showcase Experiments

Our second set of experiments compares the performance of SketchySVRG, SketchySAGA, and SketchyKatyusha, with their *default* hyperparameters, to SVRG, SAGA, L-Katyusha, and SLBFGS with both default and tuned hyperparameters on the url, yelp, and acsincome data sets. All three data sets originate in real-world applications: the url data set is used to train a l^2-regularized logistic regression classifier that detects malicious websites using features derived from URLs, the yelp data set is used to train a l^2-regularized logistic regression classifier that predicts sentiment from user reviews, and the acsincome data set is used to train a ridge regression classifier that predicts income given demographic information such as age, employment, and education. After preprocessing, all three of these data sets have $n_{\text{tr}} > 10^6$ training examples, while url and yelp have $p > 10^6$ features, putting all three of these data sets in the big-data regime. We provide two sets of comparisons: the first set compares our methods to SVRG, SAGA, and L-Katyusha with their *default* hyperparameters, while the second set compares our methods to SVRG, SAGA, L-Katyusha, and SLBFGS with their *tuned* hyperparameters. We run each optimizer with a fixed time budget: 1 hour for url and yelp, and 2 hours for acsincome.

The first set of comparisons appears in Fig. 6, which compares our methods and the competitor methods (with default hyperparameters) on test classification error (url, yelp) and test loss (acsincome) as a function of wall-clock time. When combined with either of the SSN or NySSN preconditioners, SketchySVRG, SketchySAGA, and SketchyKatyusha uniformly outperform the competitor methods on their default hyperparameters. Our methods generalize better to test data than the competitor methods while running much faster.

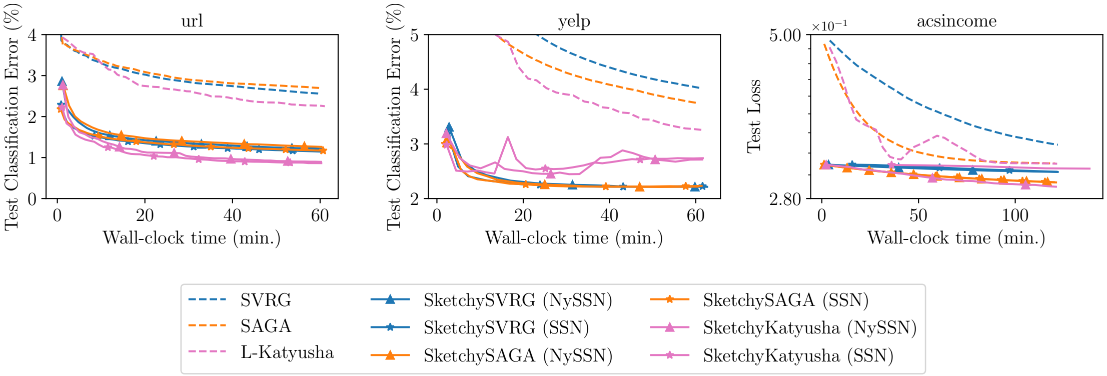

Figure 6: Comparisons to competitor methods on test metrics with default learning rates (SVRG, SAGA) and smoothness parameters (L-Katyusha).

The second set of comparisons appears in Fig. 7, which compares our methods and the competitor methods (with tuned hyperparameters) on test metrics as a function of wall-clock time. PROMISE methods outperform the competition on url and acsincome and perform comparably on yelp. Moreover, recall that the performance of the competitor methods is only possible after hyperparameter tuning, which is quite expensive for data sets of this size, whereas PROMISE methods still obtain good performance with default hyperparameters.

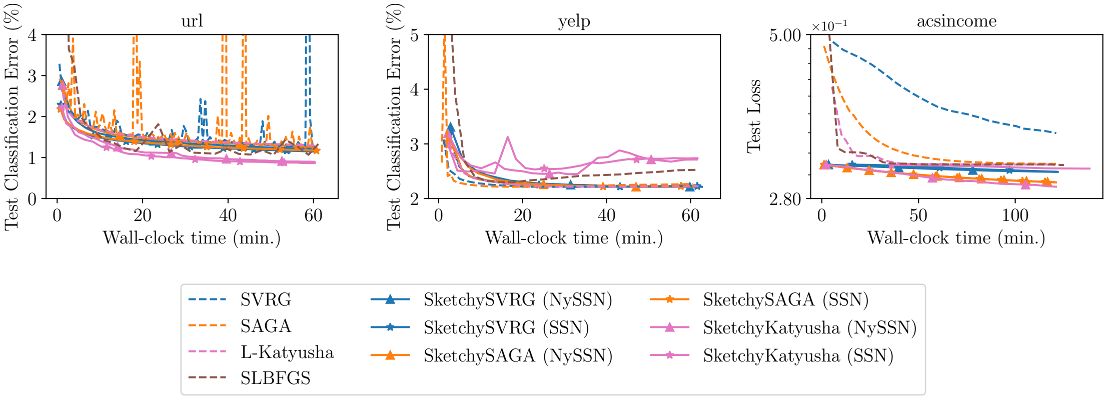

Figure 7: Comparisons to competitor methods on test metrics with tuned learning rates (SGD, SVRG, SAGA, SLBFGS) and smoothness parameters (L-Katyusha).

6.4 Streaming Experiments

We apply random features to the HIGGS data set to obtain a transformed data sets with size 840 GB (see the arxiv report for more details). This transformed data set is much larger than the hard drive and RAM of most computers. We solve a l^2-regularized logistic regression problem on this transformed data set. To perform optimization, we load the original data set in memory and at each iteration, form a minibatch of the transformed data set by applying random features to a minibatch of the data. In this setting, computing a full gradient of the objective is computationally prohibitive, so we exclude SVRG, L-Katyusha, SLBFGS, SketchySVRG, and SketchyKatyusha. We compare our methods to SGD and SAGA with their *tuned* hyperpameters. All optimization methods are run for 10 epochs.

The comparison to tuned versions of SGD and SAGA is presented in Fig. 8. On this problem, PROMISE methods (SketchySGD and SketchySAGA) perform well while the competitors (SGD and SAGA) struggle to make any progress. The NySSN preconditioner outperforms the SSN preconditioner on this large, dense problem: it achieves similar test loss at each iteration but is faster on wall-clock time. We only plot test loss, as computing the training loss suffers from the same computational issues as computing a full gradient. The plots with respect to wall-clock time only show the time taken in optimization; they do not include the time taken in repeatedly applying the random features transformation.

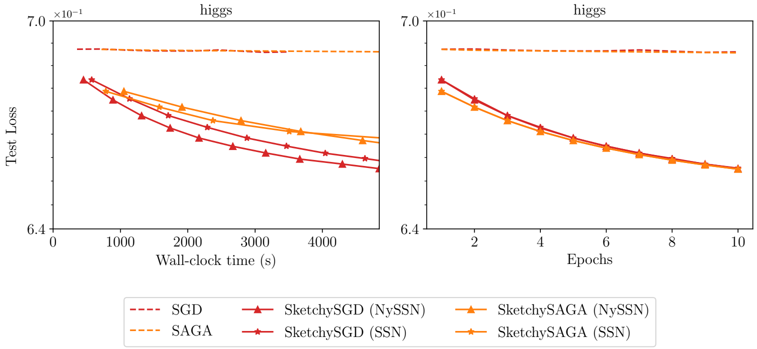

Figure 8: PROMISE methods outperform (tuned) SGD and SAGA on HIGGS.

