Non-PSD Matrix Sketching with Applications to Regression and Optimization

Zhili Feng[1] **Fred Roosta**[2] **David P. Woodruff**[3]

[1]Machine Learning Department, Carnegie Mellon University,
[2]School of Mathematics and Physics, University of Queensland,
[3] Computer Science Department, Carnegie Mellon University,

Abstract

A variety of dimensionality reduction techniques have been applied for computations involving large matrices. The underlying matrix is randomly compressed into a smaller one, while approximately retaining many of its original properties. As a result, much of the expensive computation can be performed on the small matrix. The sketching of positive semidefinite (PSD) matrices is well understood, but there are many applications where the related matrices are not PSD, including Hessian matrices in non-convex optimization and covariance matrices in regression applications involving complex numbers. In this paper, we present novel dimensionality reduction methods for non-PSD matrices, as well as their "square-roots", which involve matrices with complex entries. We show how these techniques can be used for multiple downstream tasks. In particular, we show how to use the proposed matrix sketching techniques for both convex and non-convex optimization, ℓ_p-regression for every $1 \leq p \leq \infty$, and vector-matrix-vector queries.

1 INTRODUCTION

Many modern machine learning tasks involve massive datasets, where an input matrix $\mathbf{A} \in \mathbb{R}^{n\times d}$ is such that $n \gg d$. In a number of cases, $\mathbf{A}$ is highly redundant. For example, if we want to solve the ordinary least squares problem $\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2^2$, one can solve it exactly given only $\mathbf{A}^T\mathbf{A}$ and $\mathbf{A}^T\mathbf{b}$. To exploit this redundancy, numerous techniques have been developed to reduce the size of $\mathbf{A}$. Such dimensionality reduction techniques are used to speed up various optimization tasks and are often referred to as *sketching*; for a survey, see Woodruff et al. [2014].

A lot of previous work has focused on sketching PSD matrices. For example, the Hessian matrices in convex optimization [Xu et al., 2016], the covariance matrices $\mathbf{X}^\top\mathbf{X}$ in regression over the reals, and quadratic form queries $\mathbf{x}^\top\mathbf{A}\mathbf{x}$ [Andoni et al., 2016]. Meanwhile, less is understood for non-PSD matrices. These matrices are naturally associated with complex matrices: the Hessian of a non-convex optimization problem can be decomposed into $\mathbf{H} = \mathbf{X}^\top\mathbf{X}$ where $\mathbf{X}$ is a matrix with complex entries, and a complex design matrix $\mathbf{X}$ has a non-PSD covariance matrix. However, almost all sketching techniques were developed for matrices with entries in the real field $\mathbb{R}$. While some results carry over to the complex numbers $\mathbb{C}$ (e.g., Tropp et al. [2015] develops concentration bounds that work for complex matrices), many do not and seem to require non-trivial extensions. In this work, we show how to efficiently sketch non-PSD matrices and extend several existing sketching results to the complex field. We also show how to use these in optimization, for both convex and non-convex problems, the sketch-and-solve paradigm for complex ℓ_p-regression with $1 \leq p \leq \infty$, as well as vector-matrix-vector product queries.

Finite-sum Optimization. We consider optimization problems of the form

$$\min_{\mathbf{x}\in\mathbb{R}^d} F(\mathbf{x}) \triangleq \frac{1}{n}\sum_{i=1}^{n} f_i(\mathbf{a}_i^\intercal\mathbf{x}) + r(\mathbf{x}), \tag{1}$$

where $n \gg d \geq 1$, each $f_i : \mathbb{R} \to \mathbb{R}$ is a smooth but possibly non-convex function, $r(\mathbf{x})$ is a regularization term, and $\mathbf{a}_i \in \mathbb{R}^d, i = 1, \ldots, n$, are given. Problems of the form (1) are abundant in machine learning [Shalev-Shwartz and Ben-David, 2014]. Concrete examples include robust linear regression using Tukey's biweight loss [Beaton and Tukey, 1974], i.e., $f_i(\langle\mathbf{a}_i, \mathbf{x}\rangle) = (\mathbf{a}_i^\intercal\mathbf{x} - b_i)^2/(1 + (\mathbf{a}_i^\intercal\mathbf{x} - b_i)^2)$, where $b_i \in \mathbb{R}$, and non-linear binary classification [Xu et al., 2020], i.e., $f_i(\langle\mathbf{a}_i, \mathbf{x}\rangle) = (1/(1 + \exp(-\mathbf{a}_i^\intercal\mathbf{x})) - b_i)^2$, where $b_i \in \{0, 1\}$ is the class label. By incorporating curvature information, second-order methods are gaining popularity over first-order methods in certain applications. However, when $n \gg d \geq 1$, operations involving the Hessian of F constitute a computational bottleneck. To this end, random-

Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI 2021), PMLR 161:1841–1851.

ized Hessian approximations have shown great success in reducing computational complexity ([Roosta and Mahoney, 2019, Xu et al., 2016, 2019, Pilanci and Wainwright, 2017, Erdogdu and Montanari, 2015, Bollapragada et al., 2019]).

In the context of (1), it is easy to see that the Hessian of F can be written as $\nabla^2 F(\mathbf{x}) = \sum_{i=1}^n f_i''(\mathbf{a}_i^\intercal \mathbf{x})\mathbf{a}_i\mathbf{a}_i^\intercal/n + \nabla^2 r(\mathbf{x}) = \mathbf{A}^\intercal \mathbf{D}(\mathbf{x})\mathbf{A}/n + \nabla^2 r(\mathbf{x})$, where $\mathbf{A}^\intercal = \begin{bmatrix}\mathbf{a}_1, \ldots, \mathbf{a}_n\end{bmatrix} \in \mathbb{R}^{d\times n}$ and $\mathbf{D}(\mathbf{x}) = \text{diag}\left[f_1''(\mathbf{a}_1^\intercal \mathbf{x})\ f_2''(\mathbf{a}_2^\intercal \mathbf{x})\ \ldots\ f_n''(\mathbf{a}_n^\intercal \mathbf{x})\right] \in \mathbb{R}^{n\times n}$. Of particular interest in this work is the application of randomized matrix approximation techniques [Woodruff et al., 2014, Mahoney, 2011, Drineas and Mahoney, 2016], in particular, constructing a random sketching matrix $\mathbf{S}$ to ensure that $\mathbf{H}(\mathbf{x}) \triangleq \mathbf{A}^\intercal \mathbf{D}^{1/2}\mathbf{S}^\intercal \mathbf{S}\mathbf{D}^{1/2}\mathbf{A} + \nabla^2 r(\mathbf{x}) \approx \mathbf{A}^\intercal \mathbf{D}\mathbf{A}/n + \nabla^2 r(\mathbf{x}) = \nabla^2 F(\mathbf{x})$. Notice that $\mathbf{D}^{1/2}\mathbf{A}$ may have complex entries if f_i is non-convex.

The Sketch-and-Solve Paradigm for Regression. In the overconstrained least squares regression problem, the task is to solve $\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{b}\|$ for some norm $\|\cdot\|$, and here we focus on the wide class of ℓ_p-norms, where for a vector y, $\|y\|_p = (\sum_j |y_j|^p)^{1/p}$. Setting the value p allows for adjusting the sensitivity to outliers; for $p < 2$ the regression problem is often considered more robust than least squares because one does not square the differences, while for $p > 2$ the problem is considered more sensitive to outliers than least squares. The different p-norms also have statistical motivations: for instance, the ℓ_1-regression solution is the maximum likelihood estimator given i.i.d. Laplacian noise. Approximation algorithms based on sampling and sketching have been thoroughly studied for ℓ_p-regression, see, e.g., [Clarkson, 2005, Clarkson et al., 2016, Dasgupta et al., 2009, Meng and Mahoney, 2013, Sohler and Woodruff, 2011, Woodruff and Zhang, 2013, Clarkson and Woodruff, 2017, Wang and Woodruff, 2019]. These algorithms typically follow the sketch-and-solve paradigm, whereby the dimensions of $\mathbf{A}$ and $\mathbf{b}$ are reduced, resulting in a much smaller instance of ℓ_p-regression, which is tractable. In the case of $p = \infty$, sketching is used inside of an optimization method to speed up linear programming-based algorithms [Cohen et al., 2019].

To highlight some of the difficulties in extending ℓ_p-regression algorithms to the complex numbers, consider two popular cases, of ℓ_1 and ℓ_∞-regression. The standard way of solving these regression problems is by formulating them as linear programs. However, the complex numbers are not totally ordered, and linear programming algorithms therefore do not work with complex inputs. Stepping back, what even is the meaning of the ℓ_p-norm of a complex vector y? In the definition above $\|y\|_p = (\sum_j |y_j|^p)^{1/p}$, and $|y_j|$ denotes the modulus of the complex number, i.e., if $y_j = a + b \cdot i$, where $i = \sqrt{-1}$, then $|y_j| = \sqrt{a^2 + b^2}$. Thus the ℓ_p-regression problem is really a question about minimizing the p-norm of a sum of Euclidean lengths of vectors. As we show later, this problem is very different than ℓ_p regressions over the reals.

Vector-matrix-vector queries. Many applications require queries of the form $\mathbf{u}^\top \mathbf{M}\mathbf{v}$, which we call vector-matrix-vector queries, see, e.g., Rashtchian et al. [2020]. For example, if $\mathbf{M}$ is the adjacency matrix of a graph, then $\mathbf{u}^\top \mathbf{M}\mathbf{v}$ answers whether there exists an edge between pair $\{\mathbf{u}, \mathbf{v}\}$. These queries are also useful for independent set queries, cut queries, etc. Many past works have studied how to sketch positive definite $\mathbf{M}$ (see, e.g., Andoni et al. [2016]), but it remains unclear how to handle the case when $\mathbf{M}$ is non-PSD or has complex entries.

