Proceedings of Machine Learning Research vol 75:1–31, 2018 31st Annual Conference on Learning Theory

ℓ_1 Regression using Lewis Weights Preconditioning and Stochastic Gradient Descent

David Durfee DDURFEE@GATECH.EDU
Kevin A. Lai KEVINLAI@GATECH.EDU
Saurabh Sawlani SAWLANI@GATECH.EDU
Georgia Institute of Technology

Editors: Sebastien Bubeck, Vianney Perchet and Philippe Rigollet

Abstract

We present preconditioned stochastic gradient descent (SGD) algorithms for the ℓ_1 minimization problem $\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_1$ in the overdetermined case, where there are far more constraints than variables. Specifically, we have $\boldsymbol{A} \in \mathbb{R}^{n\times d}$ for $n \gg d$. Commonly known as the Least Absolute Deviations problem, ℓ_1 regression can be used to solve many important combinatorial problems, such as minimum cut and shortest path. SGD-based algorithms are appealing for their simplicity and practical efficiency. Our primary insight is that careful preprocessing can yield preconditioned matrices $\widetilde{\boldsymbol{A}}$ with strong properties (besides good condition number and low-dimension) that allow for faster convergence of gradient descent. In particular, we precondition using Lewis weights to obtain an isotropic matrix with fewer rows and strong upper bounds on all row norms. We leverage these conditions to find a good initialization, which we use along with recent smoothing reductions and accelerated stochastic gradient descent algorithms to achieve ϵ relative error in $\widetilde{O}(nnz(\boldsymbol{A}) + d^{2.5}\epsilon^{-2})$ time with high probability, where $nnz(\boldsymbol{A})$ is the number of non-zeros in $\boldsymbol{A}$. This improves over the previous best result using gradient descent for ℓ_1 regression. We also match the best known running times for interior point methods in several settings.

Finally, we also show that if our original matrix $\boldsymbol{A}$ is approximately isotropic and the row norms are approximately equal, we can give an algorithm that avoids using fast matrix multiplication and obtains a running time of $\widetilde{O}(nnz(\boldsymbol{A})+sd^{1.5}\epsilon^{-2}+d^2\epsilon^{-2})$, where s is the maximum number of non-zeros in a row of $\boldsymbol{A}$. In this setting, we beat the best interior point methods for certain parameter regimes.

Keywords: ℓ_1 regression, stochastic gradient descent, Lewis weights

1. Introduction

Stochastic gradient descent (SGD) is one of the most widely-used practical algorithms for optimization problems due to its simplicity and practical efficiency (Nesterov and Vial, 2008; Nemirovski et al., 2009). We consider SGD methods to solve the unconstrained overdetermined ℓ_1 regression problem, commonly known as the Least Absolute Deviations problem, which is defined as follows:

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_1\,, \tag{1}$$

where $\boldsymbol{A} \in \mathbb{R}^{n\times d}$, $\boldsymbol{b} \in \mathbb{R}^n$ and $n \gg d$. Compared to Least Squares (ℓ_2) regression, the ℓ_1 regression problem is more robust and is thus useful when outliers are present in the data. Moreover, many important combinatorial problems, such as minimum cut or shortest path, can be formulated as ℓ_1 regression problems (Chin et al., 2013), and high accuracy ℓ_1 regression can be used to solve general linear programs.[1] Since (1) can be formulated as a linear program (Portnoy and Koenker, 1997; Chen et al., 2001), generic methods for solving linear programs, such as the interior-point method (IPM), can be used to solve it (Portnoy and Koenker, 1997; Portnoy, 1997; Meng and Mahoney, 2013b; Lee and Sidford, 2015).

SGD algorithms are popular in practice for ℓ_1 and other regression problems because they are simple, scalable, and efficient. State-of-the-art algorithms for solving (1) utilize sketching techniques from randomized linear algebra to achieve $\text{poly}(d, \epsilon^{-1})$ running times, whereas a naive extension of Nesterov acceleration (Nesterov, 1983) to certain classes of non-smooth functions (Nesterov, 2005b,a, 2007; Allen-Zhu and Hazan, 2016) takes $\text{poly}(n, \epsilon^{-1})$ time. This difference is significant because $n \gg d$ in the overdetermined setting. Ideally, the only dependence on n in the running time will be implicitly in an additive $nnz(\boldsymbol{A})$ term.

Sketching techniques from randomized numerical linear algebra look to find a low-distortion embedding of $\boldsymbol{A}$ into a smaller subspace, after which popular techniques for ℓ_1 regression can be applied on the reduced matrix. Efforts to build these sampled matrices or "coresets" have been made using random sampling (Clarkson, 2005), fast matrix multiplication (Sohler and Woodruff, 2011), and ellipsoidal rounding (Dasgupta et al., 2009; Clarkson et al., 2013). All of these methods produce coresets of size $\text{poly}(d, \epsilon^{-1}) \times d$ in time $O(n \cdot \text{poly}(d))$. Meng and Mahoney (2013a) and Woodruff and Zhang (2013) improve these techniques to produce similar coresets in $O\left(nnz(\boldsymbol{A}) + \text{poly}(d)\right)$ time.

In addition to using sketching as a preprocessing step, one can also apply *preconditioners*. Preconditioners transform the input matrix into one with additional desirable properties, such as good condition number, that speed up subsequent computation. For our setting, we will use the term "preconditioning" to refer to dimensionality reduction followed by additional processing of the matrix. Of particular interest for our setting is the preconditioning technique of Cohen and Peng (2015), which utilizes a Lewis change of density (Lewis, 1978) to sample rows of $\boldsymbol{A}$ with probability proportional to their Lewis weights such that the sampled matrix $\tilde{\boldsymbol{A}}$ approximately preserves ℓ_1 distances, which is to say that $\|\boldsymbol{A}\boldsymbol{x}\|_1 \approx ||\tilde{\boldsymbol{A}}\boldsymbol{x}||_1$ for any vector $\boldsymbol{x}$. Lewis weights are in essence the "correct" sampling weights for dimensionality reduction in ℓ_1 regression, and they are used by the previous best SGD-based methods for solving (1) (Yang et al., 2016). As it turns out, Lewis weights also lead to nice ℓ_2 conditions for the sampled matrix. One of the key insights of this paper is to leverage these additional guarantees to obtain significantly faster running times for SGD after Lewis weight preconditioning.

Techniques Our techniques for solving the ℓ_1 regression problem follow the popular paradigm of preconditioning and then using gradient descent methods on the resulting problem. Typically, the preconditioner is a black-box combination of a sketching method with a matrix rotation, which yields a well-conditioned low-dimensional matrix. The crucial idea in this paper is that the sketch can give us some strong properties in addition to the low-dimensional embedding. By more tightly integrating the components of the preconditioner, we obtain faster running times for ℓ_1 regression.

In particular, preconditioning the given matrix-vector pair $[\boldsymbol{A}\ \boldsymbol{b}]$ using Lewis weights (Cohen and Peng, 2015) achieves a low-dimensional $[\tilde{\boldsymbol{A}}\ \tilde{\boldsymbol{b}}]$ such that $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_1 \approx \|\tilde{\boldsymbol{A}}\boldsymbol{x} - \tilde{\boldsymbol{b}}\|_1$ for every

1. For instance, one can determine if $\{\boldsymbol{x} | \boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}, \boldsymbol{x} \geq 0\}$ is feasible by writing an objective of the form $\alpha(\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_1 + \|\boldsymbol{x}\|_1 + \|\boldsymbol{x} - \beta\mathbb{1}\|_1)$ where α and β are sufficiently large polynomials in the input size.

$\boldsymbol{x} \in \mathbb{R}^d$. It is then possible to apply the low-dimensional embedding properties of Lewis weights in a black-box manner to ℓ_1 regression algorithms, using the fact that this embedding reduces the row-dimension from n to $O(d\epsilon^{-2}\log n)$ and then plugging these new parameters into the runtime guarantees. Our critical observation will be that sampling by Lewis weights also has the important property that all leverage scores of the new matrix are approximately equal. Since rotations of a matrix do not change its leverage scores, we are free to rotate $\tilde{\boldsymbol{A}}$ to place it into isotropic position, in which case the leverage score condition implies that the row norms are tightly bounded.