6.5 Regularity Study: Why do PROMISE Methods Converge Fast Globally?

The theory in Section 5 shows that the PROMISE algorithms converge linearly, with the rate of convergence being controlled by the weighted quadratic regularity ratio $\bar{\mathfrak{q}}$. It is encouraging that the convergence rate is no longer controlled by the condition number, yet the quantity $\bar{\mathfrak{q}}$ is not local, and so could be quite large. Indeed, Proposition 12 implies in the worst case it may be as large as κ^2, which is at odds with the empirical results in Sections 6.1 to 6.4. Here we provide an empirical argument that partly explains why PROMISE methods exhibit faster convergence than stochastic first-order methods. The key observation in this regard is that the optimization trajectory does not arbitrarily traverse $\mathbb{R}^p$—it stays in localized regions. Thus, the speed of convergence is determined by the *local* values of $\bar{\mathfrak{q}}$, and not its global value over all of $\mathbb{R}^p$. As the value of $\bar{\mathfrak{q}}$ over a localized region may be better behaved than over the whole space, PROMISE methods can take larger step-sizes, which leads to faster convergence. Furthermore, as the iterates approach the optimum, the values of $\bar{\mathfrak{q}}$ over these localized regions approach 1. Hence we expect the local weighted quadratic regularity ratio to be small.

In this section, we provide empirical evidence for the hypothesis of the preceding paragraph by studying a local version of the quadratic regularity ratio $\mathfrak{q}$ along the optimization trajectory. To this end, we start by defining appropriate local versions of the quadratic regularity constants, which we base on the definitions of quadratic regularity in Appendix A. Close inspection of our analysis reveals that we only need quadratic regularity with w_0 equal to the iterate where we compute the preconditioner, w_1 set to be the current iterate, and w_2 set equal to the optimum. Hence appropriate definitions for the local quadratic regularity constants $\gamma_{u,j}, \gamma_{\ell,j}$ and local quadratic regularity ratio $\mathfrak{q}_j$ are given by

$$\gamma_{u,j} \coloneqq \max_{w \in S_j} \int_0^1 2(1-t) \frac{\|w_\star - w\|^2_{\nabla^2 F(w+t(w_\star - w))}}{\|w_\star - w\|^2_{\nabla^2 F(w_j)}} dt, \tag{12}$$

$$\gamma_{\ell,j} \coloneqq \min_{w \in S_j} \int_0^1 2(1-t) \frac{\|w_\star - w\|^2_{\nabla^2 F(w+t(w_\star - w))}}{\|w_\star - w\|^2_{\nabla^2 F(w_j)}} dt, \tag{13}$$

$$\mathfrak{q}_j \coloneqq \frac{\gamma_{u,j}}{\gamma_{\ell,j}}, \tag{14}$$

where w_j is the iterate where we compute the preconditioner, S_j is the set of iterates associated with the preconditioner P_j (i.e., iterates on the trajectory between w_j, inclusive and w_{j+1}, exclusive), and $w_\star$ is the optimum.

Fig. 9 shows $\mathfrak{q}_j$ for eight data sets over 50 epochs of training using SketchySAGA with the NySSN preconditioner. For all of these data sets, $\mathfrak{q}_j \approx 1$ after 20 epochs of training. Furthermore, this phenomenon occurs before SketchySAGA converges close to the optimum; Table 10 demonstrates that $\mathfrak{q}_j \approx 1$ well before the problem has been solved (within 10^{-4} of $F(w^\star)$ as in Section 6.1). For example, ijcnn1 takes 94 epochs to be solved by SketchySAGA with the NySSN preconditioner, but $\mathfrak{q}_j \approx 1$ in less than 5 epochs.

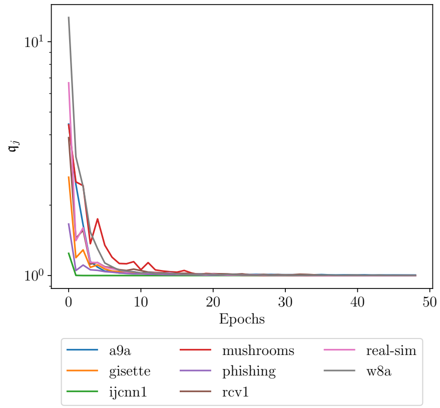

Figure 9: Plots of $\mathfrak{q}_j$ over the optimization trajectory for selected data sets.

data set	a9a	gisette	ijcnn1	mushrooms	phishing	rcv1	real-sim	w8a
# of epochs	15	86	94	20	71	49	39	28

Table 10: Number of epochs to solve logistic regression problems on selected datsets.

7. Conclusion

We introduce PROMISE, a framework for combining scalable preconditioning techniques with popular stochastic optimization methods. In particular, we present a variety of preconditioning techniques (SSN, NySSN, SASSN-C, SASSN-R, DiagSSN) and develop the preconditioned stochastic second-order methods SketchySVRG, SketchySAGA, and SketchyKatyusha. Furthermore, we provide default hyperparameters for these preconditioners and algorithms, which enable them to work out-of-the-box, even on highly ill-conditioned data.

To analyze the PROMISE methods, we introduce quadratic regularity and the quadratic regularity ratio, which generalize the notions of smoothness, strong convexity, and condition number to the Hessian norm. We also introduce Hessian dissimilarity, which allows us to give practical requirements on the gradient batchsize, a first in the literature. We show that PROMISE methods have global linear convergence, and that this convergence is condition-number free for ridge regression. Moreover, we show that SketchySVRG converges linearly at a rate independent of the condition number, once the iterates are close enough to the optimum. Hence, SketchySVRG enjoys the fast local convergence one would expect of a Newton-type method.

We empirically demonstrate the superiority of PROMISE methods over popular competitor methods for ridge and logistic regression. PROMISE methods, with their default hyperparameters, consistently outperform the competition, even when they have been tuned to achieve their best performance.

Acknowledgments

We would like to thank Catherine Chen, John Duchi, Daniel LeJeune, Michael Mahoney, Mert Pilanci, Aaron Sidford, and Ali Teshnizi for helpful discussions regarding this work. Furthermore, we would like to thank the action editor Peter Richtarik and the four anonymous reviewers for their feedback, which improved this manuscript. The authors gratefully acknowledge support from the the National Science Foundation (NSF) Award IIS-2233762, the Office of Naval Research (ONR) Award N000142212825 and N000142312203, and the Alfred P. Sloan Foundation. Finally, we would like to thank GPT-4 for inspiring the title of this work.

Appendix A. Another Definition of Quadratic Regularity

Our presentation of the upper (lower) quadratic regularity constant γ_u (γ_ℓ) in Definition 11 is based on a quadratic upper (lower) bound on F. Here, we present an equivalent definition for γ_u and γ_ℓ.

The *upper quadratic regularity constant* is defined by

$$\gamma_u(\mathcal{C}) \coloneqq \sup_{w_0 \in \mathcal{C}} \left(\sup_{w_1, w_2 \in \mathcal{C}, w_1 \neq w_2} \int_0^1 2(1-t) \frac{\|w_2 - w_1\|^2_{\nabla^2 F(w_1 + t(w_2 - w_1))}}{\|w_2 - w_1\|^2_{\nabla^2 F(w_0)}} dt \right). \tag{15}$$

Similarly, the *lower quadratic regularity constant* is defined by

$$\gamma_\ell(\mathcal{C}) \coloneqq \inf_{w_0 \in \mathcal{C}} \left(\inf_{w_1, w_2 \in \mathcal{C}, w_1 \neq w_2} \int_0^1 2(1-t) \frac{\|w_2 - w_1\|^2_{\nabla^2 F(w_1 + t(w_2 - w_1))}}{\|w_2 - w_1\|^2_{\nabla^2 F(w_0)}} dt \right). \tag{16}$$

Appendix B. Proofs of Main Results

B.1 Proof of Lemma 7

Proof The proof is by direct calculation. Indeed, by definition

$$d^\nu_{\text{eff}}\left(\frac{1}{n}A^T \Phi''(Aw) A\right) = d^\nu_{\text{eff}}\left(\Phi''(Aw)^{1/2} A\right) = \sum_{j=1}^n \frac{\frac{1}{n}\sigma_j^2(\Phi''(Aw)^{1/2}A)}{\frac{1}{n}\sigma_j^2(\Phi''(Aw)^{1/2}A) + \nu} \overset{(1)}{\leq} \sum_{j=1}^n \frac{CBj^{-2\beta}}{CBj^{-2\beta} + \nu}$$

$$\overset{(2)}{=} \sum_{j=1}^n \frac{CB}{CB + \nu j^{2\beta}} \leq \int_0^\infty \frac{CB}{CB + \nu x^{2\beta}} dx \overset{(3)}{=} CB\nu^{-1/(2\beta)} \int_0^\infty \frac{1}{CB + u^{2\beta}} du$$

$$\overset{(4)}{=} CB\nu^{-1/(2\beta)} \times (CB)^{\frac{1}{2\beta} - 1} \frac{\pi/(2\beta)}{\sin(\pi/(2\beta))} = \frac{\pi/(2\beta)}{\sin(\pi/(2\beta))} \left(\frac{CB}{\nu}\right)^{1/2\beta}.$$