Contributions. We consider non-PSD matrices and their "square-roots", which are complex matrices, in the context of optimization and the sketch-and-solve paradigm. Our goal is to provide tools for handling such matrices in a number of different problems, and to the best of our knowledge, is the first work to systematically study dimensionality reduction techniques for such matrices.

For optimization of (1), where each f_i is potentially non-convex, we investigate non-uniform data-aware methods to construct a sampling matrix $\mathbf{S}$ based on a new concept of leverage scores for complex matrices. In particular, we propose a hybrid deterministic-randomized sampling scheme, which is shown to have important properties for optimization. We show that our sampling schemes can guarantee appropriate matrix approximations (see (4) and (5)) with competitive sampling complexities. Subsequently, we investigate the application of such sampling schemes in the context of convex and non-convex Newton-type methods for (1).

For complex ℓ_p-regression, we use Dvoretsky-type embeddings as well as an isometric embedding from ℓ_1 to ℓ_∞ to construct oblivious embeddings from an instance of a complex ℓ_p-regression problem to a real-valued ℓ_p-regression problem, for $p \in [1, \infty]$. Our algorithm runs in $\mathcal{O}((\mathbf{nnz}(\mathbf{A}) + \mathbf{poly}(d/\epsilon)))$ time for constant $p \in [1, \infty)$, and $\mathcal{O}(\mathbf{nnz}(\mathbf{A})2^{\tilde{O}(1/\epsilon^2)})$ time for $p = \infty$. Here $\mathbf{nnz}(\mathbf{A})$ denotes the number of non-zero entries of the matrix $\mathbf{A}$.

For vector-matrix-vector queries, we show that if the non-PSD matrix has the form $\mathbf{M} = \mathbf{A}^\top \mathbf{B}$, then we can approximately compute $\mathbf{u}^\top \mathbf{M}\mathbf{v}$ in just $\mathcal{O}(\mathbf{nnz}(\mathbf{A}) + n/\epsilon^2)$ time, whereas the naïve approach takes $nd^2 + d^2 + d$ time.

Notation. Vectors and matrices are denoted by bold lower-case and bold upper-case letters, respectively, e.g., $\mathbf{v}$ and $\mathbf{V}$. We use regular lower-case and upper-case letters to denote scalar constants, e.g., d or L. For a complex vector $\mathbf{v}$, its real and conjugate transposes are respectively denoted by $\mathbf{v}^\intercal$ and $\mathbf{v}^*$. For two vectors $\mathbf{v}, \mathbf{w}$, their inner-product is denoted by $\langle \mathbf{v}, \mathbf{w}\rangle$. For a vector $\mathbf{v}$ and a matrix $\mathbf{V}$, $\|\mathbf{v}\|_p$, $\|\mathbf{V}\|$, and $\|\mathbf{V}\|_F$ denote vector ℓ_p norm, matrix spectral norm, and Frobenius norm, respectively. For $\|\mathbf{v}\|_2$, we write $\|\mathbf{v}\|$ as an abbreviation. Let $|\mathbf{V}|$ denote the entry-wise modulus of

matrix $\mathbf{V}$. Let $\mathbf{V}_{i,j}$ denote the (i,j)-th entry, $\mathbf{V}_i = \mathbf{V}_{i,*}$ be the i-th row, and $\mathbf{V}_{*,j}$ be the j-th column. The iteration counter for the main algorithm appears as a subscript, e.g., $\mathbf{p}_k$. For two symmetric matrices $\mathbf{A}$ and $\mathbf{B}$, the Löwner partial order $\mathbf{A} \succeq \mathbf{B}$ indicates that $\mathbf{A} - \mathbf{B}$ is symmetric positive semi-definite. $\mathbf{A}^\dagger$ denotes the Moore-Penrose generalized inverse of matrix $\mathbf{A}$. For a scalar d, we let $\mathbf{poly}(d)$ be a polynomial in d. We let $\mathrm{diag}(\cdot)$ denote a diagonal matrix.

Here we give the necessary definitions.

Definition 1 (Well-conditioned basis and ℓ_p leverage scores)**.** *An $n \times d$ matrix $\mathbf{U}$ is an (α, β, p)-well-conditioned basis for the column span of $\mathbf{A}$ if (i) $(\sum_{i\in[n]} \sum_{j\in[d]} |\mathbf{U}_{ij}|^p)^{1/p} \le \alpha$. (ii) For all $\mathbf{x} \in \mathbb{R}^d$, $\|\mathbf{x}\|_q \le \beta \|\mathbf{U}\mathbf{x}\|_p$, where $1/p + 1/q = 1$. (iii) The column span of $\mathbf{U}$ is equal to the column span of $\mathbf{A}$. For such a well conditioned basis, $\|\mathbf{U}_{i*}\|_p^p$ is defined to be the ℓ_p leverage score of the i-th row of $\mathbf{A}$. The ℓ_p leverage scores are* not *invariant to the choice of well-conditioned basis.*

Definition 2 (ℓ_p Auerbach Basis)**.** *An Auerbach basis $\mathbf{A}$ of $\mathbf{U} \in \mathbb{R}^{n\times d}$ is such that: (i) span$(\mathbf{U})$ = span$(\mathbf{A})$. (ii) For all $j \in [d]$, $\|\mathbf{A}_{*j}\|_p = 1$. (iii) For all $\mathbf{x} \in \mathbb{R}^d$, $d^{-1/q}\|\mathbf{x}\|_q \le \|\mathbf{x}\|_\infty \le \|\mathbf{A}\mathbf{x}\|_p$, where $1/p + 1/q = 1$*

Definition 3 (ℓ_p-subspace embedding)**.** *Let $\mathbf{A} \in \mathbb{R}^{n\times d}$, $\mathbf{S} \in \mathbb{C}^{s\times n}$. We call $\mathbf{S}$ an ϵ ℓ_p-subspace embedding if for all $\mathbf{x} \in \mathbb{C}^d$, $\|\mathbf{A}\mathbf{x}\|_p \le \|\mathbf{S}\mathbf{A}\mathbf{x}\|_p \le (1+\epsilon)\|\mathbf{A}\mathbf{x}\|_p$.*

2 SKETCHING NON-PSD HESSIANS FOR NON-CONVEX OPTIMIZATION

We first present our sketching strategies and then apply them to an efficient solution to (1) using different optimization algorithms. All the proofs are in the supplementary material.

2.1 COMPLEX LEVERAGE SCORE SAMPLING

Algorithm 1 Construct Leverage Score Sampling Matrix

1: **Input:** $\mathbf{D}^{1/2}\mathbf{A} \in \mathbb{C}^{n\times d}$, number s of samples, empty matrices $\mathbf{R} \in \mathbb{R}^{s\times s}$ and $\mathbf{\Omega} \in \mathbb{R}^{n\times s}$
2: Compute SVD of $\mathbf{D}^{1/2}\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$
3: **for** $i \in [n]$ **do**
4: Calculate the i^{th} leverage score $\ell_i = \|\mathbf{U}_{i,*}\|^2$
5: **for** $j \in [s]$ **do**
6: Pick row i independently and with replacement with probability $p_i = \frac{\ell_i}{\sum_i \ell_i}$
7: Set $\mathbf{\Omega}_{i,j} = 1$ and $\mathbf{R}_{i,i} = \frac{1}{\sqrt{sp_i}}$
8: **Output:** $\mathbf{S} = \mathbf{R} \cdot \mathbf{\Omega}^\intercal$

It is well-known that leverage score sampling gives an ϵ ℓ_2-subspace embedding for real matrices with high probability, see, e.g., Woodruff et al. [2014]. Here we extend the result to the complex field:

Theorem 1. *For $i \in [n]$, let $\tilde{\ell}_i \ge \ell_i$ be a constant overestimate to the leverage score of the i^{th} row of $\mathbf{B} \in \mathbb{C}^{n\times d}$. Assume that $\mathbf{B}^\intercal\mathbf{B} \in \mathbb{R}^{d\times d}$. Let $p_i = \tilde{\ell}_i / \sum_{j\in[n]} \tilde{\ell}_j$ and $t = cd\epsilon^{-2}\log(d/\delta)$ for a large enough constant c. We sample t rows of A where row i is sampled with probability p_i and rescaled to $1/\sqrt{tp_i}$. Denote the sampled matrix by $\mathbf{C}$. Then with probability $1-\delta$, $\mathbf{C}$ satisfies: $\mathbf{B}^\intercal\mathbf{B} - \epsilon\mathbf{B}^*\mathbf{B} \preceq \mathbf{C}^\intercal\mathbf{C} \preceq \mathbf{B}^\intercal\mathbf{B} + \epsilon\mathbf{B}^*\mathbf{B}$.*

Theorem 2. *Under the same assumptions and notation in Theorem 1, let $t = cd\gamma\epsilon^{-2}\log(d/\delta)$, where $\gamma = \left\|\sum_{i\in[n]} \frac{\|\mathbf{B}_i\|^2}{\tilde{\ell}_i}\mathbf{B}_i^*\mathbf{B}_i\right\|$. Then with probability $1-\delta$, $\mathbf{C}$ satisfies $\|\mathbf{C}^\intercal\mathbf{C} - \mathbf{B}^\intercal\mathbf{B}\| < \epsilon$.*