The isotropic and row norm conditions yield two essential phenomena. First, a careful choice of initial vector can be shown to be close to optimal. Second, we get strong bounds on the gradient of any row. Using these properties, it is almost immediately implied that standard SGD only requires $O(d^2\epsilon^{-2})$ iterations to arrive at a solution $\hat{x}$ with relative error[2] ϵ, leading to a total running time of $\widetilde{O}(nnz(\boldsymbol{A})+d^3\epsilon^{-2})$. These properties can be further applied to smoothing reductions by Allen-Zhu and Hazan (2016) and accelerated SGD algorithms by Allen Zhu (2017) to improve the runtime to $\widetilde{O}(nnz(\boldsymbol{A}) + nd^{\omega-1} + \sqrt{n}d^{1.5}\epsilon^{-1})$. As previously mentioned, sampling by Lewis weights guarantees that $n \leq O(d\epsilon^{-2}\log n)$, so we also obtain a running time of $\widetilde{O}(nnz(\boldsymbol{A})+d^{2.5}\epsilon^{-2})$. Algorithm 1 gives the basic framework of our ℓ_1 solver.

Algorithm 1: General structure of our algorithm

Input: Matrix $\boldsymbol{A} \in \mathbb{R}^{n\times d}$, and vector $\boldsymbol{b} \in \mathbb{R}^n$, along with error parameter $\epsilon > 0$.

1. Precondition $[\boldsymbol{A}\ \boldsymbol{b}]$ by Lewis weight sampling as in Cohen and Peng (2015), along with a matrix rotation.
2. Initialize $\boldsymbol{x}_0$ as the exact or approximate ℓ_2 minimizer of the preconditioned problem.
3. Run a stochastic gradient descent algorithm on the preconditioned problem with starting point $\boldsymbol{x}_0$.

Our Results Our first main theorem uses smoothing reductions from Allen-Zhu and Hazan (2016) with the accelerated SGD algorithm of Allen Zhu (2017):

Theorem 1 *Given $\boldsymbol{A} \in \mathbb{R}^{n\times d}$, $\boldsymbol{b} \in \mathbb{R}^n$, assume $\min_{\boldsymbol{x}} \|\boldsymbol{Ax}-\boldsymbol{b}\|_2$ is either 0 or bounded below by n^{-c} for some constant $c > 0$. Then for any $\epsilon > 0$, there is a routine that outputs $\tilde{\boldsymbol{x}}$ such that with high probability,*[3]

$$\|\boldsymbol{A}\tilde{\boldsymbol{x}} - \boldsymbol{b}\|_1 \leq (1+\epsilon)\min_{\boldsymbol{x}} \|\boldsymbol{Ax}-\boldsymbol{b}\|_1$$

with a runtime of $O\left(nnz(\boldsymbol{A})\log^2 n + d^{2.5}\epsilon^{-2}\log^{1.5} n\right)$ whenever $n \geq d\epsilon^{-2}\log n$, and a runtime of $O\left(\sqrt{n}d^2\epsilon^{-1}\log n + nd^{\omega-1}\log n\right)$ when $n \leq d\epsilon^{-2}\log n$.

Achieving the bounds when $n \leq d\epsilon^{-2}\log n$ requires some additional technical work. Theorem 1 is proved in Section 3, where we also show our $\widetilde{O}(nnz(\boldsymbol{A}) + d^3\epsilon^{-2})$ running time for standard SGD.

Our second main theorem is motivated by the fact that the theoretical running time bounds of fast matrix multiplication are difficult to achieve in practice, and most implementations of algorithms

2. Relative error ϵ means that $f(\hat{\boldsymbol{x}}) - f(\boldsymbol{x}^*) \leq \epsilon f(\boldsymbol{x}^*)$, where $f(\boldsymbol{x}) = \boldsymbol{Ax} - \boldsymbol{b}$ and $\boldsymbol{x}^* = \arg\min_x f(\boldsymbol{x})$.
3. Throughout this paper, we let "with high probability" mean that ξ is the failure probability and our runtime has dependence on $\log(1/\xi)$, which we ignore for ease of notation.

actually use naive matrix multiplication. Thus, it would be ideal for an algorithm's running time to be independent of fast matrix multiplication. It turns out that our only dependence on fast matrix multiplication is during the preconditioning stage. Accordingly, if we are given a matrix which is already approximately isotropic with all row norms approximately equal, then we can eliminate the usage of fast matrix multiplication and still prove the same time bound. Moreover, this method preserves the row-sparsity of $\boldsymbol{A}$. The primary difficulty of this approach is in computing an appropriate initialization when $\boldsymbol{A}$ is only approximately isotropic. To resolve this issue, we use efficient ℓ_2 regression solvers that do not rely on fast matrix multiplication.

Theorem 2 *Let $\boldsymbol{A} \in \mathbb{R}^{n \times d}$ and $\boldsymbol{b} \in \mathbb{R}^n$ be such that the matrix-vector pair $[\boldsymbol{A}\ \ \boldsymbol{b}]$ satisfies $[\boldsymbol{A}\ \ \boldsymbol{b}]^T[\boldsymbol{A}\ \ \boldsymbol{b}] \approx_{O(1)} \boldsymbol{I}$* [4] *and for each row i of $[\boldsymbol{A}\ \ \boldsymbol{b}]$, $\|[\boldsymbol{A}\ \ \boldsymbol{b}]_{i,:}\|_2^2 \approx_{O(1)} d/n$. Assume $\|\boldsymbol{b}\|_2 \leq n^c$ and $\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2$ is either 0 or bounded below by n^{-c} for some constant $c > 0$. Then for any $\epsilon > 0$, there is a routine that computes $\tilde{\boldsymbol{x}}$ such that with high probability,*

$$\|\boldsymbol{A}\tilde{\boldsymbol{x}} - \boldsymbol{b}\|_1 \leq (1+\epsilon) \min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_1$$

with a runtime of $O\left(nnz(\boldsymbol{A}) \log^2 n + s \cdot d^{1.5}\epsilon^{-2} \log^{1.5} n + d^2\epsilon^{-2}\log^2 n\right)$, where s is the maximum number of entries in any row of $\boldsymbol{A}$.

The added assumption on $\|b\|_2$ in Theorem 2 comes from the bounds of the ℓ_2 solvers. We prove Theorem 2 in Section 4.

Comparison of our results with previous work Our algorithms are significantly faster than the previous best SGD-based results for ℓ_1 regression, which took $O(nnz(\boldsymbol{A}) \log n + d^{4.5}\epsilon^{-2}\sqrt{\log d})$ time (Yang et al., 2016). Furthermore, our standard SGD bounds are especially likely to be achievable in practice. Table 1 compares the running time of our algorithm to the fastest gradient descent methods (Clarkson, 2005; Nesterov, 2009; Yang et al., 2016), interior point methods (Meng and Mahoney, 2013b; Lee and Sidford, 2015), and multiplicative weights update methods (Chin et al., 2013). Since we can apply sampling by Lewis weights prior to any algorithm[5], we can replace n with $O(d\epsilon^{-2}\log n)$ and $nnz(\boldsymbol{A})$ with $O(d^2\epsilon^{-2}\log n)$ at the cost of an additive $\widetilde{O}(nnz(\boldsymbol{A}) + d^\omega)$ overhead for any running time bound in Table 1, where d^ω is time to multiply two $d \times d$ matrices.[6]