Here (1) uses $A^T \Phi''(Aw) A \preceq BA^T A$, our hypotheses on that $\frac{1}{n}\lambda_j\left(A^T A\right) \leq Cj^{-2\beta}$, and that $\frac{x}{x+\nu}$ is increasing in x for $x \geq 0$, (2) multiplies the numerator and denominator by $j^{2\beta}$, (3) uses the substitution $u = \nu^{1/2\beta}$, and (4) uses the fact that $\int_0^\infty \frac{1}{CB + u^{2\beta}} du = (CB)^{\frac{1}{2\beta} - 1} \frac{\pi/(2\beta)}{\sin(\pi/(2\beta))}$ (Sutherland, 2017). The second claim follows from the first by plugging in $\nu = \mathcal{O}(1/n)$. ■

B.2 Quadratic Regularity for M-QSC Functions on Bounded Domains

In this subsection we prove the bound on $\mathfrak{q}(\mathcal{C})$ for M-qsc functions on a bounded domain presented in Section 5.4.

Let $w_0, w_1, w_2 \in \mathcal{C}$. Consider the quantity:

$$\int_0^1 2(1-t) \frac{\|w_2 - w_1\|^2_{\nabla^2 F(w_1 + t(w_2 - w_1))}}{\|w_2 - w_1\|_{\nabla^2 F(w_0)}} dt.$$

As F is M-qsc, it holds that (see for instance Lemma 2.5 in Doikov (2023))

$$\nabla^2 F(w_0) \mathrm{e}^{-M\|w_1 - w_0 + t(w_2 - w_1)\|} \preceq \nabla^2 F(w_1 + t(w_2 - w_1)) \preceq \nabla^2 F(w_0) \mathrm{e}^{M\|w_1 - w_0 + t(w_2 - w_1)\|}.$$

Observing the relation

$$\|w_1 - w_0 + t(w_2 - w_1)\| = \|(1-t)(w_1 - w_0) + t(w_2 - w_0)\| \leq D,$$

we deduce:

$$\exp(-MD) \leq \int_0^1 2(1-t)\frac{\|w_2 - w_1\|^2_{\nabla^2 F(w_1 + t(w_2 - w_1))}}{\|w_2 - w_1\|_{\nabla^2 F(w_0)}} dt \leq \exp(MD).$$

Thus, as w_0, w_1, w_2 are arbitrary, the preceding display combined with the definitions in Appendix A, yield:

$$\mathfrak{q}(\mathcal{C}) \leq \exp(2MD).$$

Hence if $D \leq \log(R)/M$, we obtain:

$$\mathfrak{q}(\mathcal{C}) \leq \exp(2\log(R)) \leq R^2 \leq 8,$$

whenever $R \leq \mathrm{e}$.

B.3 Proof of Proposition 16

We begin by recalling the following fundamental result from Gower et al. (2019b).

Theorem 25 (Theorem 3.6 and Proposition 3.8, Gower et al. (2019b)) *Suppose $F = \frac{1}{n}\sum_{i=1}^{n} F_i(w)$, where $F_i : \mathbb{R}^p \mapsto \mathbb{R}$. Let the following conditions hold:*

1. *F_i is convex, for every $i \in [n]$.*
2. *For each $i \in [n]$, there exists a matrix $M_i \in \mathbb{S}_p^{++}(\mathbb{R})$, such that for all $x, h \in \mathbb{R}^p$*

$$F_i(w + h) \leq F_i(w) + \langle \nabla F_i(w), h\rangle + \frac{1}{2}\|h\|^2_{M_i}.$$

3. *There exists a matrix $M \in \mathbb{S}_p^{++}(\mathbb{R})$, such that for all $x, h \in \mathbb{R}^p$*

$$F(w + h) \leq F(w) + \langle \nabla F(w), h\rangle + \frac{1}{2}\|h\|^2_{M}.$$

Then for any $w, w' \in \mathbb{R}^p$, it holds that

$$\mathbb{E}\|\widehat{\nabla} F(w) - \widehat{\nabla} F(w')\|^2 \leq 2\mathcal{L}\left(F(w) - F(w') - \langle \nabla F(w'), w - w'\rangle\right),$$

where

$$\mathcal{L} = \frac{n(b_g - 1)}{b_g(n-1)}\lambda_1(M) + \frac{n - b_g}{b_g(n-1)}\max_{i \in [n]} \lambda_1(M_i).$$

With these preliminaries out of the way, we commence the proof of Proposition 16.

Proof Observe that each F_i satisfies:

$$F_i(w + h) \leq F_i(w) + \langle \nabla F_i(w), h\rangle + \frac{1}{2}\|h\|^2_{M_i},$$

with $M_i = \gamma_{u_i}\nabla^2 F_i(w_0)$, where w_0 is the point where the preconditioner P is constructed. Hence performing the change of variable $w = P^{-1/2}z$ and defining $F_{P_i}(z) = F_i(P^{-1/2}z), F_P(z) = F(P^{-1/2}z)$, we reach

$$F_{P_i}(z+\tilde{h}) \leq F_{P_i}(z) + \langle \nabla F_{P_i}(z), \tilde{h}\rangle + \frac{\gamma_{u_i}}{2}\|\tilde{h}\|^2_{\nabla^2 F_{P_i}(z_0)},$$

$$F_P(z+\tilde{h}) \leq F_P(z) + \langle \nabla F_P(z), \tilde{h}\rangle + \frac{\gamma_u}{2}\|\tilde{h}\|^2_{\nabla^2 F_P(z_0)}.$$

Hence the conditions of Theorem 25 are satisfied with $M_i = \gamma_{u_i}\nabla^2 F_{P_i}(z_0), M = \gamma_u \nabla^2 F_P(z_0)$, and so we reach

$$\mathbb{E}\|\widehat{\nabla} F_{P_i}(z) - \widehat{\nabla} F_{P_i}(z')\|^2 \leq 2\mathcal{L}\left(F_{P_i}(z) - F_{P_i}(z') - \langle \nabla F_{P_i}(z'), z - z'\rangle\right),$$

with $\mathcal{L}$ as in Theorem 25. Thus, we obtain

$$\mathbb{E}\|\widehat{\nabla} F(w) - \widehat{\nabla} F(w')\|^2_{P^{-1}} \leq 2\mathcal{L}\left(F(w) - F(w') - \langle \nabla F(w'), w - w'\rangle\right).$$