Remark 1. *It is hard to directly compare Theorem 2 to the sample complexity of Xu et al. [2019], where they require $t \ge \frac{\widehat{K}^2}{\epsilon^2}\log(2d/\delta)$, $\widehat{K} \ge \|\mathbf{B}^*\mathbf{B}\| = \mathcal{O}(\sigma_1^2)$. To apply Theorem 2 to a Hessian of the form $\mathbf{A}^\intercal\mathbf{D}\mathbf{A}$, one should set $\mathbf{B} = \mathbf{D}^{1/2}\mathbf{A}$. Compared to the previously proposed sketching $\mathbf{A}^\intercal\mathbf{S}^\intercal\mathbf{S}\mathbf{D}\mathbf{A}$ for non-convex F, our proposed sketching $\mathbf{A}^\intercal\mathbf{D}^{1/2}\mathbf{S}^\intercal\mathbf{S}\mathbf{D}^{1/2}\mathbf{A}$ in practice often has better performance (see Section 2.2). We conjecture this is because $\mathbf{B}$ has several large singular values, but many rows have small row norms $\|\mathbf{B}_i\|^2 \ll \tilde{\ell}_i$. Hence $d\gamma$ can be much smaller than $\widehat{K}^2$.*

Note that Theorem 1 cannot be guaranteed by row norm sampling. Consider $\mathbf{B} = \mathrm{diag}(\infty, 1)$. Then row norm sampling will never sample the second row, yet the leverage scores of both rows are 1. All leverage scores can be computed up to a constant factor simultaneously in $\mathcal{O}\left((\mathbf{nnz}(\mathbf{A}) + d^2)\log n\right)$ time. See the appendix for details.

Hybrid of Randomized-Deterministic Sampling. We propose Algorithm 2 to speed up the approximation of Hessian matrices by deterministically sampling the "heavy" rows. The proposed method provably outperforms the vanilla leverage score sampling algorithm under a relaxed RIP condition.

Theorem 3. *Let $\mathbf{A} \in \mathbb{R}^{n\times d}$ and $\mathbf{D} = \mathrm{diag}(d_1, \ldots, d_n) \in \mathbb{R}^{n\times n}$. For any matrix $\mathbf{A}$ and any index set N, let $\mathbf{A}_N \in \mathbb{R}^{n\times d}$ be such that for all $i \in N$, $(\mathbf{A}_N)_i = \mathbf{A}_i$, and all other rows of $\mathbf{A}_N$ are 0. Suppose $\mathbf{A}^\intercal\mathbf{D}\mathbf{A} = \sum_{i=1}^T \mathbf{E}^i + \mathbf{N}$, where $\mathbf{E}^i = \mathbf{A}_{E_i}^\intercal\mathbf{D}_{E_i}\mathbf{A}_{E_i} \in \mathbb{R}^{d\times d}$, E_i is an index set with size $\mathcal{O}(d)$ (that is, at each outer iteration in Algorithm 2 step 3 below, we deterministically select at most $|E_i| = \mathcal{O}(d)$ rows). Let $E = \cup_{i=1}^T E_i$, $N = \{1, \ldots, n\} \backslash E$, $\mathbf{N} = \mathbf{A}_N^\intercal\mathbf{D}_N\mathbf{A}_N \in \mathbb{R}^{d\times d}$.*

Assume $\mathbf{D}_N^{1/2}\mathbf{A}_N$ has the following **relaxed restricted isometry property (RIP) with parameter** *ρ. That is, with probability $1 - 1/n$ over uniformly random sampling matrices $\mathbf{S}$ with $t = \mathcal{O}(d\|\mathbf{A}^\intercal\mathbf{D}\mathbf{A}\|/\epsilon^2)$ rows each scaled by*

$\sqrt{n/t}$, *we have* $\forall \mathbf{x} \notin$ *kernel*$(\mathbf{D}_N^{1/2}\mathbf{A}_N)$, $\|\mathbf{S}\mathbf{D}_N^{1/2}\mathbf{A}_N\mathbf{x}\| = (1 \pm \epsilon)\rho\|\mathbf{x}\|$.

Also assume that for some constant $c > 1$: $c\|\mathbf{A}_N^\intercal|\mathbf{D}_N|\mathbf{A}_N\| \leq \|\sum_i \mathbf{E}^i\|$. *Then the sketch can be expressed as* $\sum_i \mathbf{E}^i + \mathbf{A}_N^\intercal\mathbf{D}_N^{1/2}\mathbf{S}^\intercal\mathbf{S}\mathbf{D}_N^{1/2}\mathbf{A}_N$ *and, with probability* $1 - \mathcal{O}(1/d)$, *we have* $\|\sum_i \mathbf{E}^i + \mathbf{A}_N^\intercal\mathbf{D}_N^{1/2}\mathbf{S}^\intercal\mathbf{S}\mathbf{D}_N^{1/2}\mathbf{A}_N - \mathbf{A}^\intercal\mathbf{D}\mathbf{A}\| \leq \epsilon$.

Remark 2. *The takeaway from Theorem 3 is that the total sample complexity of such a sampling scheme, i.e., Algorithm 2, is* $\mathcal{O}(Td + d\|\mathbf{A}^\intercal\mathbf{D}\mathbf{A}\|/\epsilon^2) = \mathcal{O}(d\|\mathbf{A}^\intercal\mathbf{D}\mathbf{A}\|)$ *for constant* ϵ *and* T. *On the other hand, the vanilla leverage score sampling scheme requires* $\Omega(d\gamma \log d)$ *rows. Notice that often* $\gamma \gg \|\mathbf{A}^\intercal\mathbf{D}\mathbf{A}\|$ *because* γ *involves* $\|\mathbf{A}^\intercal|\mathbf{D}|\mathbf{A}\|$. *Although* T *is a tunable parameter, we found in the experiments that* $T = 1$ *performs well.*

Algorithm 2 Hybrid Randomized-Deterministic Sampling (LS-Det)

Input: $\mathbf{D}^{1/2}\mathbf{A} \in \mathbb{C}^{n\times d}$, iteration number T, threshold m, precision ϵ, number k of rows left
Set $k = n$
for $t \in [T]$ **do**
  Calculate the leverage scores $\{\ell_1, \ldots, \ell_k\}$ of $\mathbf{D}^{1/2}\mathbf{A}$
  for $i \in [k]$ **do**
    if $\ell_i \geq m$ **then**
      Select row i, set $\mathbf{D}^{1/2}\mathbf{A}$ to be the set of remaining rows, set $k = k - 1$
Sample $h = \mathcal{O}(d/\epsilon^2)$ rows from the remaining rows using either their leverage score distribution, or uniformly at random and scaled by $\sqrt{\frac{n}{h}}$
Output: The set of sampled and rescaled rows

Which Matrix to Sketch? We give a general rule of thumb that guides which matrix we should sample to get a better sample complexity. Recall that the Hessian matrix we try to sketch is of the form $\mathbf{A}^\intercal\mathbf{D}\mathbf{A}$ where $\mathbf{D}$ is diagonal. There are two natural candidates:

$$\frac{s(\mathbf{A}_i) + s((\mathbf{D}\mathbf{A})_i)}{\sum_i (s(\mathbf{A}_i) + s((\mathbf{D}\mathbf{A})_i))} \quad (2) \qquad \frac{s((\mathbf{D}^{1/2}\mathbf{A})_i)}{\sum_i s((\mathbf{D}^{1/2}\mathbf{A})_i)} \quad (3)$$

for a score function $s : \mathbb{R}^d \to \mathbb{R}$ (e.g., leverage scores or row norms). It turns out that sampling according to (2) can lead to an arbitrarily worse upper bound than sampling using (3).

Theorem 4. *Let* $\mathbf{A} \in \mathbb{R}^{n\times d}$. *Let* $\mathbf{D} \in \mathbb{R}^{n\times n}$ *be diagonal. Let* $\mathbf{T}$ *be an* ϵ ℓ_2*-subspace embedding for span*$(\mathbf{A}, \mathbf{D}\mathbf{A})$ *and* $\mathbf{S}$ *be an* ϵ ℓ_2*-subspace embedding for span*$(\mathbf{D}^{1/2}\mathbf{A}, (\mathbf{D}^{1/2})^*\mathbf{A})$. *Sampling by* (2) *can give an arbitrarily worse upper bound than sampling by* (3).

2.2 APPLICATION TO OPTIMIZATION ALGORITHMS

As mentioned previously, to accelerate convergence of second-order methods with an inexact Hessian, one needs to construct the sub-sampled matrix such that $\mathbf{H}(\mathbf{x}) \approx \nabla^2 F(\mathbf{x})$. In randomized sub-sampling of the Hessian matrix, we select the i-th term in $\sum_{i=1}^n \nabla^2 f_i(\mathbf{x})$ with probability p_i, restricting $\sum_{i\in[n]} p_i = 1$. Let $\mathcal{S}$ denote the sample collection and define $\mathbf{H}(\mathbf{x}) \triangleq \frac{1}{n|\mathcal{S}|}\sum_{i\in\mathcal{S}} \frac{1}{p_i}\nabla^2 f_i(\mathbf{x}) + \nabla^2 r(\mathbf{x})$. Uniform oblivious sampling is done with $p_i = 1/n$, which often results in a poor approximation unless $|\mathcal{S}| \gg 1$. Leverage score sampling is in some sense an optimal data-aware sampling scheme where each p_i is proportional to the leverage score ℓ_i (see Algorithm 1).