Note that we match the theoretical performance of the current best IPM (Lee and Sidford, 2015) in several regimes. In low to medium-precision ranges, for example $\epsilon \geq 10^{-3}$, both the best IPM and our algorithm have a running time of $\widetilde{O}(nnz(\boldsymbol{A}) + d^{2.5})$. If all of the algorithms are implemented with naive matrix multiplication, Lee and Sidford (2015) takes $\widetilde{O}(nnz(\boldsymbol{A}) + d^3)$ time, while we prove an identical running time for standard non-accelerated SGD with Lewis weights preconditioning. For general ϵ, if $nnz(\boldsymbol{A}) \geq d^2\epsilon^{-2}\log n$, then Lee and Sidford (2015) will use Lewis weights sampling and both their algorithm and our algorithm will achieve a running time of $\widetilde{O}(nnz(\boldsymbol{A}) + d^{2.5}\epsilon^{-2})$. This is significant because our SGD-based algorithms are likely to be far more practical[7]. Finally, we also note that in the setting where $\boldsymbol{A}$ is approximately isotropic with approximately equal row norms and $\boldsymbol{A}$ is row-sparse, with at most s non-zeros per row, our algorithm in Theorem 2 has the best dimensional-dependence out of any existing algorithm for

4. As defined in the notation section, we say that $\boldsymbol{A} \approx_\kappa \boldsymbol{B}$ if and only if $\frac{1}{\kappa}\boldsymbol{B} \preceq \boldsymbol{A} \preceq \kappa\boldsymbol{B}$.
5. Sampling $\boldsymbol{A}$ by Lewis weights creates a dense $(d\epsilon^{-2}\log n) \times d$ matrix $\tilde{\boldsymbol{A}}$
6. The current best value for ω is approximately 2.373 (Williams, 2012; Davie and Stothers, 2013; Le Gall, 2014).
7. Lewis weights preconditioning is also fast in practice

$s < d$. In particular, we beat Lee and Sidford (2015) whenever $s < d\epsilon^2$ or whenever $s < d$ and $nnz(\boldsymbol{A}) \geq d^2\epsilon^{-2}\log n$.

Solver	**Running time** [8]
Subgradient descent Clarkson (2005)	$\widetilde{O}\left(nd^5\epsilon^{-2}\log(1/\epsilon)\right)$
Smoothing and gradient descent Nesterov (2009)	$\widetilde{O}\left(n^{1.5}d\epsilon^{-1}\right)$
Randomized IPCPM[9] Meng and Mahoney (2013b)	$\widetilde{O}\left(nd^{\omega-1} + (nnz(\boldsymbol{A}) + \text{poly}(d))\, d\log(1/\epsilon)\right)$
Multiplicative weights Chin et al. (2013)	$\widetilde{O}(n^{1/3}(nnz(\boldsymbol{A}) + d^2)\epsilon^{-8/3})$
IPM and fast inverse maintenance Lee and Sidford (2015)	$\widetilde{O}((nnz(\boldsymbol{A}) + d^2)\sqrt{d}\log(1/\epsilon))$
Weighted SGD Yang et al. (2016)	$\widetilde{O}(nnz(\boldsymbol{A}) + d^{4.5}\epsilon^{-2})$
Lewis weights and SGD (this work)	$\widetilde{O}\left(nnz(\boldsymbol{A}) + d^3\epsilon^{-2}\right)$
Lewis weights and accelerated SGD (this work)[10]	$\widetilde{O}\left(nnz(\boldsymbol{A}) + d^{2.5}\epsilon^{-2}\right)$

Table 1: Running time of several iterative ℓ_1 regression algorithms. All running times are to find a solution with ϵ relative error, with constant failure probability. Note that the first three algorithms in the table could also be sped up using the preconditioning method we use, i.e., Lewis weights row sampling (Cohen and Peng, 2015). Doing this would need a preconditioning time of $\widetilde{O}(nnz(\boldsymbol{A}) + d^\omega)$, and enable us to use the fact that $n \leq O(d\epsilon^{-2}\log n)$ after preconditioning. However, this still does not make them faster than later algorithms.

Another related work by Bubeck et al. (2017) gives algorithms for ℓ_p regression for $p \in (1, \infty)$ that run in time $\widetilde{O}(nnz(\boldsymbol{A}) \cdot n^{|1/2-1/p|}\log(1/\epsilon))$ and $\widetilde{O}(nnz(\boldsymbol{A})n^{|1/2-1/p|-1/2}d^{1/2} + n^{|1/2-1/p|}d^2 + d^\omega)\log(1/\epsilon))$. They also use preconditioning and accelerated stochastic gradient descent techniques as a subroutine. However, they don't give bounds for ℓ_1 regression. Also, for p close to 1, these bounds are worse than ours. In contrast to Bubeck et al. (2017), our algorithms are specific to the ℓ_1 setting. We use the fact that the gradients are bounded for ℓ_1 regression, which doesn't hold for general ℓ_p regression. Moreover, our initialization using an ℓ_2 minimizer doesn't give good bounds for general ℓ_p regression. In this sense, our algorithms really leverage the special structure of the ℓ_1 problem.

1.1. Organization

The paper is organized as follows. Section 2 contains definitions and basic lemmas which we will use throughout the paper. Section 3 contains our main contribution, i.e., once we are given a suitably preconditioned matrix, it shows how we arrive at an approximate ℓ_1 minimizer within the claimed time bounds, for both non-accelerated and accelerated versions of stochastic gradient descent. Section 4 shows that if we restrict our input to slightly weaker preconditions, then we can eliminate the need for fast matrix multiplication to achieve the same time bounds. Appendix A contains the primary proof details for our main result in Section 3. In Appendix B, we show that row sampling using Lewis weights Cohen and Peng (2015), along with matrix rotation, suffices to

8. $\widetilde{O}$ hides terms polylogarithmic in d and n.
9. Interior Point Cutting Plane Methods
10. This running time only assumes that $\omega \leq 2.5$

give us a matrix satisfying our precondition requirements. Appendix C contains proof details from Section 4. Finally, Appendix D will give secondary and straightforward technical details for our main results that we include for completeness.

2. Preliminaries

In this section, we describe some of the notation and important definitions we use in the paper. We represent matrices and vectors using bold variables. We let $\boldsymbol{A}_{i,:}$ denote the i^{th} row of a matrix $\boldsymbol{A}$, and we use $nnz(\boldsymbol{A})$ to denote the number of non-zero elements in $\boldsymbol{A}$. $\boldsymbol{A}^\dagger$ refers to the Moore-Penrose pseudoinverse of $\boldsymbol{A}$. When $\boldsymbol{A}$ has linearly-independent columns, $\boldsymbol{A}^\dagger = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T$. Also, we assume that the input $\boldsymbol{A}$ has full rank.

Definition 3 (ℓ_p-norm) *The ℓ_p norm of a vector $\boldsymbol{v} \in \mathbb{R}^n$ is defined as*

$$\|\boldsymbol{v}\|_p \overset{\text{def}}{=} \left(\sum_{i=1}^n \boldsymbol{v}_i^p\right)^{1/p}.$$

Accordingly, the ℓ_p norm of a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times d}$ is defined as

$$\|\boldsymbol{A}\|_p \overset{\text{def}}{=} \sup_{\boldsymbol{x}\in\mathbb{R}^d, \boldsymbol{x}\neq 0} \frac{\|\boldsymbol{A}\boldsymbol{x}\|_p}{\|\boldsymbol{x}\|_p}.$$

Definition 4 (Matrix approximation) *We say that $\boldsymbol{A} \approx_\kappa \boldsymbol{B}$ if and only if*

$$\frac{1}{\kappa}\boldsymbol{B} \preceq \boldsymbol{A} \preceq \kappa\boldsymbol{B}.$$

Here, $\preceq$ refers to the Löwner partial ordering of matrices, where we say that $\boldsymbol{A} \preceq \boldsymbol{B}$ if $\boldsymbol{B} - \boldsymbol{A}$ is positive semi-definite.

Note that we also use $\approx$ similarly in the case of scalars, as is commonplace.