Now,

$$\begin{aligned}
\mathcal{L} &= \frac{n(b_g-1)}{b_g(n-1)}\lambda_1(M) + \frac{n-b_g}{b_g(n-1)}\max_{i\in[n]}\lambda_1(M_i) \\
&= \frac{n(b_g-1)}{b_g(n-1)}\gamma_u\lambda_1\left(\frac{1}{n}\sum_{i=1}^{n}\nabla^2 F_{P_i}(z_0)\right) + \frac{n-b_g}{b_g(n-1)}\max_{i\in[n]}\lambda_1\left(\gamma_{u_i}\nabla^2 F_{P_i}(z_0)\right) \\
&\overset{(1)}{\leq} \frac{n(b_g-1)}{b_g(n-1)}\gamma_u(1+\zeta) + \frac{n-b_g}{b_g(n-1)}\gamma_u^{\max}\lambda_1\left(\nabla^2 F_{P_i}(z_0)\right) \\
&\overset{(2)}{\leq} \left(\frac{n(b_g-1)}{b_g(n-1)}\gamma_u + \tau_\star^\nu\frac{n-b_g}{b_g(n-1)}\gamma_u^{\max}\right)(1+\zeta) = \mathcal{L}_P,
\end{aligned}$$

where (1), (2) both use that P is a ζ-spectral approximation, and (2) uses $\nabla^2 F_i(w) \preceq \tau_\star^\nu \nabla^2 F(w)$, which follows by definition of $\tau_\star^\nu$. Hence for all $w, w' \in \mathbb{R}^p$

$$\mathbb{E}\|\widehat{\nabla} F(w) - \widehat{\nabla} F(w')\|^2_{P^{-1}} \leq 2\mathcal{L}_P\left(F(w) - F(w') - \langle \nabla F(w'), w - w'\rangle\right),$$

as desired. ■

B.4 SketchySVRG: Fast Local Convergence

In this section, we prove Theorem 19, which shows local condition number-free convergence of SketchySVRG in the neighborhood

$$\mathcal{N}_{\varepsilon_0}(w_\star) = \left\{w \in \mathbb{R}^p : \|w - w_\star\|_{\nabla^2 F(w_\star)} \leq \frac{\varepsilon_0 \nu^{3/2}}{2M}\right\}.$$

The result proven here, substantially improves upon the local convergence result of Dereziński (2022), which requires a gradient batch size of $\widetilde{\mathcal{O}}(\kappa)$ to obtain fast local convergence. In contrast, Theorem 19 only requires the gradient batchsize to satisfy $b_g = \widetilde{\mathcal{O}}(\tau_\star^\nu(\mathcal{N}_{\varepsilon_0}(w_\star))$, which is often orders of magnitude smaller than κ in the ill-conditioned setting (see Corollary 21).

The overarching idea of the proof is similar to other local analyses of stochastic Newton methods (Li et al., 2020; Derezinski et al., 2021; Dereziński, 2022). Namely, we seek to show the iterates belong to progressively smaller neighborhoods of the optimum, with a contraction rate independent of the condition number.

We start with standard notation, which will be used throughout the proof.

B.4.1 Notation

We define the following quantities:

$$\Delta_k^{(s)} := w_k^{(s)} - w_\star, \quad p_k^{(s)} := \nabla^2 F(w_k^{(s)})^{-1} v_k^{(s)}, \quad \tilde{p}_k^{(s)} := P^{-1} v_k^{(s)}.$$

$\Delta_k^{(s)}$ is the distance of the current iterate to the optimum, $p_k^{(s)}$ is the exact Newton direction, and $\tilde{p}_k^{(s)}$ is the approximate Newton direction actually computed by the algorithm.

B.4.2 Preliminary Lemmas

We begin with the following technical lemma, which shows the following items hold in $\mathcal{N}_{\varepsilon_0}(w_\star)$: (1) the quadratic regularity constants are close to unity, (2) the Hessians are uniformly close in the Loewner ordering, (3) taking an exact Newton step moves the iterate closer to the optimum in the Hessian norm, (4) $\nabla F_i(w)$, $\nabla F(w)$ are $(1+\varepsilon_0)$ Lipschitz in $\mathcal{N}_{\varepsilon_0}(w_\star)$, and (5) P^{-1} is uniformly good approximation to the inverse Hessian.

Lemma 26 *Let $w, w' \in \mathcal{N}_{\varepsilon_0}(w_\star)$, and suppose P is a ε_0-spectral approximation constructed at some $w_0 \in \mathcal{N}_{\varepsilon_0}(w_\star)$, then the following items hold.*

1.
$$\frac{1}{1+\varepsilon_0} \leq \gamma_\ell^{\min}(\mathcal{N}_{\varepsilon_0}(w_\star)) \leq \gamma_u^{\max}(\mathcal{N}_{\varepsilon_0}(w_\star)) \leq (1+\varepsilon_0).$$

2.
$$(1-\varepsilon_0)\nabla^2 F(w) \preceq \nabla^2 F(w') \preceq (1+\varepsilon_0)\nabla^2 F(w).$$

3.
$$\|w - w_\star - \nabla^2 F(w)^{-1}\nabla F(w)\|_{\nabla^2 F(w)} \leq \varepsilon_0 \|w - w_\star\|_{\nabla^2 F(w)}.$$

4.
$$\|\nabla F_i(w) - \nabla F_i(w_\star)\|_{\nabla^2 F_i(w')^{-1}} \leq (1+\varepsilon_0)\|w - w_\star\|_{\nabla^2 F_i(w')}, \quad \textit{for all } i \in [n],$$
$$\|\nabla F(w) - \nabla F(w_\star)\|_{\nabla^2 F(w')^{-1}} \leq (1+\varepsilon_0)\|w - w_\star\|_{\nabla^2 F(w')}.$$

5.
$$\left\|\nabla^2 F(w)^{1/2}(\nabla^2 F(w)^{-1} - P^{-1})\nabla^2 F(w)^{1/2}\right\| \leq 3\varepsilon_0.$$

We will use the following version of Bernstein's inequality for vectors, which slightly refines a result of Minsker (2017).

Lemma 27 (Bernstein's inequality for vectors) *Let $\{X_i\}_{1\leq i\leq m}$, be a sequence of independent mean zero random vectors in $\mathbb{R}^p$ satisfying $\|X_i\| \leq R$ and $\mathbb{E}[\|X_i\|^2] \leq \varsigma^2$ for all $i \in [m]$, Then*

$$\mathbb{P}\left(\left\|\frac{1}{m}\sum_{i=1}^{m} X_i\right\| \geq t\right) \leq 8\exp\left(-\min\left\{\frac{mt^2}{4\varsigma^2}, \frac{-3mt}{4R}\right\}\right),$$

for all $t \geq \sqrt{\frac{\varsigma^2}{m}} + \frac{R}{3m}$.

Proof The result follows immediately from applying Theorem 7.3.1. of Tropp et al. (2015) to the scaled sequence $\{X_i/m\}_{1\leq i\leq m}$. ■

Our next lemma controls the deviation of the stochastic gradient: $\widehat{\nabla} F(w) - \widehat{\nabla} F(w_\star) - \nabla F(w)$, in the norm $\|\cdot\|_{\nabla^2 F(w')^{-1}}$. This lemma is the key to improving over the local convergence analysis of Dereziński (2022), which requires a gradient batchsize of $\widetilde{\mathcal{O}}(\kappa)$. The improvement is possible thanks to quadratic regularity and Hessian dissimilarity. Quadratic regularity enables us to directly reason in the norms $\left(\|\cdot\|_{\nabla^2 F(w')}, \|\cdot\|_{\nabla^2 F(w')^{-1}}\right)$, while Hessian dissimilarity allows for the tightest control possible over the gradient batchsize.

Lemma 28 *Let $\beta_g \in (0,1)$. Suppose $w, w' \in \mathcal{N}_{\varepsilon_0}(w_\star)$, and $b_g \geq \frac{32\tau_\star^\nu(\mathcal{N}_{\varepsilon_0}(w_\star))\log(\frac{8}{\delta})}{\beta_g^2}$. Then with probability at least $1-\delta$,*

$$\|\widehat{\nabla} F(w) - \widehat{\nabla} F(w_\star) - \nabla F(w)\|_{\nabla^2 F(w')^{-1}} \leq \beta_g \|w - w_\star\|_{\nabla^2 F(w')}.$$

Proof We begin by observing that

$$\left\|\widehat{\nabla} F(w) - \widehat{\nabla} F(w_\star) - \nabla F(w)\right\|^2_{\nabla^2 F(w')^{-1}} = \left\|\frac{1}{b_g}\sum_{i\in\mathcal{B}} \tilde{v}_i\right\|^2,$$

where

$$\tilde{v}_i = \nabla^2 F(w')^{-1/2}(\nabla F_i(w) - \nabla F_i(w_\star) - \nabla F(w)).$$