One condition on the quality of approximation $\mathbf{H}(\mathbf{x}) \approx \nabla^2 F(\mathbf{x})$ is typically taken to be

$$\|\mathbf{H}(\mathbf{x}) - \nabla^2 F(\mathbf{x})\| \leq \epsilon, \quad \text{for some } 0 < \epsilon \leq 1, \qquad (4)$$

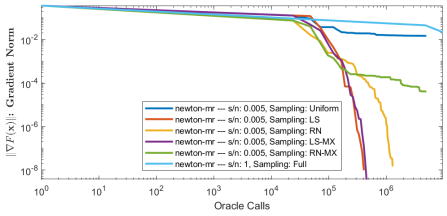
which has been considered both in the contexts of convex and non-convex Newton-type optimization methods [Roosta and Mahoney, 2019, Bollapragada et al., 2019, Xu et al., 2019, Yao et al., 2018, Liu and Roosta, 2019]. For convex settings where $\nabla^2 F(x) \succeq \mathbf{0}$, a stronger condition can be considered as

$$(1 - \epsilon)\nabla^2 F(x) \preceq \mathbf{H}(\mathbf{x}) \preceq (1 + \epsilon)\nabla^2 F(x), \qquad (5)$$

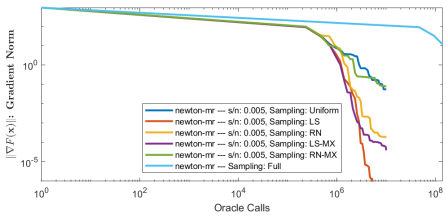
which, in the context of sub-sampled Newton's method, leads to a faster convergence rate than (4) Roosta and Mahoney [2019], Xu et al. [2016], Liu et al. [2017]. However, in all prior work, (5) has only been considered in the restricted case where each f_i is convex. Here, using the result of Section 2.1, we show that (5) can also be guaranteed in a more general case where the f_i's in (1) are allowed to be non-convex. We demonstrate the theoretical advantages of complex leverage score sampling in Algorithms 1 and 2 as a way to guarantee (4) and (5) in convex and non-convex settings, respectively. For the convex case, we consider sub-sampled Newton-CG [Roosta and Mahoney, 2019, Xu et al., 2016]. For non-convex settings, we have chosen two examples of Newton-type methods: the classical trust-region [Conn et al., 2000] and the more recently introduced Newton-MR method [Roosta et al., 2018]. We emphasize that the choice of these non-convex algorithms was, to an extent, arbitrary and we could instead have picked any Newton-type method whose convergence has been previously studied under Hessian approximation models, e.g., adaptive cubic regularization [Yao et al., 2018, Tripuraneni et al., 2018]. The details of these optimization methods and theoretical convergence results are deferred to the supplementary.

We verify the results of Section 2.1 by evaluating the empirical performance of the non-uniform sampling strategies

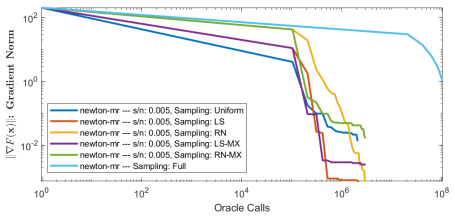
proposed in the context of Newton-CG, Newton-MR and trust-region, see details in the appendix.



(a) $\|\nabla F(x_k)\|$ vs. Oracle calls (`Drive Diagnostics`)



(b) $\|\nabla F(x_k)\|$ vs. Oracle calls (`Cover Type`)



(c) $\|\nabla F(x_k)\|$ vs. Oracle calls (`UJIIndoorLoc`)

Figure 1: Comparison of Newton-MR with various sampling schemes.

Sub-sampling Schemes. We focus on several sub-sampling strategies (all are done with replacement). *Uniform*: For this we have $p_i = 1/n,\ i = 1, \ldots, n$. *Leverage Scores (LS)*: Complex leverage score sampling by considering the leverage scores of $\mathbf{D}^{1/2}\mathbf{A}$ as in Algorithm 1. *Row Norms (RN)*: Row-norm sampling of $\mathbf{D}^{1/2}\mathbf{A}$ using (3) where $s((\mathbf{D}^{1/2}\mathbf{A})_i) = |f_i''(\langle \mathbf{a}_i, \mathbf{x}\rangle)|\,\|\mathbf{a}_i\|_2^2$. *Mixed Leverage Scores (LS-MX)*: A mixed leverage score sampling strategy arising from a non-symmetric viewpoint of the product $\mathbf{A}^\intercal(\mathbf{DA})$ using (2) with $s(\mathbf{A}_i) = \ell_i(\mathbf{A})$ and $s((\mathbf{DA})_i) = \ell_i(\mathbf{DA})$. *Mixed Norm Mixture (RN-MX)*: A mixed row-norm sampling strategy with the same non-symmetric viewpoint as in (2) with $s(\mathbf{A}_i) = \|(\mathbf{A})_i\|$ and $s((\mathbf{DA})_i) = \|(\mathbf{DA})_i\|$. *Hybrid Randomized-Deterministic (LS-Det)*: Sampling using Algorithm 2. *Full*: In this case, the exact Hessian is used.

Model Problems and Datasets. We consider the task of binary classification using the non-linear least squares formulation of (1). Numerical experiments in this section are done using `covertype`, `Drive Diagnostics`, and `UJIIndoorLoc` from the UC Irvine ML Repository Dua and Graff [2017].

Performance Evaluation. For Newton-MR, the convergence is measured by the norm of the gradient, and hence we evaluate it using various sampling schemes by plotting $\|\nabla F(\mathbf{x}_k)\|$ vs. the total number of oracle calls. For Newton-CG and trust-region, which guarantee descent in objective function, we plot $F(\mathbf{x}_k)$ vs. the total number of oracle calls. We deliberately choose not to use "wall-clock" time since it heavily depends on the implementation details and system specifications.

Comparison Among Various Sketching Techniques. We present empirical evaluations of Uniform, LS, RN, LS-MX, RN-MX and Full sampling in the context of Newton-MR in Figure 1, and evaluation of all sampling schemes in Newton-CG, Newton-MR, Trust-region, and hybrid sampling on `covertype` in Figure 2. For all algorithms, LS and RN sampling amounts to a more efficient algorithm than that with LS-MX and RN-MX variants respectively, and at times this difference is more pronounced than other times, as predicted in Theorem 4. Meanwhile, LS and LS-MX often outperform RN and RN-MX, as proven in Theorem 1 and Theorem 2.

Evaluation of Hybrid Sketching Techniques. To verify the result of Algorithm 2, we evaluate the performance of the trust-region algorithm by varying the terms involved in $\mathbf{E}$, where we call the rows with large leverage scores heavy, and denote the matrix formed by the heavy rows by $\mathbf{E}$. The matrix formed by the remaining rows is denoted by $\mathbf{N}$, see Theorem 3 for details. We do this for a simple splitting of $\mathbf{H} = \mathbf{E} + \mathbf{N}$. We fix the overall sample size and change the fraction of samples that are deterministically picked in $\mathbf{E}$. The results are depicted in Figure 2. The value in brackets after LS-Det is the fraction of samples that are included in $\mathbf{E}$, i.e., deterministic samples. "LS-Det (0)" and "LS-Det (1)" correspond to $\mathbf{E} = \mathbf{0}$ and $\mathbf{N} = \mathbf{0}$, respectively. The latter strategy has been used in low rank matrix approximations [McCurdy, 2018]. As can be seen, the hybrid sampling approach is always competitive with, and at times strictly better than, LS-Det (0). As expected, LS-Det (1), which amounts to entirely deterministic samples, consistently performs worse. This can be easily attributed to the high bias of such a deterministic estimator.

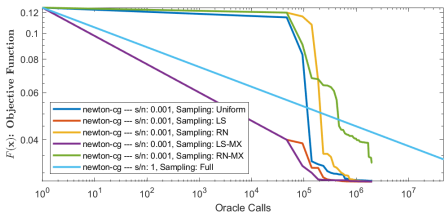
3 SKETCH-AND-SOLVE PARADIGM FOR COMPLEX REGRESSION

3.1 THEORETICAL RESULTS

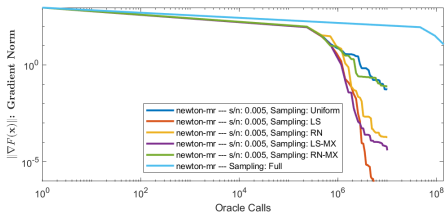
Recall the ℓ_p-regression problem:

$$\min_{\mathbf{x}} \|\mathbf{Ax} - \mathbf{b}\|_p \tag{6}$$

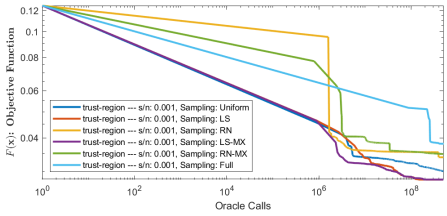
Here we consider the complex version: $\mathbf{A} \in \mathbb{C}^{n\times d}, \mathbf{b} \in \mathbb{C}^n, \mathbf{x} \in \mathbb{C}^d$.