Definition 5 (IRB) *A matrix $\boldsymbol{A} \in \mathbb{R}^{n\times d}$ with $n \geq d$ is said to be isotropic row-bounded (IRB) if the following hold:*

1. *$\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}$,*
2. *For all rows of $\boldsymbol{A}$, $\|\boldsymbol{A}_{i,:}\|_2^2 \leq O(d/n)$.*

Definition 6 *Given a matrix $\boldsymbol{A}$, we define the statistical leverage score of row $\boldsymbol{A}_{i,:}$ to be*

$$\tau_i(\boldsymbol{A}) \overset{\text{def}}{=} \boldsymbol{A}_{i,:}\left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-1}\boldsymbol{A}_{i,:}^T = \left\|\left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-1/2}\boldsymbol{A}_{i,:}^T\right\|_2^2.$$

Definition 7 *For a matrix $\boldsymbol{A}$, the ℓ_1 Lewis weights $\overline{\boldsymbol{w}}$ are the unique weights such that for each row i we have*

$$\overline{\boldsymbol{w}}_i = \tau_i\left(\overline{\boldsymbol{W}}^{-1/2}\boldsymbol{A}\right)$$

or equivalently

$$\overline{\boldsymbol{w}}^2 = \boldsymbol{A}_{i,:}\left(\boldsymbol{A}^T\overline{\boldsymbol{W}}^{-1}\boldsymbol{A}\right)^{-1}\boldsymbol{A}_{i,:}^T$$

where $\overline{\boldsymbol{W}}$ is the diagonal matrix formed by putting the elements of $\overline{\boldsymbol{w}}$ on the diagonal.

Definition 8 *A function f is L-smooth if for any $x, y \in \mathbb{R}^d$, $||\nabla f(x) - \nabla f(y)||_2 \leq L||x - y||_2$.*

Definition 9 *A function f is σ-strongly convex if for any $x, y \in \mathbb{R}^d$,*

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \tfrac{\sigma}{2}||x - y||_2^2.$$

Definition 10 *A function f is G-Lipschitz continuous if for any $x, y \in \mathbb{R}^d$,*

$$||f(x) - f(y)||_2 \leq G||x - y||_2.$$

3. Stochastic Gradient Descent for ℓ_1 Regression

In this section, we describe how we achieve the bounds in Theorem 1. We first introduce the preconditioning technique by Cohen and Peng (2015), which, along with rotating the matrix, will reduce our problem to ℓ_1 minimization where the input matrix $\boldsymbol{A}$ is isotropic and the norms of all its rows have strong upper bounds, i.e. it is IRB by Definition 5. We relegate the details and proof of this preconditioning procedure to Appendix B.

In Section 3.2, we prove that known stochastic gradient descent algorithms will run provably faster if we assume that $\boldsymbol{A}$ is IRB. In particular, if $\boldsymbol{A}$ is IRB, we can find an initialization $\boldsymbol{x}_0$ that is close to the optimum $\boldsymbol{x}^*$, which in addition to bounding the gradient of our objective, will then allow us to plug these bounds into standard stochastic gradient descent algorithms and achieve a runtime of $\widetilde{O}(nnz(\boldsymbol{A}) + d^3\epsilon^{-2})$. Finally, in Section 3.3 we take known smoothing techniques by Allen-Zhu and Hazan (2016) along with the Katyusha accelerated stochastic gradient descent by Allen Zhu (2017) to achieve a runtime of $\widetilde{O}(nnz(\boldsymbol{A}) + d^{2.5}\epsilon^{-2})$.

3.1. Preconditioning with Lewis weights

The primary tool in our preconditioning routine will be a sampling scheme by Lewis weights, introduced in Cohen and Peng (2015), that was shown to approximately preserve the ℓ_1 norm. Specifically, we will use the combination of two primary theorems from Cohen and Peng (2015) that approximately compute the Lewis weights of a matrix quickly and then sample accordingly while still approximately preserving ℓ_1 norm distances with high probability.

Theorem 11 (Theorem 2.3 and 6.1 from Cohen and Peng (2015)) *Given a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times d}$ with ℓ_1 Lewis weights $\overline{\boldsymbol{w}}$ and an error parameter $\epsilon > 0$, then for any function $h(n, \epsilon) \geq O(\epsilon^{-2} \log n)$, we can find sampling values*

$$\boldsymbol{p}_i \approx_{O(1)} \overline{\boldsymbol{w}}_i h(n, \epsilon)$$

for each $i \in \{1, 2, \ldots, n\}$, and generate a matrix $\boldsymbol{S}$ with $N = \sum_i \boldsymbol{p}_i$ rows, each chosen independently as the i^{th} standard basis vector of dimension n, times $\frac{1}{\boldsymbol{p}_i}$ with probability proportional to $\boldsymbol{p}_i$, such that with high probability we have

$$\|\tilde{\boldsymbol{A}}\boldsymbol{x}\|_1 \approx_{1+\epsilon} \|\boldsymbol{A}\boldsymbol{x}\|_1$$

for all $\boldsymbol{x} \in \mathbb{R}^d$, where $\tilde{\boldsymbol{A}} = \boldsymbol{S}\boldsymbol{A}$. Computing these sampling values requires $O(nnz(\boldsymbol{A}) \log n + d^\omega)$ time.

In Appendix B, we will show that this sampling scheme also ensures that each row of $\tilde{\boldsymbol{A}}$ has approximately the same leverage score. This proof will involve applying known facts about leverage scores and their connections with Lewis weights, along with standard matrix concentration bounds. Furthermore, we will obtain additional nice properties by rotating $\boldsymbol{A}$ and showing that a solution to our reduced problem gives an approximate solution to the original problem, culminating in the following lemma:

Lemma 12 *There is a routine that takes a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times d}$, a vector $\boldsymbol{b} \in \mathbb{R}^n$ and $\epsilon > 0$, then produces a matrix $[\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}}] \in \mathbb{R}^{N\times(d+1)}$ with $N = O(d\epsilon^{-2}\log n)$ and an invertible matrix $\boldsymbol{U} \in \mathbb{R}^{d\times d}$ such that matrix $\tilde{\boldsymbol{A}}\boldsymbol{U}$ is IRB and if $\tilde{\boldsymbol{x}}^*_U$ minimizes $\|\tilde{\boldsymbol{A}}\boldsymbol{U}\boldsymbol{x} - \tilde{b}\|_1$, then for any $\tilde{\boldsymbol{x}}$ such that*

$$\|\tilde{\boldsymbol{A}}\boldsymbol{U}\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{b}}\|_1 \leq (1+\delta)\|\tilde{\boldsymbol{A}}\boldsymbol{U}\tilde{\boldsymbol{x}}^*_U - \tilde{\boldsymbol{b}}\|_1,$$

holds for some $\delta > 0$, we must have

$$\|\boldsymbol{A}(\boldsymbol{U}\tilde{\boldsymbol{x}}) - \boldsymbol{b}\|_1 \leq (1+\epsilon)^2(1+\delta)\|\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b}\|_1$$

with high probability.

Furthermore, the full running time is $O(nnz(\boldsymbol{A})\log n + d^{\omega-1}\min\{d\epsilon^{-2}\log n, n\} + \Upsilon)$ where $\Upsilon = \min\{d\epsilon^{-2}\log n, (d\epsilon^{-2}\log n)^{1/2+o(1)} + n\log^2 n\}$.

As a result, we will assume that all of our matrices $\boldsymbol{A}$ are already in the same form as $\tilde{\boldsymbol{A}}\boldsymbol{U}$, i.e. we assume $\boldsymbol{A}$ is IRB, since relative error guarantees for the preconditioned system apply to the original system.

3.2. Isotropic and Row-Bounded A for Stochastic Gradient Descent

To demonstrate the usefulness of the properties of our preconditioned $\boldsymbol{A}$, we consider standard stochastic gradient descent and the bounds on its running time. We let $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_1$. We will use the following theorem to prove runtime bounds for standard SGD:

Theorem 13 (Ruszczynski and Syski (1986)) *Given a function f and $\boldsymbol{x}_0$ such that $\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_2 \leq R$ and L is an upper bound on the ℓ_2 norm of the stochastic subgradients of f, then projected subgradient descent ensures that after t steps:*

$$f(\boldsymbol{x}_t) - f(\boldsymbol{x}^*) \leq O\left(\frac{RL}{\sqrt{t}}\right).$$

where $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} f(\boldsymbol{x})$.