Now,

$$\begin{aligned}\|\tilde{v}_i\|^2 &\leq 2\|\nabla F_i(w) - \nabla F_i(w_\star)\|^2_{\nabla^2 F(w')^{-1}} + 2\|\nabla F(w)\|^2_{\nabla^2 F(w')^{-1}} \\ &\leq 2\tau_\star^\nu(\mathcal{N}_{\varepsilon_0}(w_\star))\|\nabla F_i(w) - \nabla F_i(w_\star)\|^2_{\nabla^2 F_i(w')^{-1}} + 2\|\nabla F(w)\|^2_{\nabla^2 F(w')^{-1}} \\ &= 2\tau_\star^\nu(\mathcal{N}_{\varepsilon_0}(w_\star))\|\nabla F_i(w) - \nabla F_i(w_\star)\|^2_{\nabla^2 F_i(w')^{-1}} + 2\|\nabla F(w) - \nabla F(w_\star)\|^2_{\nabla^2 F(w')^{-1}}.\end{aligned}$$

Here, the second inequality is due to the definition of Hessian dissimilarity. Now, invoking item 4 of Lemma 26 we reach

$$\begin{aligned}\|\tilde{v}_i\|^2 &\leq 2\tau_\star^\nu(\mathcal{N}_{\varepsilon_0}(w_\star))(1+\varepsilon_0)^2\|w - w_\star\|^2_{\nabla^2 F_i(w')} + 2(1+\varepsilon_0)^2\|w - w_\star\|^2_{\nabla^2 F(w')} \\ &\leq 4(1+\varepsilon_0)^2\tau_\star^\nu(\mathcal{N}_{\varepsilon_0}(w_\star))^2\|w - w_\star\|^2_{\nabla^2 F(w')},\end{aligned}$$

So, for each $i \in \mathcal{B}$, it holds with probability 1 that

$$\|\tilde{v}_i\| \leq 2(1+\varepsilon_0)\tau_\star^\nu(\mathcal{N}_{\varepsilon_0}(w_\star))\|w-w_\star\|_{\nabla^2 F(w')}.$$

Thus, we may set $R = 2(1+\varepsilon_0)\tau_\star^\nu(\mathcal{N}_{\varepsilon_0}(w_\star))\|w-w_\star\|_{\nabla^2 F(w')}$.

Next, observe that

$$\begin{aligned}\mathbb{E}[\|\tilde{v}_i\|]^2 &\leq \mathbb{E}\|\nabla F_i(w)-\nabla F_i(w_\star)\|^2_{\nabla^2 F(w')^{-1}}\\ &\leq \tau_\star^\nu(\mathcal{N}_{\varepsilon_0}(w_\star))\mathbb{E}\|\nabla F_i(w)-\nabla F_i(w_\star)\|^2_{\nabla^2 F_i(w')^{-1}}\\ &\overset{(1)}{\leq} \tau_\star^\nu(\mathcal{N}_{\varepsilon_0}(w_\star))\mathbb{E}[2(1+\varepsilon_0)(F_i(w)-F_i(w_\star)-\langle\nabla F_i(w_\star), w-w_\star\rangle)]\\ &\leq 2(1+\varepsilon_0)\tau_\star^\nu(\mathcal{N}_{\varepsilon_0}(w_\star))\left(F(w)-F(w_\star)\right)\\ &\overset{(2)}{\leq} 2(1+\varepsilon_0)^2\tau_\star^\nu(\mathcal{N}_{\varepsilon_0}(w_\star))\|w-w_\star\|^2_{\nabla^2 F(w')}.\end{aligned}$$

Here (1) uses $F_i(w) \geq F_i(w_\star)+\langle\nabla F_i(w_\star), w-w_\star\rangle + \frac{1}{2(1+\varepsilon_0)}\|w-w_\star\|^2_{\nabla^2 F_i(w_\star)}$, which follows from lower quadratic regularity of F and item 1 of Lemma 26. Last, (2) applies upper quadratic regularity and item 1 of Lemma 26. Hence, the variance is bounded by

$$\varsigma^2 = 2(1+\varepsilon_0)^2\tau_\star^\nu(\mathcal{N}_{\varepsilon_0}(w_\star))\|w-w_\star\|^2_{\nabla^2 F(w')}.$$

Now, we invoke Lemma 27 to find

$$\mathbb{P}\left(\left\|\frac{1}{b_g}\sum_{i\in\mathcal{B}}\tilde{v}_i\right\| \geq \sqrt{\frac{4\varsigma^2\log\left(\frac{8}{\delta}\right)}{b_g}}+\frac{4R\log\left(\frac{8}{\delta}\right)}{3b_g}\right) \leq \delta.$$

The last display immediately implies with probability at least $1-\delta$, that

$$\begin{aligned}&\left\|\widehat{\nabla}F(w)-\widehat{\nabla}F(w_\star)-\nabla F(w)\right\|^2_{\nabla^2 F(w')^{-1}} \leq \frac{8\varsigma^2\log(8/\delta)}{b_g}+\frac{32R^2\log^2(8/\delta)}{9b_g^2}\\ &= \left[\frac{16(1+\varepsilon_0)^2\tau_\star^\nu(\mathcal{N}_{\varepsilon_0}(w_\star))\log(8/\delta)}{b_g}+2\left(\frac{8(1+\varepsilon_0)\tau_\star^\nu(\mathcal{N}_{\varepsilon_0}(w_\star))\log(8/\delta)}{3b_g}\right)^2\right]\|w-w_\star\|^2_{\nabla^2 F(w')}\\ &\leq \beta_g^2\|w-w_\star\|^2_{\nabla^2 F(w')}\end{aligned}$$

where the last inequality follows from $b_g \geq \frac{32\tau_\star^\nu(\mathcal{N}_{\varepsilon_0}(w_\star))\log\left(\frac{8}{\delta}\right)}{\beta_g^2}$ and $\varepsilon \in (0, 1/6)$. The desired claim now follows by taking square roots. ■

The next lemma shows that for sufficiently large b_g (with high probability) the distance to the optimum in the $\nabla^2 F(w_\star)$-norm decreases when an *exact* Newton step based on the current iterate is taken.

Lemma 29 *Let $w_k^{(s)} \in \mathcal{N}_{\varepsilon_0}(w_\star)$, and $\beta_g \in (0,1)$. Suppose the gradient batchsize satisfies $b_g = \mathcal{O}\left(\frac{\tau_\star^\nu(\mathcal{N}_{\varepsilon_0}(w_\star))\log\left(\frac{k+1}{\delta}\right)}{\beta_g^2}\right)$. Then with probability at least $1-\frac{\delta}{(k+1)^2}$,*

$$\|\Delta_k^{(s)}-p_k^{(s)}\|_{\nabla^2 F(w_\star)} \leq (1+\varepsilon_0)\left[(\varepsilon_0+\beta_g)\|\Delta_k^{(s)}\|_{\nabla^2 F(w_\star)}+\beta_g\|\Delta_0^{(s)}\|_{\nabla^2 F(w_\star)}\right].$$

Proof We begin by applying the triangle inequality to reach

$$\begin{aligned}
&\|\Delta_k^{(s)}-p_k^{(s)}\|_{\nabla^2 F(w_k^{(s)})} = \|\Delta_k^{(s)}-\nabla^2 F(w_k^{(s)})^{-1}v_k^{(s)}\|_{\nabla^2 F(w_k^{(s)})} \\
&= \|\nabla^2 F(w_k^{(s)})\Delta_k^{(s)}-(\widehat{\nabla} F(w_k^{(s)})-\widehat{\nabla} F(\hat{w}^{(s)})+\nabla F(\hat{w}^{(s)}))\|_{\nabla^2 F(w_k^{(s)})^{-1}} \\
&\leq \|\nabla^2 F(w_k^{(s)})\Delta_k^{(s)}-\nabla F(w_k^{(s)})\|_{\nabla^2 F(w_k^{(s)})^{-1}} \\
&+ \|\nabla F(w_k^{(s)})-\widehat{\nabla} F(w_k^{(s)})+\widehat{\nabla} F(\hat{w}^{(s)})-\nabla F(\hat{w}^{(s)})\|_{\nabla^2 F(w_k^{(s)})^{-1}}.
\end{aligned}$$