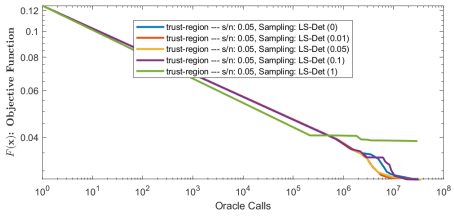
(a) $F(x_k)$ vs. Oracle calls



(b) $\|\nabla F(x_k)\|$ vs. Oracle calls



(c) $F(x_k)$ vs. Oracle calls



(d) $F(x_k)$ vs. Oracle calls

Figure 2: Comparison of (a) Newton-CG, (b) Newton-MR, and (c) Trust-region using various sampling schemes. (d) Performance of the hybrid sampling scheme.

The inner product over the complex field can be embedded into a higher-dimensional real vector space. It suffices to consider the scalar case.

Lemma 1. *Let $x, y \in \mathbb{C}$ where $x = a + bi, y = c + di$, let $\phi : \mathbb{C} \to \mathbb{R}^2$ via $\phi(x) = \phi(a + bi) = [a \;\; b]$. Let $\sigma : \mathbb{C} \to \mathbb{R}^{2\times 2}$ via: $\sigma(y) = \sigma(c + di) = \begin{bmatrix} c & d \\ -d & c \end{bmatrix}$. Then ϕ and σ are bijections (between their domains and images), and we have $\phi(\bar{y}x)^\intercal = \sigma(y)\phi(x)^\intercal$*

Proof of Lemma 1. It is clear that ϕ, σ are bijections between their domains and images. $\sigma(y)\phi(x)^\intercal = \begin{bmatrix} c & d \\ -d & c \end{bmatrix}\begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} ac + bd \\ cb - ad \end{bmatrix} = \phi(\bar{y}x)^\intercal$. □

We apply σ to each entry in $\mathbf{A}$ and concatenate in the natural way. Abusing notation, we then write: $\mathbf{A}' = \sigma(\mathbf{A}) \in \mathbb{R}^{2n\times 2d}$. Similarly, we write $\mathbf{x}' = \phi(\mathbf{x}) \in \mathbb{R}^{2d}$, $\mathbf{b}' = \phi(\mathbf{b}) \in \mathbb{R}^{2d}$.

Fact 1. *For $1 \le p < q \le \infty$ and $\mathbf{x} \in \mathbb{R}^d$, we have $\|\mathbf{x}\|_q \le \|\mathbf{x}\|_p \le d^{1/p-1/q}\|\mathbf{x}\|_q$.*

Fact 2. *An ℓ_p Auerbach basis is well-conditioned and always exists.*

Definition 4. *Let $\mathbf{x} \in \mathbb{R}^{2d}$ for some $d \in \mathbb{N}$. Define $|||\mathbf{x}|||_{p,2} = \left(\sum_{i=0}^{d-1}\left(\mathbf{x}_{2i+1}^2 + \mathbf{x}_{2i+2}^2\right)^{p/2}\right)^{1/p}$. Note that letting $\mathbf{x} \in \mathbb{C}^d$, we have $\|\mathbf{x}\|_p = |||\phi(\mathbf{x})|||_{p,2}$.*

By Definition 4, we can "lift" the original ℓ_p-regression to $\mathbb{R}^{2d}$ and solve $\min_{\phi(\mathbf{x})\in\mathbb{R}^{2d}} |||\sigma(\mathbf{A})\phi(\mathbf{x}) - \phi(\mathbf{b})|||_{p,2}$ instead of (6). This equivalence allows us to consider sketching techniques on real matrices with proper modification. See Algorithm 3 for details. In turn, such an embedding gives an arbitrarily good approximation with high probability, as shown in Theorem 5.

Theorem 5. *Let $\mathbf{A} \in \mathbb{C}^{n\times d}, \mathbf{b} \in \mathbb{C}^n$. Then Algorithm 3 with input $\mathbf{A}' := \sigma(\mathbf{A})$ and $\mathbf{b}' := \phi(\mathbf{b})$ returns a regression instance whose optimizer is an ϵ approximation to (6), with probability at least 0.98. The total time complexity for $p \in [1, \infty)$ is $\mathcal{O}(\mathbf{nnz}(\mathbf{A}) + \mathbf{poly}(d/\epsilon))$; for $p = \infty$ it is $\mathcal{O}(2^{\tilde{\mathcal{O}}(1/\epsilon^2)}\mathbf{nnz}(\mathbf{A}))$. The returned instance can then be optimized by any ℓ_p-regression solver.*

Proof. For simplicity, we let $\mathbf{A} \in \mathbb{R}^{2n\times 2d}$ to avoid repeatedly writing the prime symbol in $\mathbf{A}'$.

Let $\mathbf{y} \in \mathbb{R}^{2n}$ be arbitrary. Let $P, \mathcal{P}, \mathcal{P}_l, \mathcal{P}_h$ be as defined in Algorithm 3. We say a pair $P = (a, b)$ is heavy if $P \in \mathcal{P}_h$, and light otherwise.

As an overview, when $p \in [1, \infty)$, for the heavy pairs, we use a large Gaussian matrix and apply Dvoretsky's theorem to show the $|||\cdot|||_{p,2}$ norm is preserved. For the light pairs, we use a single Gaussian vector and use Bernstein's concentration. This is intuitive since the heavy pairs represent the important directions in $\mathbf{A}$, and hence we need more Gaussian vectors to preserve their norms more accurately; but the light pairs are less important and so the variance of the light pairs can be averaged across multiple coordinates. Hence, using one Gaussian vector suffices for each light pair. For $p = \infty$, we need to preserve the ℓ_2 norm of every pair, and so in this case we apply Dvoretsky's theorem to sketch every single pair in $\mathcal{P}$.

We split the analysis into two cases: $p \in [1, \infty)$ and $p = \infty$. In the main text we only present $p \in [1, \infty)$ and defer the

Algorithm 3 Fast Algorithm for Complex ℓ_p-regression

Input: $\mathbf{A}' \in \mathbb{R}^{2n\times 2d}, \mathbf{b}' \in \mathbb{R}^{2d}$, precision $\epsilon > 0$, $p \in [1,\infty]$
Set $t \geq C\frac{\log(1/\epsilon)\mathbf{poly}(d)}{\epsilon^2}$, $\gamma = d^{-1/q-1}$, $\sigma = \frac{\sqrt{\pi}}{2^{p/2}\Gamma((p+1)/2)}$, $\mathcal{P} = \{P = (2i+1, 2i+2) \in \mathbb{R}^2, \forall i \in \{0,\ldots,d-1\}\}$
if $p \neq \infty$ **then**
  Compute the ℓ_p leverage score $\{\ell([\mathbf{A}'_1 \quad \mathbf{b}'_1]),\ldots,\ell([\mathbf{A}'_n \quad \mathbf{b}'_n])\}$ for each row of $[\mathbf{A}' \quad \mathbf{b}']$
  Let $\mathcal{P}_h$ be the collection of $P = (a,b) \in \mathcal{P}$ such that $\ell([\mathbf{A}'_a \quad \mathbf{b}'_a]) \geq \gamma$ or $\ell([\mathbf{A}'_b \quad \mathbf{b}'_b]) \geq \gamma$. Denote the collection of all other pairs $\mathcal{P}_l$
  Rearrange $[\mathbf{A}' \quad \mathbf{b}']$ such that the rows in $\mathcal{P}_h$ are on top
  for Each $P_i \in \mathcal{P}_h$ **do**
    Sample a Gaussian random matrix $\mathbf{G}_{P_i} \in \mathbb{R}^{t\times 2}$, where each entry follows $\mathcal{N}(0,\sigma^2)$
  for Each $Q_i \in \mathcal{P}_l$ **do**
    Sample Gaussian random matrix $\mathbf{G}_{Q_i} \in \mathbb{R}^{1\times 2}$, where each entry follows $\mathcal{N}(0,\sigma^2)$
  Define a block diagonal matrix $\mathbf{G} = \mathrm{diag}(\mathbf{G}_{P_1},\ldots,\mathbf{G}_{P_{|\mathcal{P}_h|}},\mathbf{G}_{Q_1},\ldots,\mathbf{G}_{Q_{|\mathcal{P}_l|}})$
  Output: $\|\mathbf{G}\mathbf{A}'\mathbf{y} - \mathbf{G}\mathbf{b}'\|_p$
else
  for Each $P_i \in \mathcal{P}$ **do**
    Sample Gaussian matrix $\mathbf{G}_{P_i} \in \mathbb{R}^{s\times 2}$ with entries from $\mathcal{N}(0,\frac{\pi}{2})$, where $s = \mathcal{O}\left(\frac{\log(1/\epsilon)}{\epsilon^2}\right)$
    Let $\mathbf{R}_{P_i} \in \mathbb{R}^{2^s\times s}$ where each row of $\mathbf{R}_{P_i}$ is a vector in $\{-1,+1\}^s$
  Let $\mathbf{R} = \mathrm{diag}(\mathbf{R}_1,\ldots,\mathbf{R}_{|\mathcal{P}|})$, $\mathbf{G} = \mathrm{diag}(\mathbf{G}_1,\ldots,\mathbf{G}_{|\mathcal{P}|})$
  Output: $\|\mathbf{R}\mathbf{G}\mathbf{A}'\mathbf{y} - \mathbf{R}\mathbf{G}\mathbf{b}'\|_\infty$

other case to the appendix.