To use this theorem, we must prove bounds on the initialization distance $\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_2$ and the norm of the stochastic subgradients we use, i.e. $\nabla(n \cdot |\boldsymbol{A}_{i,:}\boldsymbol{x} - \boldsymbol{b}_i|)$ for each i. We show that both of these bounds come from our assumptions on $\boldsymbol{A}$.

Lemma 14 *If $\boldsymbol{A}$ is IRB, then by setting $\boldsymbol{x}_0 = \boldsymbol{A}^T\boldsymbol{b}$ we have*

$$\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_2^2 \leq O(d/n)\|\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b}\|_1^2.$$

Proof

$$\begin{aligned}
\left\|\boldsymbol{x}^{(0)} - \boldsymbol{x}^{(*)}\right\|_2^2 &= \left\|\boldsymbol{A}^T\boldsymbol{b} - \boldsymbol{x}^{(*)}\right\|_2^2 \\
&= \left\|\boldsymbol{A}^T\left(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(*)}\right)\right\|_2^2 && \text{by assumption } \boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I} \\
&= \left\|\textstyle\sum_i \boldsymbol{A}_{i,:}\left(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^{(*)}\right)\right\|_2^2 \\
&\le \|\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b}\|_1^2 \cdot \max_i ||\boldsymbol{A}_{i,:}||_2^2 && \text{by convexity of } \|\cdot\|_2 \text{, also shown in Lemma 40} \\
&= O\left(\frac{d}{n}\right)\|\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b}\|_1^2 .
\end{aligned}$$

■

Lemma 15 *If $\boldsymbol{A}$ is IRB, then $\|\nabla(n \cdot |\boldsymbol{A}_{i,:}\boldsymbol{x} - \boldsymbol{b}_i|)\|_2^2 \le O(nd)$ for all i.*

Proof We see that $\nabla(n \cdot |\boldsymbol{A}_{i,:}\boldsymbol{x} - \boldsymbol{b}_i|) = n \cdot \boldsymbol{A}_{i,:}^T \text{sgn}(\boldsymbol{A}_{i,:}\boldsymbol{x} - \boldsymbol{b}_i)$, and $\|\boldsymbol{A}_{i,:}\|_2^2 \le O(d/n)$ for all i by our assumption that $\boldsymbol{A}$ is IRB. This then implies our desired inequality. ■

These bounds, particularly the initialization distance, are stronger than the bounds for general $\boldsymbol{A}$, and together will give our first result that improves upon the runtime given by Yang et al. (2016) by using our preconditioning.

Theorem 16 *Given $\boldsymbol{A} \in \mathbb{R}^{n\times d}$, we can find $\tilde{\boldsymbol{x}} \in \mathbb{R}^d$ using preconditioning and stochastic gradient descent such that*

$$\|\boldsymbol{A}\tilde{\boldsymbol{x}} - \boldsymbol{b}\|_1 \le (1+\epsilon)\|\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b}\|_1$$

in time $O(nnz(\boldsymbol{A})\log^2 n + d^3\epsilon^{-2}\log n)$.

Proof By preconditioning with Lemma 12 and error $O(\epsilon)$ we obtain an $N \times d$ matrix $\tilde{\boldsymbol{A}}\boldsymbol{U}$ in time $O(nnz(\boldsymbol{A})\log n + d^\omega\epsilon^{-2})$.

By Theorem 13, we then need to run $O(d^2\epsilon^{-2})$ iterations of standard stochastic gradient descent to achieve absolute error of $O(\epsilon \cdot f(\boldsymbol{x}^*))$ which is equivalent to relative error of $O(\epsilon)$. The required runtime is then $O(d^3\epsilon^{-2})$. Technically, the input to stochastic gradient descent will require the value R, i.e. the upper bound on initialization distance, which requires access to a constant factor approximation of $f(\boldsymbol{x}^*)$. We will show in Appendix D.2 that we can assume that we have such an approximation at the cost of a factor of $\log n$ in the running time.

Combining the preconditioning and stochastic gradient descent will produce $\tilde{\boldsymbol{x}}$ with $O(\epsilon)$ relative error to the optimal objective function value in time $O(nnz(\boldsymbol{A})\log n + d^3\epsilon^{-2})$. Adding the factor $\log n$ overhead from estimating $f(\boldsymbol{x}^*)$ gives the desired runtime. ■

3.3. Smoothing Reductions and Katyusha Accelerated SGD

In order to improve the running time guarantees, we consider whether our strong initialization distance bound will allow us to apply black-box accelerated stochastic gradient descent methods. These methods generally require smoothness and strong convexity of the objective function, neither of which are necessarily true for our objective function. Previous results (Nesterov, 2005b, 2007;

Duchi et al., 2012; Ouyang and Gray, 2012; Allen-Zhu and Hazan, 2016) have addressed this general issue and given reductions from certain classes of objective functions to similar functions with smoothness and strong convexity, while still maintaining specific error and runtime guarantees. Accordingly, we will first show how our initialization distance fits into the reduction of Allen-Zhu and Hazan (2016), then apply Katyusha's accelerated gradient descent algorithm by Allen Zhu (2017) to their framework. We state the theorem below, and relegate the details of its proof to Appendix A.

Theorem 1 *Given $\boldsymbol{A} \in \mathbb{R}^{n\times d}$, $\boldsymbol{b} \in \mathbb{R}^n$, assume $\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2$ is either 0 or bounded below by n^{-c} for some constant $c > 0$. Then for any $\epsilon > 0$, there is a routine that outputs $\tilde{\boldsymbol{x}}$ such that with high probability,*[11]

$$\|\boldsymbol{A}\tilde{\boldsymbol{x}} - \boldsymbol{b}\|_1 \leq (1+\epsilon)\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_1$$

with a runtime of $O\left(nnz(\boldsymbol{A})\log^2 n + d^{2.5}\epsilon^{-2}\log^{1.5} n\right)$ whenever $n \geq d\epsilon^{-2}\log n$, and a runtime of $O\left(\sqrt{n}d^2\epsilon^{-1}\log n + nd^{\omega-1}\log n\right)$ when $n \leq d\epsilon^{-2}\log n$.

The last bound in Theorem 1 may seem odd, as Lewis weights sampling produces a matrix with row-dimension $d\epsilon^{-2}\log n$, which will in fact hurt our running time if $n \ll d\epsilon^{-2}\log n$. Moreover, we cannot simply avoid running the Lewis weights sampling because our algorithm critically relies on the resulting leverage score properties. Instead, if $n \ll d\epsilon^{-2}\log n$, we can simulate the sampling procedure in $O(n)$ time and keep a count for each of the n unique rows. Since the simulated sample matrix will look like the original but with duplicated rows, we will be able to carry out the rest of our linear algebraic manipulations in time dependent on n rather than $d\epsilon^{-2}\log n$. We will address this in Appendix D.1.

4. Row-Sparsity Bounds for ℓ_1 Regression

In this section, we explain how to avoid using matrix multiplication, which we use in Lemma 12, to get an IRB matrix, and in Lemma 14, to get a good initialization. To avoid both procedures, we assume that our given matrix $\boldsymbol{A} \in \mathbb{R}^{n\times d}$ and vector $\boldsymbol{b} \in \mathbb{R}^n$ are such that $[\boldsymbol{A}\ \boldsymbol{b}]^T[\boldsymbol{A}\ \boldsymbol{b}] \approx_{O(1)} \boldsymbol{I}$ and for each row i of $[\boldsymbol{A}\ \boldsymbol{b}]$ we have $\|[\boldsymbol{A}\ \boldsymbol{b}]_{i,:}\|_2^2 \approx_{O(1)} d/n$. Notice that these conditions imply $\boldsymbol{A}^T\boldsymbol{A} \approx_{O(1)} \boldsymbol{I}$ and $\|\boldsymbol{A}_{i,:}\|_2^2 \leq O(d/n)$, which are nearly the properties of the preconditioned matrix $\tilde{\boldsymbol{A}}\boldsymbol{U}$ generated in Lemma 12. However, we still face two new complications: (1) the row count of matrix $\boldsymbol{A}$ has not been reduced from n to $O(d\epsilon^{-2}\log n)$, and (2) $\boldsymbol{A}$ is only approximately isotropic.