To bound the first term, we apply item 3. of Lemma 26, which yields

$$\|\Delta_k^{(s)}-\nabla^2 F(w_k^{(s)})^{-1}\nabla F(w_k^{(s)})\|_{\nabla^2 F(w_k^{(s)})} \leq \varepsilon_0\|\Delta_k^{(s)}\|_{\nabla^2 F(w_k^{(s)})}.$$

For the second term, the triangle inequality yields

$$\begin{aligned}
&\|\nabla F(w_k^{(s)})-\widehat{\nabla} F(w_k^{(s)})+\widehat{\nabla} F(\hat{w}^{(s)})-\nabla F(\hat{w}^{(s)})\|_{\nabla^2 F(w_k^{(s)})^{-1}} \\
&\leq \|\widehat{\nabla} F(w_k^{(s)})-\widehat{\nabla} F(w_\star)-\nabla F(w_k^{(s)})\|_{\nabla^2 F(w_k^{(s)})^{-1}} \\
&+ \|\widehat{\nabla} F(\hat{w}^{(s)})-\widehat{\nabla} F(w_\star)-\nabla F(\hat{w}^{(s)})\|_{\nabla^2 F(w_k^{(s)})^{-1}}.
\end{aligned}$$

Now, we can apply Lemma 28, to find that

$$\begin{aligned}
&\|\widehat{\nabla} F(w_k^{(s)})-\widehat{\nabla} F(w_\star)-\nabla F(w_k^{(s)})\|_{\nabla^2 F(w_k^{(s)})^{-1}} \leq \beta_g\|\Delta_k^{(s)}\|_{\nabla^2 F(w_k^{(s)})}, \\
&\|\widehat{\nabla} F(\hat{w}^{(s)})-\widehat{\nabla} F(w_\star)-\nabla F(\hat{w}^{(s)})\|_{\nabla^2 F(w_k^{(s)})^{-1}} \leq \beta_g\|\Delta_0^{(s)}\|_{\nabla^2 F(w_k^{(s)})},
\end{aligned}$$

with probability at least $1-\frac{\delta}{(k+1)^2}$. So,

$$\begin{aligned}
&\|\nabla F(w_k^{(s)})-\widehat{\nabla} F(w_k^{(s)})+\widehat{\nabla} F(\hat{w}^{(s)})-\nabla F(\hat{w}^{(s)})\|_{\nabla^2 F(w_k^{(s)})^{-1}} \\
&\leq \beta_g\left(\|\Delta_k^{(s)}\|_{\nabla^2 F(w_k^{(s)})}+\|\Delta_0^{(s)}\|_{\nabla^2 F(w_k^{(s)})}\right),
\end{aligned}$$

Combining the upper bounds on terms 1 and 2, we find

$$\|\Delta_k^{(s)}-p_k^{(s)}\|_{\nabla^2 F(w_k^{(s)})} \leq (\varepsilon_0+\beta_g)\|\Delta_k^{(s)}\|_{\nabla^2 F(w_k^{(s)})}+\beta_g\|\Delta_0^{(s)}\|_{\nabla^2 F(w_k^{(s)})}.$$

Hence applying item 2. of Lemma 26 twice, we conclude

$$\|\Delta_k^{(s)}-p_k^{(s)}\|_{\nabla^2 F(w_\star)} \leq (1+\varepsilon_0)\left[(\varepsilon_0+\beta_g)\|\Delta_k^{(s)}\|_{\nabla^2 F(w_\star)}+\beta_g\|\Delta_0^{(s)}\|_{\nabla^2 F(w_\star)}\right],$$

with probability at least $1-\frac{\delta}{(k+1)^2}$. ■

Next we have the following result, which shows (with high probability) the distance to the optimum of the iterate actually computed by Algorithm 3 is decreasing in the $\nabla^2 F(w_\star)$-norm. In particular, this implies $w_{k+1}^{(s)}$ remains in $\mathcal{N}_{\varepsilon_0}(w_\star)$.

Lemma 30 *Instate the hypotheses of Lemma 29. Then the following items hold with probability at least* $1-\frac{\delta}{(k+1)^2}$.

1. $\|\Delta_{k+1}^{(s)}\|_{\nabla^2 F(w_\star)} \leq \frac{7}{12}\|\Delta_k^{(s)}\|_{\nabla^2 F(w_\star)} + \frac{1}{4}\|\Delta_0^{(s)}\|_{\nabla^2 F(w_\star)}$

2. $w_{k+1}^{(s)} \in \mathcal{N}_{\varepsilon_0}(w_\star)$.

Proof To start off, we apply the triangle inequality to reach

$$\|\Delta_{k+1}^{(s)}\|_{\nabla^2 F(w_\star)} \leq \|\Delta_k^{(s)} - p_k^{(s)}\|_{\nabla^2 F(w_\star)} + \|p_k^{(s)} - \tilde{p}_k^{(s)}\|_{\nabla^2 F(w_\star)}.$$

The first term may be bounded by invoking Lemma 29, so for now we focus on bounding the second term, which represents the error from computing an approximate Newton step. To this end, observe that

$$\begin{aligned}
\|p_k^{(s)} - \tilde{p}_k^{(s)}\|_{\nabla^2 F(w_\star)} &= \left\|\nabla^2 F(w_k^{(s)})^{1/2}(p_k^{(s)} - \tilde{p}_k^{(s)})\right\|_{\nabla^2 F(w_k^{(s)})^{-1/2}\nabla^2 F(w_\star)\nabla^2 F(w_k^{(s)})^{-1/2}} \\
&\overset{(1)}{\leq} (1+\varepsilon_0)\left\|\nabla^2 F(w_k^{(s)})^{1/2}(p_k^{(s)} - \tilde{p}_k^{(s)})\right\| \\
&= (1+\varepsilon_0)\left\|\nabla^2 F(w_k^{(s)})^{1/2}(\nabla^2 F(w_k^{(s)})^{-1} - P^{-1})\nabla^2 F(w_k^{(s)})^{1/2}(\nabla^2 F(w_k^{(s)})^{1/2} p_k^{(s)})\right\| \\
&\overset{(2)}{\leq} \frac{21}{6}\varepsilon_0 \|p_k^{(s)}\|_{\nabla^2 F(w_k^{(s)})} \leq 4\varepsilon_0 \|p_k^{(s)}\|_{\nabla^2 F(w_\star)} \\
&\leq 4\varepsilon_0\left(\|\Delta_k^{(s)}\|_{\nabla^2 F(w_\star)} + \|\Delta_k^{(s)} - p_k^{(s)}\|_{\nabla^2 F(w_\star)}\right),
\end{aligned}$$

where (1) uses item 2 of Lemma 26, and (2) uses item 4 of Lemma 26, along with $\varepsilon_0 \leq 1/6$. Combining the preceding upper bound with our initial bound, we reach

$$\|\Delta_{k+1}^{(s)}\|_{\nabla^2 F(w_\star)} \leq (1+4\varepsilon_0)\|\Delta_k^{(s)} - p_k^{(s)}\|_{\nabla^2 F(w_\star)} + 4\varepsilon_0\|\Delta_k^{(s)}\|_{\nabla^2 F(w_\star)}.$$

Now, invoking Lemma 29 to bound $\|\Delta_k^{(s)} - p_k^{(s)}\|_{\nabla^2 F(w_\star)}$, we find with probability at least $1-\delta/(k+1)^2$, that

$$\|\Delta_{k+1}^{(s)}\|_{\nabla^2 F(w_\star)} \leq \left[(1+\varepsilon_0)(1+4\varepsilon_0)(\varepsilon_0+\beta_g) + 4\varepsilon_0\right]\|\Delta_k^{(s)}\|_{\nabla^2 F(w_\star)} + (1+\varepsilon_0)(1+4\varepsilon_0)\beta_g\|\Delta_0^{(s)}\|_{\nabla^2 F(w_\star)}.$$