Case 1: $p \in [1,\infty)$. *For light rows*:

Let $\mathbf{U}$ be an Auerbach basis of $\mathbf{A}$ and $\mathbf{y} = \mathbf{U}\mathbf{x}$, where $\mathbf{x} \in \mathbb{R}^d$ is arbitrary. Then for any row index $i \in [n]$:

$$|\mathbf{y}_i| = |\mathbf{U}_i\mathbf{x}| \leq \|\mathbf{U}_i\|_p\|\mathbf{x}\|_q \leq d^{1/q}\|\mathbf{U}_i\|_p\|\mathbf{y}\|_p$$
$$\Longrightarrow \frac{d^{-1/q}|\mathbf{y}_i|}{\|\mathbf{y}\|_p} \leq \|\mathbf{U}_i\|_p$$

where the first step is because the Auerbach basis satisfies $\|\mathbf{U}\mathbf{x}\|_p = \|\mathbf{y}\|_p \geq d^{-1/q}\|\mathbf{x}\|_q$. This implies that if $|\mathbf{y}_i| \geq \gamma d^{1/q}\|\mathbf{y}\|_p$, then $\|\mathbf{U}_i\|_p \geq \gamma$. Hence, by definition of γ, if $\|\mathbf{U}_i\|_p \leq d^{-1/q-1}$, then $|\mathbf{y}_i| \leq d^{-1}\|\mathbf{y}\|_p$.

For any light pair $P = (a,b)$, we sample two i.i.d. Gaussians $\mathbf{g}_P = (\mathbf{g}_a, \mathbf{g}_b)$ from $\mathcal{N}(0,\sigma^2)$ where $\sigma = \frac{\sqrt{\pi}}{2^{p/2}\Gamma((p+1)/2)}$. Since $\mathbf{g}_a\mathbf{y}_a + \mathbf{g}_b\mathbf{y}_b \sim \mathcal{N}(0, \|\mathbf{y}_P\|_2^2\sigma^2)$, we have $\mathbb{E}[|\mathbf{g}_a\mathbf{y}_a + \mathbf{g}_b\mathbf{y}_b|^p] = \|\mathbf{y}_P\|_2^p$.

Let $\mathcal{P}_l$ be the set of all light pairs. We have, $\mathbb{E}[\sum_{P\in\mathcal{P}_l}|\langle\mathbf{g}_P,\mathbf{y}_P\rangle|^p] = \sum_{P\in\mathcal{P}_l}\|\mathbf{y}_P\|_2^P$.

Let random variable $Z_P = |\langle\mathbf{g}_P,\mathbf{y}_P\rangle|$. This is $\sigma^2\|\mathbf{y}_P\|_2^2$-sub-Gaussian (the parameter here can be improved by subtracting $\mu^2_{Z_P}$).

Define event $\mathcal{A}$ to be : $\max_{P\in\mathcal{P}_l} Z_P^p \geq \mathcal{O}((\log(|\mathcal{P}_l|) + \mathbf{poly}(d))^p) \triangleq t$. Since the Z_P variables are sub-Gaussian and $p \geq 1$, we have that $(\cdot)^p$ is monotonically increasing, so we can use the standard sub-Gaussian bound to get

$$P(\mathcal{A}) < \exp\left(-\mathbf{poly}(d)\right). \tag{7}$$

Note that p is a constant, justifying the above derivation.

Also note that $\mathrm{Var}(Z_P) = \mathcal{O}(\|\mathbf{y}_P\|_2^{2p})$. Condition on $\neg\mathcal{A}$. By Bernstein's inequality, we have:

$$\begin{aligned}
&P\left(\left|\sum_{P\in\mathcal{P}_l} Z_P^p - \mathbb{E}[|\sum_{P\in\mathcal{P}_l} Z_P^p|]\right| > \epsilon\mathbb{E}[|\sum_{P\in\mathcal{P}} Z_P^p|]\right)\\
=&P\left(\left|\sum_{P\in\mathcal{P}_l} Z_P^p - \sum_{P\in\mathcal{P}_l}\|\mathbf{y}_P\|_2^p\right| > \epsilon\sum_{P\in\mathcal{P}}\|\mathbf{y}_P\|_2^p\right)\\
\leq&\exp\left(-\frac{\epsilon^2(\sum_{P\in\mathcal{P}}\|\mathbf{y}_P\|_2^p)^2}{\sum_{P\in\mathcal{P}_l}\|\mathbf{y}_P\|_2^{2p} + \epsilon t\sum_{P\in\mathcal{P}_l}\|\mathbf{y}_P\|_2^p}\right)\\
\leq&\exp\left(-\mathcal{O}\left(\frac{\epsilon^2(\sum_{P\in\mathcal{P}}\|\mathbf{y}_P\|_2^p)^2}{\sum_{P\in\mathcal{P}_l}\|\mathbf{y}_P\|_2^{2p}}\right)\right)\\
\leq&\exp\left(-\mathcal{O}(\epsilon^2/\gamma)\right) = \exp\left(\mathcal{O}(-\epsilon^2 d\log d)\right)
\end{aligned} \tag{8}$$

where the last step follows from the definition of light pairs. Using that if for all $P = (a,b) \in \mathcal{P}_l$, $|\mathbf{y}_a|, |\mathbf{y}_b| < \mathcal{O}(d^{-1})\|\mathbf{y}\|_p$, then $\|\mathbf{y}_P\|_2^p < \mathcal{O}(d^{-1})\|\mathbf{y}\|_p^p$, we have

$$\sum_{P\in\mathcal{P}_l}\|\mathbf{y}_P\|_2^{2p} \leq \frac{1}{\mathcal{O}(d^{-1})}\mathcal{O}(d^{-2})\|\mathbf{y}\|_p^{2p} = \mathcal{O}(d^{-1})\|\mathbf{y}\|_p^{2p}.$$

Since $\sum_{P\in\mathcal{P}}\|\mathbf{y}_P\|_2^p \leq \mathcal{O}(\|\mathbf{y}\|_p^p)$, we will have $\frac{(\sum_{P\in\mathcal{P}}\|\mathbf{y}_P\|_2^p)^2}{\sum_{P\in\mathcal{P}_l}\|\mathbf{y}_P\|_2^{2p}} = \mathcal{O}(d)$.

Net argument. In the above derivation, we fix a vector $\mathbf{y} \in \mathbb{R}^{2n}$. Hence for the above argument to hold for all pairs, a naïve argument will not work since there are an infinite number of pairs. However, note that each pair lives in a two-dimensional subspace. Hence, we can take a finer union bound over $\mathcal{O}((1+\epsilon)^2/\epsilon^2)$ items in a two-dimensional ℓ_2 space using a net argument. This argument is standard, see, e.g., [Woodruff et al., 2014, Chapter 2].

Using the net argument, (7) and (8) holds with probability at least $1 - \mathcal{O}(e^{-d})$ for all $\mathbf{y} \in \mathbb{R}^{2n}$ simultaneously. In particular, with probability at least 0.99:

$$\sum_{P\in\mathcal{P}_l}|\langle\mathbf{g}_P,\mathbf{y}_P\rangle|^p - \sum_{P\in\mathcal{P}_l}\|\mathbf{y}_P\|_2^p \in (\pm\epsilon)\sum_{P\in\mathcal{P}}\|\mathbf{y}_P\|_2^p. \tag{9}$$

For heavy rows:

Since the ℓ_p leverage scores sum to d, there can be at most $d/\gamma = \mathbf{poly}(d)$ heavy rows. For each pair $P \in \mathcal{P}_h$, we construct a Gaussian matrix $\mathbf{G}_P \in \mathbb{R}^{s\times 2}$, where $s = \mathbf{poly}(d/\epsilon)$. Applying Dvoretsky's theorem for ℓ_p (Paouris et al. [2017] Theorem 1.2), with probability at least 0.99, for all 2-dimensional vectors $\mathbf{y}_P$, $\|\mathbf{G}_P\mathbf{y}_P\|_p = (1\pm\epsilon)\|\mathbf{y}_P\|_2$. Hence,

$$\sum_{P\in\mathcal{P}_h} \|\mathbf{G}_P\mathbf{y}_P\|_p^p = (1\pm\Theta(\epsilon)) \sum_{P\in\mathcal{P}_h} \|\mathbf{y}_P\|_2^p. \tag{10}$$

Combining (9) and (10), we have with probability at least 0.98, for all $\mathbf{y}\in\mathbb{R}^{2n}$:

$$\|\mathbf{G}\mathbf{y}\|_p^p = (1\pm\Theta(\epsilon)) \sum_{P\in\mathcal{P}} \|\mathbf{y}_P\|_2^p = (1\pm\Theta(\epsilon))\,|||\mathbf{y}|||_{p,2}^p$$

Letting $\mathbf{y} = \mathbf{U}\mathbf{x} - \mathbf{b}$ and taking the $1/p$-th root, we obtain the final claim.

Case 2: $p = \infty$**.** In this case, we first construct a sketch $\mathbf{G}$ to embed every pair into ℓ_1. That is, for all $P\in\mathcal{P}$, construct a Gaussian matrix $\mathbf{G}_p \in \mathbb{R}^{\mathcal{O}(\frac{\log(1/\epsilon)}{\epsilon^2})\times 2}$. By Dvoretsky's theorem, with probability at least 0.99, for all $\mathbf{y}_P\in\mathbb{R}^2$, we have $\|\mathbf{G}_P\mathbf{y}_P\|_1 = (1\pm\epsilon)\|\mathbf{y}_P\|_2$. Hence

For all $\mathbf{y}\in\mathbb{R}^{2n}$ and any $P\in\mathcal{P}$:

$$\max_{P\in\mathcal{P}} \|\mathbf{G}_P\mathbf{y}_P\|_1 = \max_{P\in\mathcal{P}} (1\pm\epsilon)\|\mathbf{y}_P\|_2 = |||\mathbf{y}|||_{\infty,2} \tag{11}$$

However, we do not want to optimize the left hand side directly.