We will account for the first complication by showing that, under these conditions, the Lewis weights are approximately equal, which implies that uniform row sampling is nearly equivalent to that in Theorem 11. We then describe how to find a good initialization point using conjugate gradient methods when $\tilde{\boldsymbol{A}}$ is only approximately isotropic. Finally, we use the reduction in Allen-Zhu and Hazan (2016) and the Katyusha stochastic gradient descent algorithm from Allen Zhu (2017) to achieve a total running time of $\tilde{O}(nnz(\boldsymbol{A}) + sd^{1.5}\epsilon^{-2} + d^2\epsilon^{-2})$, where s is the maximum number of non-zeros in a row of $\boldsymbol{A}$. Note that as a byproduct of this algorithm, we achieve row-sparsity-preserving ℓ_1 minimization.

11. Throughout this paper, we let "with high probability" mean that ξ is the failure probability and our runtime has dependence on $\log(1/\xi)$, which we ignore for ease of notation.

Uniform Sampling of $\boldsymbol{A}$

We deal with the first complication of avoiding Lemma 12 by sampling $\boldsymbol{A}$ uniformly. In particular, if we uniformly sample $N = O(d\epsilon^{-2}\log d)$ rows of $[\boldsymbol{A}\ \boldsymbol{b}]$, this yields a smaller matrix $[\tilde{\boldsymbol{A}}\ \tilde{\boldsymbol{b}}]$ such that $\tilde{\boldsymbol{A}}^T\tilde{\boldsymbol{A}} \approx_{O(1)} \boldsymbol{I}$ and $\|\tilde{\boldsymbol{A}}_{i,:}\|_2^2 \le O(d/N)$ for each row i, which is to say $\tilde{\boldsymbol{A}}$ is almost IRB. This then culminates in the following lemma whose proof we relegate to Appendix C.1.

Lemma 17 *Suppose we are given a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times d}$ such that $\boldsymbol{A}^T\boldsymbol{A} \approx_{O(1)} \boldsymbol{I}$ and $\|\boldsymbol{A}_{i,:}\|_2^2 \approx_{O(1)} d/n$. If we uniformly sample $N = O(d\epsilon^{-2}\log n)$ rows independently and rescale each row by n/N to obtain matrix $\tilde{\boldsymbol{A}}$, then with high probability the following properties hold:*

1. $\|\boldsymbol{A}\boldsymbol{x}\|_1 \approx_{1+\epsilon} \|\tilde{\boldsymbol{A}}\boldsymbol{x}\|_1$ *for all* $\boldsymbol{x} \in \mathbb{R}^d$.
2. $\tilde{\boldsymbol{A}}^T\tilde{\boldsymbol{A}} \approx_{O(1)} \left(\frac{n}{N}\right)\boldsymbol{I}$.
3. $\|\tilde{\boldsymbol{A}}_{i,:}\|_2^2 \approx_{O(1)} dn/N^2$ *for all rows* $i \in \{1,2,\ldots,N\}$.

Initialization using Approximate ℓ_2 Minimizer

It now remains to show that we can still find a good initialization $\boldsymbol{x}_0$ for gradient descent even with our relaxed assumptions on $\boldsymbol{A}$. Previously, when we had $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}$, we used $\boldsymbol{x}_0 = \boldsymbol{A}^T\boldsymbol{b} = \arg\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2$. It turns out that for $\boldsymbol{A}^T\boldsymbol{A} \approx_{O(1)} \boldsymbol{I}$, the ℓ_2 minimizer $\boldsymbol{x}_0 = \arg\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2$ is still a good initialization point. But finding an exact ℓ_2 minimizer would take a prohibitive amount of time or would require matrix multiplication. However, an approximate ℓ_2 minimizer suffices, and we can find such a point quickly using the conjugate gradient method.

For this section, we define $\boldsymbol{x}^* \stackrel{\text{def}}{=} \arg\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_1$ and $\boldsymbol{x}_0 \stackrel{\text{def}}{=} \arg\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_2$. Our main result is the following:

Lemma 18 (Approximate ℓ_2 minimizer is close to x^*) *Let $\boldsymbol{A} \in \mathbb{R}^{n\times d}$ be such that $\boldsymbol{A}^T\boldsymbol{A} \approx_{O(1)} \boldsymbol{I}$ and for each row i of $\boldsymbol{A}$, $\|\boldsymbol{A}_{i,:}\|_2^2 \le O(d/n)$. Assume that $\|\boldsymbol{b}\|_2 \le n^c$ and $\|\boldsymbol{A}\boldsymbol{x}_0-\boldsymbol{b}\|_2 \ge n^{-c}$ for some constant $c > 0$.*

$$\|\tilde{\boldsymbol{x}}_0 - \boldsymbol{x}^*\|_2 \le O(\sqrt{d/n})\,\|\boldsymbol{A}\boldsymbol{x}^*-\boldsymbol{b}\|_1 .$$

Moreover, $\tilde{\boldsymbol{x}}_0$ can be computed in $O((t_{\boldsymbol{A}^T\boldsymbol{A}}+d)\log(n/\epsilon))$ time, where $t_{\boldsymbol{A}^T\boldsymbol{A}}$ denotes the time to multiply a vector by $\boldsymbol{A}^T\boldsymbol{A}$.

We prove Lemma 18 using the following two lemmas whose proofs are deferred until Appendix C.2. Lemma 19 shows that the ℓ_2 minimizer is close to the ℓ_1 minimizer even when $\boldsymbol{A}$ is only approximately isotropic, and the proof is similar to that of Lemma 14. Lemma 20 shows that we can find a good estimate for the ℓ_2 minimizer.

Lemma 19 (Exact ℓ_2 minimizer is close to $\boldsymbol{x}^*$) *Let $\boldsymbol{A} \in \mathbb{R}^{n\times d}$ be such that $\boldsymbol{A}^T\boldsymbol{A} \approx_{O(1)} \boldsymbol{I}$ and for each row i of $\boldsymbol{A}$, $\|\boldsymbol{A}_{i,:}\|_2^2 \le O(d/n)$. Then*

$$\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_2 \le O(\sqrt{d/n})\,\|\boldsymbol{A}\boldsymbol{x}^*-\boldsymbol{b}\|_1 .$$

Lemma 20 (Conjugate gradient finds an approximate ℓ_2 minimizer) *Let $\boldsymbol{A} \in \mathbb{R}^{n\times d}$ be such that $\boldsymbol{A}^T\boldsymbol{A} \approx_{O(1)} \boldsymbol{I}$ and for each row i of $\boldsymbol{A}$, $\|\boldsymbol{A}_{i,:}\|_2^2 \le O(d/n)$. Assume that $\|\boldsymbol{b}\|_2 \le n^c$ and $\|\boldsymbol{A}\boldsymbol{x}_0 - \boldsymbol{b}\|_2 \ge n^{-c}$. Then for any $\epsilon > 0$, the conjugate gradient method can find an $\tilde{\boldsymbol{x}}_0$ such that*

$$\|\tilde{\boldsymbol{x}}_0 - \boldsymbol{x}_0\|_2 \le \epsilon \|\boldsymbol{A}\boldsymbol{x}_0 - \boldsymbol{b}\|_2 .$$

Moreover, $\tilde{\boldsymbol{x}}_0$ can be found in time $O((t_{\boldsymbol{A}^T\boldsymbol{A}} + d)\log(n/\epsilon))$, where $t_{\boldsymbol{A}^T\boldsymbol{A}}$ is the time to multiply a vector by $\boldsymbol{A}^T\boldsymbol{A}$.