Using $\varepsilon_0 \leq \frac{1}{6}$, the preceding display becomes

$$\|\Delta_{k+1}^{(s)}\|_{\nabla^2 F(w_\star)} \leq \left(\frac{1}{3} + 2\beta_g\right)\|\Delta_k^{(s)}\|_{\nabla^2 F(w_\star)} + 2\beta_g\|\Delta_0^{(s)}\|_{\nabla^2 F(w_\star)}.$$

Now, let us set $\beta_g = \frac{1}{8}$, then the preceding display simplifies to

$$\|\Delta_{k+1}^{(s)}\|_{\nabla^2 F(w_\star)} \leq \frac{7}{12}\|\Delta_k^{(s)}\|_{\nabla^2 F(w_\star)} + \frac{1}{4}\|\Delta_0^{(s)}\|_{\nabla^2 F(w_\star)},$$

which proves the first claim. To see the second claim, note that

$$\max\{\|\Delta_k^{(s)}\|_{\nabla^2 F(w_\star)}, \|\Delta_0^{(s)}\|_{\nabla^2 F(w_\star)}\} \leq \varepsilon_0 \nu^{3/2}/(2M),$$

as $w_k^{(s)}, \hat{w}^{(s)} \in \mathcal{N}_{\varepsilon_0}(w_\star)$. Hence, the second claim follows immediately from the first. ■

Lemma 31 (One-stage analysis) *Let $\hat{w}^{(s)} \in \mathcal{N}_{\varepsilon_0}(w_\star)$. Run Algorithm 3 with $m = 6$ inner iterations and gradient batchsize satisfies $b_g = \mathcal{O}\left(\tau_\star^\nu(\mathcal{N}_{\varepsilon_0}(w_\star))\log\left(\frac{m+1}{\delta}\right)\right)$. Then with probability at least $1-\delta$:*

1. *$\hat{w}^{(s+1)} \in \mathcal{N}_{\frac{2}{3}\varepsilon_0}(w_\star)$.*

2. *$F(\hat{w}^{(s+1)}) - F(w_\star) \leq \frac{2}{3}(F(\hat{w}^{(s)}) - F(w_\star))$.*

Proof As $\hat{w}^{(s)} \in \mathcal{N}_{\varepsilon_0}(w_\star)$, it follows by union bound that the conclusions of Lemma 30 hold for all $w_k^{(s)}$, where $k \in \{0,\ldots,m-1\}$, with probability at least

$$1 - \sum_{k=0}^{m-1}\frac{\delta}{(m+1)^2} = 1 - \frac{m}{(m+1)^2}\delta \geq 1-\delta.$$

Consequently,

$$\|\Delta_m^{(s)}\|_{\nabla^2 F(w_\star)} \leq \frac{7}{12}\|\Delta_{m-1}^{(s)}\|_{\nabla^2 F(w_\star)} + \frac{1}{4}\|\Delta_0^{(s)}\|_{\nabla^2 F(w_\star)}.$$

We now recurse on the previous display, and use $m = 6 > \frac{\log(1/15)}{\log(7/12)}$, to reach

$$\begin{aligned}
\|\Delta_m^{(s)}\|_{\nabla^2 F(w_\star)} &\leq \left(\frac{7}{12}\right)^m \|\Delta_0^{(s)}\|_{\nabla^2 F(w_\star)} + \left(\sum_{k=0}^{m-1}\left(\frac{7}{12}\right)^k\right)\frac{1}{4}\|\Delta_0^{(s)}\|_{\nabla^2 F(w_\star)} \\
&\leq \frac{1}{15}\|\Delta_0^{(s)}\|_{\nabla^2 F(w_\star)} + \frac{1}{4(1-\frac{7}{12})}\|\Delta_0^{(s)}\|_{\nabla^2 F(w_\star)} \\
&= \left(\frac{1}{15}+\frac{3}{5}\right)\|\Delta_0^{(s)}\|_{\nabla^2 F(w_\star)} \leq \frac{2}{3}\|\Delta_0^{(s)}\|_{\nabla^2 F(w_\star)}.
\end{aligned}$$

Hence $\hat{w}^{(s+1)} = w_m^{(s)} \in \mathcal{N}_{\frac{2}{3}\varepsilon_0}(w_\star)$. Using this last inclusion, and applying upper quadratic regularity, followed by lower quadratic regularity, we find

$$\begin{aligned}
F(\hat{w}^{(s+1)}) - F(w_\star) &\leq \frac{1+\varepsilon_0}{2}\|\Delta_m^{(s)}\|^2_{\nabla^2 F(w_\star)} \leq \frac{1+\varepsilon_0}{2}\frac{4}{9}\|\Delta_0^{(s)}\|^2_{\nabla^2 F(w_\star)} \\
&\leq (1+\varepsilon_0)^2\frac{4}{9}\left(F(\hat{w}^{(s)}) - F(w_\star)\right) \leq \frac{2}{3}\left(F(\hat{w}^{(s)}) - F(w_\star)\right),
\end{aligned}$$

as desired. ■

B.4.3 Proof of Theorem 19

We now come to the proof of Theorem 19, which is reduced to union bounding over the conclusion of Lemma 31.

Proof By hypothesis, we may invoke Lemma 31 to conclude the output of the first outer iteration satisfies

$$F(\hat{w}^{(1)}) - F(w_\star) \leq \frac{2}{3}(F(w_0) - F(w_\star)), \quad \text{and } \hat{w}^{(1)} \in \mathcal{N}_{\varepsilon_0}(w_\star)$$

with probability at least $1-\delta$. Hence we can apply Lemma 31 again to $\hat{w}^{(s)}$, the output of the second outer iteration. Repeating this logic for all the remaining outer iterations, we find by union bound, that with probability at least $1-s\delta$,

$$F(\hat{w}^{(s)}) - F(w_\star) \leq \left(\frac{2}{3}\right)^s (F(w_0) - F(w_\star)) \leq \epsilon.$$

The theorem now follows by scaling δ down by $3\log\left((F(w_0) - F(w_\star))/\epsilon\right)$. ■

References

Ahmed Alaoui and Michael W Mahoney. Fast randomized kernel ridge regression with statistical guarantees. In *Advances in Neural Information Processing Systems*, 2015.

Zeyuan Allen-Zhu. Katyusha: the first direct acceleration of stochastic gradient methods. *Journal of Machine Learning Research*, 18(221):1–51, 2018.

Yossi Arjevani and Ohad Shamir. Oracle complexity of second-order methods for finite-sum problems. In *International Conference on Machine Learning*, 2017.

Yossi Arjevani, Ohad Shamir, and Ron Shiff. Oracle complexity of second-order methods for smooth convex optimization. *Mathematical Programming*, 178:327–360, 2019.

Francis Bach. Sharp analysis of low-rank kernel matrix approximations. In *Conference on Learning Theory*, 2013.

Raghu Bollapragada, Jorge Nocedal, Dheevatsa Mudigere, Hao-Jun Shi, and Ping Tak Peter Tang. A progressive batching L-BFGS method for machine learning. In *International Conference on Machine Learning*, 2018.

Raghu Bollapragada, Richard H Byrd, and Jorge Nocedal. Exact and inexact subsampled Newton methods for optimization. *IMA Journal of Numerical Analysis*, 39(2):545–578, 2019.

Stephen P Boyd and Lieven Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

Richard H Byrd, Gillian M Chin, Will Neveitt, and Jorge Nocedal. On the use of stochastic Hessian information in optimization methods for machine learning. *SIAM Journal on Optimization*, 21(3):977–995, 2011.

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

Andrea Caponnetto and Ernesto De Vito. Optimal rates for the regularized least-squares algorithm. *Foundations of Computational Mathematics*, 7:331–368, 2007.

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.

Michael B Cohen, Cameron Musco, and Christopher Musco. Input sparsity time low-rank approximation via ridge leverage score sampling. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 1758–1777. SIAM, 2017.

Aaron Defazio, Francis Bach, and Simon Lacoste-Julien. SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives. In *Advances in Neural Information Processing Systems*, 2014.

Michał Dereziński. Stochastic Variance-Reduced Newton: Accelerating Finite-Sum Minimization with Large Batches. *arXiv preprint arXiv:2206.02702*, 2022.