Recall that by construction $\mathbf{G} = \mathrm{diag}(\mathbf{G}_1,\ldots,\mathbf{G}_{|\mathcal{P}|})$ is a block diagonal matrix.

Construct $\mathbf{R}$ as in Algorithm 3. By Indyk [2001], for all $P\in\mathcal{P}$, $\mathbf{R}_P$ is an isometric embedding $\ell_1 \hookrightarrow \ell_\infty$, i.e., for all $\mathbf{y}_P\in\mathbb{R}^2$: $\|\mathbf{R}_P\mathbf{G}_P\mathbf{y}_P\|_\infty = \|\mathbf{G}_P\mathbf{y}_P\|_1$.

Combining this with (11), we get that with probability at least 0.99 for all $\mathbf{y}\in\mathbb{R}^{2n}$:

$$\|\mathbf{R}\mathbf{G}\mathbf{y}\|_\infty = (1\pm\Theta(\epsilon))\,|||\mathbf{y}|||_{\infty,2}\,.$$

Letting $\mathbf{y} = \mathbf{U}\mathbf{x} - \mathbf{b}$, we obtain the final claim.

Running time. For $p\in[1,\infty)$, calculating a well-conditioned basis takes $\mathcal{O}(\mathbf{nnz}(\mathbf{A}) + \mathbf{poly}(d/\epsilon))$ time. Since $\mathbf{G}$ is a block diagonal matrix and $\mathbf{A}$ is sparse, computing $\mathbf{G}\mathbf{A}$ takes $\mathcal{O}(\mathbf{nnz}(\mathbf{A}) + \mathbf{poly}(d/\epsilon))$ time. Calculating $\mathbf{G}\mathbf{b}$ takes $n + \mathbf{poly}(d/\epsilon)$ time. Minimizing $\|\mathbf{G}\mathbf{A}\mathbf{x} - \mathbf{G}\mathbf{b}\|$ up to a $(1+\epsilon)$ factor takes $\mathcal{O}(\mathbf{nnz}(\mathbf{A}) + \mathbf{poly}(d/\epsilon))$ time. Using the fact that $n < \mathbf{nnz}(\mathbf{A})$, the total running time is $\mathcal{O}(\mathbf{nnz}(\mathbf{A}) + \mathbf{poly}(d/\epsilon))$.

For the case of $p=\infty$, note that $\mathbf{R}$ is also a block diagonal matrix, so $\mathbf{R}\mathbf{G}$ can be computed by multiplying the corresponding blocks, which amounts to $\mathcal{O}(2^{\tilde{\mathcal{O}}(1/\epsilon^2)} n \frac{\log(1/\epsilon)}{\epsilon^2})$ time. $\mathbf{A}$ is sparse and $\mathbf{R}\mathbf{G}$ is a block matrix so computing $\mathbf{R}\mathbf{G}\mathbf{A}$ takes another $\mathcal{O}(2^{\tilde{\mathcal{O}}(1/\epsilon^2)}\mathbf{nnz}(\mathbf{A}))$ time. Computing $\mathbf{R}\mathbf{G}\mathbf{b}$ takes $\mathcal{O}(2^{\tilde{\mathcal{O}}(1/\epsilon^2)} n \frac{\log(1/\epsilon)}{\epsilon^2})$ time. Since $n < \mathbf{nnz}(\mathbf{A})$, in total these take $\mathcal{O}(2^{\tilde{\mathcal{O}}(1/\epsilon^2)}\mathbf{nnz}(\mathbf{A}))$ time. This concludes the proof. □

Remark 3. *Definition 4 shows for sketching complex vectors in the ℓ_p norm, all one needs is an embedding $\ell_2 \hookrightarrow \ell_p$. In particular, for complex ℓ_2-regression, the identity map is such an embedding with no distortion. Hence, complex ℓ_2-regression can be sketched exactly as for real-valued ℓ_2-regression, while for other complex ℓ_p-regression the transformation is non-trivial.*

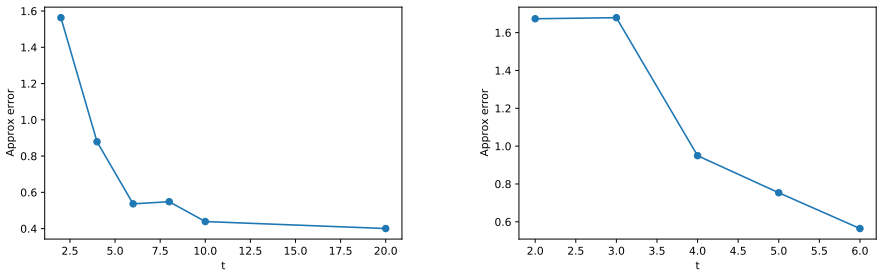
3.2 NUMERICAL EVALUATION

We evaluate the performance of our proposed embedding for ℓ_1 and ℓ_∞ regression on synthetic data. With $\mathbf{A}\in\mathbb{C}^{100\times 50}, \mathbf{b}\in\mathbb{C}^{100}$, we solve $\min_{\mathbf{x}\in\mathbb{C}^{50}}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_1$ or $\min_{\mathbf{x}\in\mathbb{C}^{50}}\|\mathbf{A}\mathbf{x}-\mathbf{b}\|_\infty$. Each entry of $\mathbf{A}$ and $\mathbf{b}$ is sampled from a standard normal distribution (the real and imaginary coefficients are sampled according to this distribution independently). Instead of picking the heavy ($\mathcal{P}_h$) and light ($\mathcal{P}_l$) pairs , we construct a $t\times 2$ (or $s\times 2$ if $p=\infty$) Gaussian matrix for each pair (that is, we treat all pairs as heavy), as it turns out in the experiments that very small t or s is sufficient. For ℓ_1 complex regression, we test with $t = 2,4,6,8,10,20$, and the result is shown in Figure 3(a). For ℓ_∞ complex regression, we test with $s = 2,3,4,5,6$ as shown in Figure 3(b). In both figures, the x-axis represents our choice of t or s, and the y-axis is the approximation error $\|\hat{\mathbf{x}}-\mathbf{x}^*\|_2$, where $\hat{\mathbf{x}}$ is the minimizer of our sketched regression problem.

4 SKETCHING VECTOR-MATRIX-VECTOR QUERIES

The sketches in Section 2 can also be used for vector-matrix-vector queries, but they are sub-optimal when there are a lot of cancellations. For example, if we have $\mathbf{A}^\top\mathbf{D}\mathbf{A} = \sum_{i=1}^n d_i\mathbf{a}_i\mathbf{a}_i^\top$ where $\mathbf{a}_1 = \mathbf{a}_2 = \cdots = \mathbf{a}_n$, $d_1 = d_2 = \cdots d_{n/2}$, and $d_{n/2+1} = d_{n/2+2} = \cdots = d_n$, then $\mathbf{A}^\top\mathbf{D}\mathbf{A} = 0$, yet our sampling techniques need their number of rows to scale with $\|\mathbf{A}^\top|\mathbf{D}|\mathbf{A}\|_F$, which can be arbitrarily large. In this section, we give a sketching technique for vector-matrix-vector product queries that scales with $\|\mathbf{A}^\top\mathbf{D}\mathbf{A}\|_F$ instead of $\|\mathbf{A}^\top|\mathbf{D}|\mathbf{A}\|_F$. Therefore, for vector-matrix-vector product queries, this new technique works well, even if the matrices are complex. Such queries are widely used, including standard graph queries and independent set queries [Rashtchian et al., 2020].

In particular, we consider a vector-matrix-vector product query $\mathbf{u}^\top\mathbf{M}\mathbf{v}$, where $\mathbf{M} = \mathbf{A}^\top\mathbf{B} = \sum_{i=1}^n \mathbf{a}_i\mathbf{b}_i^\top$ has a tensor product form, $\mathbf{a}_i,\mathbf{b}_i\in\mathbb{C}^d$, for all $i\in[n]$. One has to



(a) ℓ_1 regression (b) ℓ_∞ regression

Figure 3: Approximation error of sketched ℓ_p regression with complex entries.

either sketch $\mathbf{M}$ or compute $\mathbf{M}$ first. Then the queries $\mathbf{u}$ and $\mathbf{v}$ arrive [Andoni et al., 2016]. In reality, this may be due to the fact that $n \gg d$ and one cannot afford to store $\mathbf{A}$ and $\mathbf{B}$. Our approach is interesting when $\mathbf{M}$ is non-PSD and $\mathbf{A}, \mathbf{B}$ might be complex. This can indeed happen, for example, in a graph Laplacian with negative weights [Chen et al., 2016].