Proof [Proof of Lemma 18] We use conjugate gradient with $\epsilon = \sqrt{d/n}$ to find an $\tilde{\boldsymbol{x}}_0$ in $O((t_{\boldsymbol{A}^T\boldsymbol{A}} + d)\log(n/\epsilon))$ time by Lemma 20 to achieve our desired initialization time bounds. Note that by definition of $\boldsymbol{x}_0$ and by a standard norm inequality, we have:

$$\|\boldsymbol{A}\boldsymbol{x}_0 - \boldsymbol{b}\|_2 \le \|\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b}\|_2 \le \|\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b}\|_1$$

Then by the triangle inequality and Lemma 19 we have:

$$\begin{aligned}\|\tilde{\boldsymbol{x}}_0 - \boldsymbol{x}^*\|_2 &\le \|\tilde{\boldsymbol{x}}_0 - \boldsymbol{x}_0\|_2 + \|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_2 \\ &\le \sqrt{d/n}\,\|\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b}\|_1 + O(\sqrt{d/n})\,\|\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b}\|_1\end{aligned}$$

which gives our desired initialization accuracy. ■

FAST ROW-SPARSITY-PRESERVING ℓ_1 MINIMIZATION

Finally, we combine the matrix achieved by uniform sampling in Lemma 17 and the initialization from Lemma 18 to achieve fast row-sparsity-preserving ℓ_1 minimization. This then gives the following main theorem that we prove in Appendix C.3.

Theorem 2 *Let $\boldsymbol{A} \in \mathbb{R}^{n\times d}$ and $\boldsymbol{b} \in \mathbb{R}^n$ be such that the matrix-vector pair $[\boldsymbol{A}\ \ \boldsymbol{b}]$ satisfies $[\boldsymbol{A}\ \ \boldsymbol{b}]^T[\boldsymbol{A}\ \ \boldsymbol{b}] \approx_{O(1)} \boldsymbol{I}$ [12] and for each row i of $[\boldsymbol{A}\ \ \boldsymbol{b}]$, $\|[\boldsymbol{A}\ \ \boldsymbol{b}]_{i,:}\|_2^2 \approx_{O(1)} d/n$. Assume $\|\boldsymbol{b}\|_2 \le n^c$ and $\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2$ is either 0 or bounded below by n^{-c} for some constant $c > 0$. Then for any $\epsilon > 0$, there is a routine that computes $\tilde{\boldsymbol{x}}$ such that with high probability,*

$$\|\boldsymbol{A}\tilde{\boldsymbol{x}} - \boldsymbol{b}\|_1 \le (1+\epsilon)\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_1$$

with a runtime of $O\left(nnz(\boldsymbol{A})\log^2 n + s\cdot d^{1.5}\epsilon^{-2}\log^{1.5} n + d^2\epsilon^{-2}\log^2 n\right)$, where s is the maximum number of entries in any row of $\boldsymbol{A}$.

Acknowledgments

We are grateful to Richard Peng for many insightful discussions. We would also like to thank Michael Cohen for his helpful comments and Junxing Wang for his feedback. We thank the anonymous reviews of an earlier version of this manuscript for their very valuable comments, especially with regards to the performances of the many existing algorithms in the ℓ_1 case.

12. As defined in the notation section, we say that $\boldsymbol{A} \approx_\kappa \boldsymbol{B}$ if and only if $\frac{1}{\kappa}\boldsymbol{B} \preceq \boldsymbol{A} \preceq \kappa\boldsymbol{B}$.

References

Zeyuan Allen Zhu. Katyusha: the first direct acceleration of stochastic gradient methods. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 1200–1205, 2017. doi: 10.1145/3055399.3055448. URL http://doi.acm.org/10.1145/3055399.3055448.

Zeyuan Allen-Zhu and Elad Hazan. Optimal black-box reductions between optimization objectives. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 1614–1622, USA, 2016. Curran Associates Inc. ISBN 978-1-5108-3881-9. URL http://dl.acm.org/citation.cfm?id=3157096.3157277.

Sébastien Bubeck, Michael B Cohen, Yin Tat Lee, and Yuanzhi Li. An homotopy method for ℓ_p regression provably beyond self-concordance and in input-sparsity time. *arXiv preprint arXiv:1711.01328*, 2017. Available at: https://arxiv.org/abs/1711.01328.

Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. *SIAM Rev.*, 43(1):129–159, January 2001. ISSN 0036-1445. doi: 10.1137/S003614450037906X. URL http://dx.doi.org/10.1137/S003614450037906X.

Hui Han Chin, Aleksander Madry, Gary L. Miller, and Richard Peng. Runtime guarantees for regression problems. In *Proceedings of the 4th Conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 269–282, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1859-4. doi: 10.1145/2422436.2422469. URL http://doi.acm.org/10.1145/2422436.2422469.

Kenneth L. Clarkson. Subgradient and sampling algorithms for ℓ_1 regression. In *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '05, pages 257–266, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics. ISBN 0-89871-585-7. URL http://dl.acm.org/citation.cfm?id=1070432.1070469.

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. In *Proceedings of the Twenty-fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '13, pages 466–477, Philadelphia, PA, USA, 2013. Society for Industrial and Applied Mathematics. ISBN 978-1-611972-51-1. URL http://dl.acm.org/citation.cfm?id=2627817.2627851.

Michael B. Cohen and Richard Peng. ℓ_p row sampling by lewis weights. In *Proceedings of the Forty-seventh Annual ACM Symposium on Theory of Computing*, STOC '15, pages 183–192, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3536-2. doi: 10.1145/2746539.2746567. URL http://doi.acm.org/10.1145/2746539.2746567.

Michael B. Cohen, Yin Tat Lee, Cameron Musco, Christopher Musco, Richard Peng, and Aaron Sidford. Uniform sampling for matrix approximation. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, ITCS '15, pages 181–190, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3333-7. doi: 10.1145/2688073.2688113. URL http://doi.acm.org/10.1145/2688073.2688113.

Anirban Dasgupta, Petros Drineas, Boulos Harb, Ravi Kumar, and Michael W. Mahoney. Sampling algorithms and coresets for ℓ_p regression. *SIAM J. Comput.*, 38(5):2060–2078, February 2009. ISSN 0097-5397. doi: 10.1137/070696507. URL http://dx.doi.org/10.1137/070696507.

A. M. Davie and A. J. Stothers. Improved bound for complexity of matrix multiplication. *Proceedings of the Royal Society of Edinburgh: Section A Mathematics*, 143(2):351–369, 2013. doi: 10.1017/S0308210511001648.

John C Duchi, Peter L Bartlett, and Martin J Wainwright. Randomized smoothing for stochastic optimization. *SIAM Journal on Optimization*, 22(2):674–701, 2012.

Martín Farach-Colton and Meng-Tsung Tsai. Exact sublinear binomial sampling. *Algorithmica*, 73(4):637–651, December 2015. ISSN 0178-4617. doi: 10.1007/s00453-015-0077-8. URL http://dx.doi.org/10.1007/s00453-015-0077-8.

F. G. Foster. On the stochastic matrices associated with certain queuing processes. *The Annals of Mathematical Statistics*, 24(3):355–360, 1953. ISSN 00034851. URL http://www.jstor.org/stable/2236286.

Nick Harvey. Matrix concentration and sparsification. Workshop on "Randomized Numerical Linear Algebra (RandNLA): Theory and Practice", 2012. Available at: http://www.drineas.org/RandNLA/slides/Harvey_RandNLA@FOCS_2012.pdf.

François Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation*, ISSAC '14, pages 296–303, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2501-1. doi: 10.1145/2608628.2608664. URL http://doi.acm.org/10.1145/2608628.2608664.

Yin Tat Lee and Aaron Sidford. Efficient inverse maintenance and faster algorithms for linear programming. In *Proceedings of the 2015 IEEE 56th Annual Symposium on Foundations of Computer Science (FOCS)*, FOCS '15, pages 230–249, Washington, DC, USA, 2015. IEEE Computer Society. ISBN 978-1-4673-8191-8. doi: 10.1109/FOCS.2015.23. URL http://dx.doi.org/10.1109/FOCS.2015.23.

D. Lewis. Finite dimensional subspaces of ℓ_p. *Studia Mathematica*, 63(2):207–212, 1978. URL http://eudml.org/doc/218208.

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*, STOC '13, pages 91–100, New York, NY, USA, 2013a. ACM. ISBN 978-1-4503-2029-0. doi: 10.1145/2488608.2488621. URL http://doi.acm.org/10.1145/2488608.2488621.