Michal Derezinski, Feynman T Liang, Zhenyu Liao, and Michael W Mahoney. Precise expressions for random projections: Low-rank approximation and randomized Newton. In *Advances in Neural Information Processing Systems*, 2020.

Michal Derezinski, Jonathan Lacotte, Mert Pilanci, and Michael W Mahoney. Newton-LESS: Sparsification without trade-offs for the sketched Newton update. In *Advances in Neural Information Processing Systems*, 2021.

Nikita Doikov. Minimizing quasi-self-concordant functions by gradient regularization of newton method. *arXiv preprint arXiv:2308.14742*, 2023.

Murat A Erdogdu and Andrea Montanari. Convergence rates of sub-sampled Newton methods. In *Advances in Neural Information Processing Systems*, 2015.

Zachary Frangella, Pratik Rathore, Shipu Zhao, and Madeleine Udell. SketchySGD: Reliable stochastic optimization via randomized curvature estimates. *arXiv preprint arXiv:2211.08597*, 2023a.

Zachary Frangella, Joel A. Tropp, and Madeleine Udell. Randomized Nyström preconditioning. *SIAM Journal on Matrix Analysis and Applications*, 44(2):718–752, 2023b.

Nidham Gazagnadou, Robert Gower, and Joseph Salmon. Optimal mini-batch and step sizes for SAGA. In *International Conference on Machine Learning*, 2019.

Alex Gittens and Michael W Mahoney. Revisiting the Nyström method for improved large-scale machine learning. *The Journal of Machine Learning Research*, 17(1):3977–4041, 2016.

Alon Gonen, Francesco Orabona, and Shai Shalev-Shwartz. Solving ridge regression using sketched preconditioned SVRG. In *International Conference on Machine Learning*, 2016.

Robert Gower, Donald Goldfarb, and Peter Richtárik. Stochastic block BFGS: Squeezing more curvature out of data. In *International Conference on Machine Learning*, 2016.

Robert Gower, Nicolas Le Roux, and Francis Bach. Tracking the gradients using the Hessian: A new look at variance reducing stochastic methods. In *International Conference on Artificial Intelligence and Statistics*, 2018.

Robert Gower, Dmitry Kovalev, Felix Lieder, and Peter Richtárik. RSN: randomized subspace Newton. In *Advances in Neural Information Processing Systems*, 2019a.

Robert Mansel Gower, Nicolas Loizou, Xun Qian, Alibek Sailanbayev, Egor Shulgin, and Peter Richtárik. SGD: General analysis and improved rates. In *International Conference on Machine Learning*, 2019b.

Nicholas J Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, 2002.

Daniel Hsu, Sham M Kakade, and Tong Zhang. Random design analysis of ridge regression. *Foundations of Computational Mathematics*, 3(14):569–600, 2014.

Rie Johnson and Tong Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems*, 2013.

Sai Praneeth Karimireddy, Sebastian U Stich, and Martin Jaggi. Global linear convergence of Newton's method without strong-convexity or Lipschitz gradients. *arXiv preprint arXiv:1806.00413*, 2018.

Dmitry Kovalev, Konstantin Mishchenko, and Peter Richtárik. Stochastic Newton and cubic Newton methods with simple local linear-quadratic rates. *arXiv preprint arXiv:1912.01597*, 2019.

Dmitry Kovalev, Samuel Horváth, and Peter Richtárik. Don't jump through hoops and remove those loops: SVRG and Katyusha are better without the outer loop. In *International Conference on Algorithmic Learning Theory*, 2020.

Jonathan Lacotte, Yifei Wang, and Mert Pilanci. Adaptive Newton sketch: Linear-time optimization with quadratic convergence and effective Hessian dimensionality. In *International Conference on Machine Learning*, 2021.

Xiang Li, Shusen Wang, and Zhihua Zhang. Do subsampled Newton methods work for high-dimensional data? In *AAAI Conference on Artificial Intelligence*, 2020.

Yanli Liu, Fei Feng, and Wotao Yin. Acceleration of SVRG and Katyusha X by inexact preconditioning. In *International Conference on Machine Learning*, 2019.

Ulysse Marteau-Ferey, Francis Bach, and Alessandro Rudi. Globally convergent Newton methods for ill-conditioned generalized self-concordant losses. In *Advances in Neural Information Processing Systems*, 2019a.

Ulysse Marteau-Ferey, Dmitrii Ostrovskii, Francis Bach, and Alessandro Rudi. Beyond least-squares: Fast rates for regularized empirical risk minimization through self-concordance. In *Conference on Learning Theory*, 2019b.

Per-Gunnar Martinsson and Joel A Tropp. Randomized numerical linear algebra: Foundations and algorithms. *Acta Numerica*, 29:403–572, 2020.

Song Mei and Andrea Montanari. The generalization error of random features regression: Precise asymptotics and the double descent curve. *Communications on Pure and Applied Mathematics*, 75(4):667–766, 2022.

Stanislav Minsker. On some extensions of Bernstein's inequality for self-adjoint operators. *Statistics & Probability Letters*, 127:111–119, 2017.

Philipp Moritz, Robert Nishihara, and Michael Jordan. A linearly-convergent stochastic L-BFGS algorithm. In *Artificial Intelligence and Statistics*, 2016.

Eric Moulines and Francis Bach. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. In *Advances in Neural Information Processing Systems*, 2011.

Cameron Musco and Christopher Musco. Randomized block Krylov methods for stronger and faster approximate singular value decomposition. In *Advances in Neural Information Processing Systems*, 2015.

Sen Na, Michał Dereziński, and Michael W Mahoney. Hessian averaging in stochastic Newton methods achieves superlinear convergence. *Mathematical Programming*, pages 1–48, 2022.

Arkadi Nemirovski, Anatoli Juditsky, Guanghui Lan, and Alexander Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM Journal on optimization*, 19(4):1574–1609, 2009.

Yurii Nesterov. *Lectures on Convex Optimization*. Springer, 2018.

Jorge Nocedal and Stephen J Wright. *Numerical Optimization*. Springer, 1999.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

Mert Pilanci and Martin J Wainwright. Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence. *SIAM Journal on Optimization*, 27(1):205–245, 2017.

Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*, 2007.

Farbod Roosta-Khorasani and Michael W Mahoney. Sub-sampled Newton methods. *Mathematical Programming*, 174(1):293–326, 2019.

Alessandro Rudi, Luigi Carratino, and Lorenzo Rosasco. Falkon: An optimal large scale kernel method. In *Advances in Neural Information Processing Systems*, 2017.

Danica J Sutherland. Fixing an error in Caponnetto and de Vito (2007). *arXiv preprint arXiv:1702.02982*, 2017.

Joel A Tropp, Alp Yurtsever, Madeleine Udell, and Volkan Cevher. Fixed-rank approximation of a positive-semidefinite matrix from streaming data. In *Advances in Neural Information Processing Systems*, 2017.

Joel A Tropp et al. An introduction to matrix concentration inequalities. *Foundations and Trends® in Machine Learning*, 8(1-2):1–230, 2015.

Joaquin Vanschoren, Jan N. van Rijn, Bernd Bischl, and Luis Torgo. OpenML: Networked science in machine learning. *SIGKDD Explorations*, 15(2):49–60, 2013.

Jialei Wang and Tong Zhang. Utilizing second order information in minibatch stochastic variance reduced proximal iterations. *Journal of Machine Learning Research*, 20(42): 1–56, 2019.

Christopher Williams and Matthias Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, 2000.

Blake E Woodworth and Nati Srebro. Tight complexity bounds for optimizing composite objectives. In *Advances in Neural Information Processing Systems*, 2016.

Haishan Ye, Luo Luo, and Zhihua Zhang. Approximate Newton methods. *Journal of Machine Learning Research*, 22(66):1–41, 2021.

Shipu Zhao, Zachary Frangella, and Madeleine Udell. NysADMM: faster composite convex optimization via low-rank approximation. In *International Conference on Machine Learning*, 2022.