Algorithm 4 Tensor Sketch For Vector-Matrix-Vector Products

1: **Input:** $\{\mathbf{a}_i\}_{i=1}^n, \{\mathbf{b}_i\}_{i=1}^n \subseteq \mathbb{C}^d, \mathbf{u}, \mathbf{v} \in \mathbb{C}^d$
2: Let $\mathbf{S} : \mathbb{C}^d \otimes \mathbb{C}^d \to \mathbb{C}^k$ be a TensorSketch [Pham and Pagh, 2013] with k hash buckets.
3: Compute $\mathbf{q} = \sum_{i=1}^n \mathbf{S}(\mathbf{a}_i \otimes \mathbf{b}_i) \in \mathbb{C}^k$.
4: Compute $\mathbf{p} = \mathbf{S}(\mathbf{u} \otimes \mathbf{v}) \in \mathbb{C}^k$.
5: **Output:** $\langle \mathbf{p}, \mathbf{q} \rangle$

Theorem 6. *With probability at least* 0.99*, for given input vectors* $\mathbf{u}, \mathbf{v} \in \mathbb{C}^d$*,* $\mathbf{A}, \mathbf{B} \in \mathbb{C}^{n \times d}$*, Algorithm 4 returns an answer* z *such that* $\left|z - \mathbf{u}^\top \mathbf{A}^\top \mathbf{B}\mathbf{v}\right| \leq \epsilon$ *in time* $\tilde{\mathcal{O}}\left(\mathbf{nnz}(\mathbf{A}) + \frac{n\|\mathbf{v}\|_2^2 \|\mathbf{u}\|_2^2 \|\mathbf{A}^\top \mathbf{B}\|_F^2}{\epsilon^2}\right)$.

Proof. It is known that TensorSketches are unbiased, that is,

$$\mathbb{E}[\langle \mathbf{S}(\mathbf{A}^\top \mathbf{B}), \mathbf{S}(\mathbf{u} \otimes \mathbf{v}) \rangle] = \mathbf{u}^\top \mathbf{A}^\top \mathbf{B}\mathbf{v},$$

where $\mathbf{S}(\mathbf{A}^\top \mathbf{B}) = \sum_{i=1}^n \mathbf{S}(\mathbf{a}_i, \mathbf{b}_i)$ follows from the linearity of TensorSketch Pham and Pagh [2013]. The variance is bounded by

$$\mathrm{Var}\left(\langle \mathbf{S}(\mathbf{A}^\top \mathbf{B}), \mathbf{S}(\mathbf{u} \otimes \mathbf{v}) \rangle\right) \leq \mathcal{O}\left(\frac{1}{k} \|\mathbf{v}\|_2^2 \|\mathbf{u}\|_2^2 \|\mathbf{A}^\top \mathbf{B}\|_F^2\right),$$

see, e.g., Pham and Pagh [2013]. By Chebyshev's inequality, setting $k = \mathcal{O}\left(\frac{\|\mathbf{v}\|_2^2 \|\mathbf{u}\|_2^2 \|\mathbf{A}^\top \mathbf{B}\|_F^2}{\epsilon^2}\right)$ produces an estimate of $\mathbf{u}^\top \mathbf{A}^\top \mathbf{B}\mathbf{v}$ with an additive error ϵ.

Note that computing the sketches $\mathbf{S}(\mathbf{A}^\top \mathbf{B})$ takes time $\mathbf{nnz}(\mathbf{A}) + \mathbf{nnz}(\mathbf{B}) + nk \log k$, and computing the inner product $\langle \mathbf{S}(\mathbf{A}^\top \mathbf{B}), \mathbf{S}(\mathbf{u} \otimes \mathbf{v}) \rangle$ takes only k time. As a comparison, computing $\mathbf{u}^\top \mathbf{A}^\top \mathbf{B}\mathbf{v}$ naïvely takes $nd^2 + O(d^2)$ time, which can be arbitrarily worse than our sketched version. Note that it is prohibitive in our setting to compute $\mathbf{u}^\top \mathbf{A}^\top$ and $\mathbf{B}\mathbf{v}$ separately. □

5 CONCLUSION

Our work highlights the many places where non-PSD matrices and their "square roots", which are complex matrices, arise in optimization and randomized numerical linear algebra. We give novel dimensionality reduction methods for such matrices in optimization, the sketch-and-solve paradigm, and for vector-matrix-vector queries. These methods can be used for approximating indefinite Hessian matrices, which constitute a major bottleneck for second-order optimization. We also propose a hybrid sampling method for matrices that satisfy a relaxed RIP condition. We verify these numerically using Newton-CG, trust region, and Newton-MR algorithms. We also show how to reduce complex ℓ_p-regression to real ℓ_p-regression in a black box way using random linear embeddings, showing that the many sketching techniques developed for real matrices can be applied to complex matrices as well. In addition, we also present how to efficiently sketch complex matrices for vector-matrix-vector queries.

Acknowledgements

The authors would like to thank partial support from NSF grant No. CCF-181584, Office of Naval Research (ONR) grant N00014-18-1-256, and a Simons Investigator Award.

References

Alexandr Andoni, Jiecao Chen, Robert Krauthgamer, Bo Qin, David P Woodruff, and Qin Zhang. On sketching

quadratic forms. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 311–319, 2016.

Albert E Beaton and John W Tukey. The fitting of power series, meaning polynomials, illustrated on band-spectroscopic data. *Technometrics*, 16(2):147–185, 1974.

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.

Yongxin Chen, Sei Zhen Khong, and Tryphon T Georgiou. On the definiteness of graph laplacians with negative weights: Geometrical and passivity-based approaches. In *2016 American Control Conference (ACC)*, pages 2488–2493. IEEE, 2016.

Kenneth L Clarkson. Subgradient and sampling algorithms for l 1 regression. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 257–266. Society for Industrial and Applied Mathematics, 2005.

Kenneth L Clarkson and David P Woodruff. Low-rank approximation and regression in input sparsity time. *Journal of the ACM (JACM)*, 63(6):1–45, 2017.

Kenneth L Clarkson, Petros Drineas, Malik Magdon-Ismail, Michael W Mahoney, Xiangrui Meng, and David P Woodruff. The fast cauchy transform and faster robust linear regression. *SIAM Journal on Computing*, 45(3): 763–810, 2016.

Michael B Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. In *Proceedings of the 51st annual ACM SIGACT symposium on theory of computing*, pages 938–942, 2019.

Andrew R Conn, Nicholas IM Gould, and Philippe L Toint. *Trust region methods*. SIAM, 2000.

Anirban Dasgupta, Petros Drineas, Boulos Harb, Ravi Kumar, and Michael W Mahoney. Sampling algorithms and coresets for \ell_p regression. *SIAM Journal on Computing*, 38(5):2060–2078, 2009.

Petros Drineas and Michael W Mahoney. RandNLA: Randomized Numerical Linear Algebra. *Communications of the ACM*, 59(6):80–90, 2016.

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

Murat A Erdogdu and Andrea Montanari. Convergence rates of sub-sampled Newton methods. In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, pages 3052–3060, 2015.

Piotr Indyk. Algorithmic applications of low-distortion geometric embeddings. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 10–33. IEEE, 2001.

Xuanqing Liu, Cho-Jui Hsieh, Jason D Lee, and Yuekai Sun. An inexact subsampled proximal Newton-type method for large-scale machine learning. *arXiv preprint arXiv:1708.08552*, 2017.

Yang Liu and Fred Roosta. Stability Analysis of Newton-MR Under Hessian Perturbations. *arXiv preprint arXiv:1909.06224*, 2019.

Michael W Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends® in Machine Learning*, 3(2):123–224, 2011.

Shannon McCurdy. Ridge regression and provable deterministic ridge leverage score sampling. In *Advances in Neural Information Processing Systems*, pages 2463–2472, 2018.

Xiangrui Meng and Michael W Mahoney. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 91–100, 2013.

Grigoris Paouris, Petros Valettas, and Joel Zinn. Random version of dvoretzky's theorem in ℓ_p^n. *Stochastic Processes and their Applications*, 127(10):3187–3227, 2017.

Ninh Pham and Rasmus Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 239–247, 2013.

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.

Cyrus Rashtchian, David P Woodruff, and Hanlin Zhu. Vector-matrix-vector queries for solving linear algebra, statistics, and graph problems. *arXiv preprint arXiv:2006.14015*, 2020.

Farbod Roosta and Michael W Mahoney. Sub-sampled Newton methods. *Mathematical Programming*, 174(1-2): 293–326, 2019.

Farbod Roosta, Yang Liu, Peng Xu, and Michael W Mahoney. Newton-MR: Newton's Method Without Smoothness or Convexity. *arXiv preprint arXiv:1810.00303*, 2018.

Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

Christian Sohler and David P Woodruff. Subspace embeddings for the l1-norm with applications. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 755–764, 2011.

Nilesh Tripuraneni, Mitchell Stern, Chi Jin, Jeffrey Regier, and Michael I Jordan. Stochastic cubic regularization for fast nonconvex optimization. In *Advances in neural information processing systems*, pages 2899–2908, 2018.

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

Ruosong Wang and David P Woodruff. Tight bounds for ℓ_p oblivious subspace embeddings. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1825–1843. SIAM, 2019.

David Woodruff and Qin Zhang. Subspace embeddings and\ell_p-regression using exponential random variables. In *Conference on Learning Theory*, pages 546–567, 2013.

David P Woodruff et al. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157, 2014.

Peng Xu, Jiyan Yang, Farbod Roosta, Christopher Ré, and Michael W Mahoney. Sub-sampled newton methods with non-uniform sampling. In *Advances in Neural Information Processing Systems*, pages 3000–3008, 2016.

Peng Xu, Farbod Roosta, and Michael W Mahoney. Newton-type methods for non-convex optimization under inexact Hessian information. *Mathematical Programming*, 2019. doi:10.1007/s10107-019-01405-z.

Peng Xu, Farbod Roosta, and Michael W. Mahoney. Second-Order Optimization for Non-Convex Machine Learning: An Empirical Study. In *Proceedings of the 2020 SIAM International Conference on Data Mining*. SIAM, 2020.

Zhewei Yao, Peng Xu, Farbod Roosta, and Michael W Mahoney. Inexact non-convex Newton-type methods. arXiv preprint arXiv:1802.06925, 2018. Under review.