Xiangrui Meng and Michael W. Mahoney. Robust regression on mapreduce. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28*, ICML'13, pages III–888–III–896. JMLR.org, 2013b. URL http://dl.acm.org/citation.cfm?id=3042817.3043036.

A. Nemirovski, A. Juditsky, G. Lan, and A. Shapiro. Robust stochastic approximation approach to stochastic programming. *SIAM J. on Optimization*, 19(4):1574–1609, January 2009. ISSN 1052-6234. doi: 10.1137/070704277. URL http://dx.doi.org/10.1137/070704277.

Yu. Nesterov and J. Ph. Vial. Confidence level solutions for stochastic programming. *Automatica*, 44(6):1559–1568, June 2008. ISSN 0005-1098. doi: 10.1016/j.automatica.2008.01.017. URL http://dx.doi.org/10.1016/j.automatica.2008.01.017.

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

Yurii Nesterov. Excessive gap technique in nonsmooth convex minimization. *SIAM Journal on Optimization*, 16(1):235–249, May 2005a. ISSN 1052-6234. doi: 10.1137/S1052623403422285. URL http://dx.doi.org/10.1137/S1052623403422285.

Yurii Nesterov. Smooth minimization of non-smooth functions. *Mathematical Programming*, 103 (1):127–152, May 2005b. ISSN 0025-5610. doi: 10.1007/s10107-004-0552-5. URL http://dx.doi.org/10.1007/s10107-004-0552-5.

Yurii Nesterov. Smoothing technique and its applications in semidefinite optimization. *Mathematical Programming*, 110(2):245–259, March 2007. ISSN 0025-5610. doi: 10.1007/s10107-006-0001-8. URL http://dx.doi.org/10.1007/s10107-006-0001-8.

Yurii Nesterov. Unconstrained convex minimization in relative scale. *Mathematics of Operations Research*, 34(1):180–193, February 2009. ISSN 0364-765X. doi: 10.1287/moor.1080.0348. URL http://dx.doi.org/10.1287/moor.1080.0348.

Hua Ouyang and Alexander Gray. Stochastic smoothing for nonsmooth minimizations: Accelerating SGD by exploiting structure. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, pages 1523–1530, USA, 2012. Omnipress. ISBN 978-1-4503-1285-1. URL http://dl.acm.org/citation.cfm?id=3042573.3042768.

Stephen Portnoy. On computation of regression quantiles: Making the laplacian tortoise faster. *Lecture Notes-Monograph Series*, 31:187–200, 1997. ISSN 07492170. URL http://www.jstor.org/stable/4355977.

Stephen Portnoy and Roger Koenker. The gaussian hare and the laplacian tortoise: computability of squared-error versus absolute-error estimators. *Statistical Science*, 12(4):279–300, 11 1997. doi: 10.1214/ss/1030037960. URL http://dx.doi.org/10.1214/ss/1030037960.

Andrzej Ruszczynski and Wojciech Syski. On convergence of the stochastic subgradient method with on-line stepsize rules. *Journal of Mathematical Analysis and Applications*, 114(2):512 – 527, 1986. ISSN 0022-247X. doi: http://dx.doi.org/10.1016/0022-247X(86)90104-6. URL http://www.sciencedirect.com/science/article/pii/0022247X86901046.

Sushant Sachdeva and Nisheeth K. Vishnoi. Faster algorithms via approximation theory. *Foundations and Trends® in Theoretical Computer Science*, 9(2):125–210, March 2014. ISSN 1551-305X. doi: 10.1561/0400000065. URL http://dx.doi.org/10.1561/0400000065.

Christian Sohler and David P. Woodruff. Subspace embeddings for the ℓ_1-norm with applications. In *Proceedings of the Forty-third Annual ACM Symposium on Theory of Computing*, STOC '11, pages 755–764, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0691-1. doi: 10.1145/1993636.1993736. URL http://doi.acm.org/10.1145/1993636.1993736.

Joel A. Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of Computational Mathematics*, 12(4):389–434, August 2012. ISSN 1615-3375. doi: 10.1007/s10208-011-9099-z. URL http://dx.doi.org/10.1007/s10208-011-9099-z. Available at http://arxiv.org/abs/1004.4389.

Virginia Vassilevska Williams. Multiplying matrices faster than coppersmith-winograd. In *Proceedings of the Forty-fourth Annual ACM Symposium on Theory of Computing*, STOC '12, pages 887–898, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1245-5. doi: 10.1145/2213977.2214056. URL http://doi.acm.org/10.1145/2213977.2214056.

David P. Woodruff and Qin Zhang. Subspace embeddings and ℓ_p-regression using exponential random variables. In *COLT 2013 - The 26th Annual Conference on Learning Theory, June 12-14, 2013, Princeton University, NJ, USA*, pages 546–567, 2013. URL http://jmlr.org/proceedings/papers/v30/Woodruff13.html.

Jiyan Yang, Yin-Lam Chow, Christopher Ré, and Michael W. Mahoney. Weighted SGD for ℓ_p regression with randomized preconditioning. In *Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, pages 558–569, Philadelphia, PA, USA, 2016. Society for Industrial and Applied Mathematics. ISBN 978-1-611974-33-1. URL http://dl.acm.org/citation.cfm?id=2884435.2884476.

Appendix A. Proof of Theorem 1

In this section, we prove our primary result, stated in Theorem 1. Specifically, we further examine whether our strong initialization distance bound will allow us to improve the running time with black-box accelerated stochastic gradient descent methods. The first step towards this is to apply smoothing reductions to our objective function.

A.1. Smoothing the Objective Function and Adding Strong Convexity

As before, we let $\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2$. For clarity, we will borrow some of the notation from Allen-Zhu and Hazan (2016) to more clearly convey their black-box reductions.

Definition 21 *A function $f(x)$ is (L, σ)-smooth-sc if it is both L-smooth and σ-strongly-convex.*

Definition 22 *An algorithm $\mathcal{A}(f(x), x_0)$ is a $\textsc{Time}_{\mathcal{A}}(L, \sigma)$-minimizer if $f(x)$ is (L, σ)-smooth-sc and $\textsc{Time}_{\mathcal{A}}(L, \sigma)$ is the time it takes $\mathcal{A}$ to produce x' such that $f(x') - f(x^*) \leq \frac{f(x_0) - f(x^*)}{4}$ for any starting x_0.*

Allen-Zhu and Hazan assume access to efficient $\textsc{Time}_{\mathcal{A}}(L, \sigma)$-*minimizer* algorithms, and show how a certain class of objective functions can be slightly altered to meet the smoothness and strong convexity conditions to apply these algorithms without losing too much in terms of error and runtime.

Theorem 23 (Theorem C.2 from Allen-Zhu and Hazan (2016)) *Consider the problem of minimizing an objective function*

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$$

such that each $f_i(\cdot)$ is G-Lipschitz continuous. Let $\boldsymbol{x}_0$ be a starting vector such that $f(\boldsymbol{x}_0) - f(\boldsymbol{x}^*) \leq \Delta$ and $\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_2^2 \leq \Theta$. Then there is a routine that takes as input a $\textsc{Time}_{\mathcal{A}}(L,\sigma)$-minimizer, $\mathcal{A}$, alongside $f(x)$ and x_0, with $\beta_0 = \Delta/G^2$, $\sigma_0 = \Delta/\Theta$ and $T = \log_2(\Delta/\epsilon)$, and produces x_T satisfying $f(x_T) - f(x^*) \leq O(\epsilon)$ in total running time

$$\sum_{t=0}^{T-1} \textsc{Time}_{\mathcal{A}}(2^t/\beta_0, \sigma_0 \cdot 2^{-t}).$$

It is then straightforward to show that our objective function fits the necessary conditions to utilize Theorem 23.

Lemma 24 *If $\boldsymbol{A}$ is IRB, then the function $\|\boldsymbol{Ax} - \boldsymbol{b}\|_1$ can be written as $\frac{1}{n}\sum_{i=1}^{n} f_i(\boldsymbol{x})$ such that each $f_i(\cdot)$ is $O(\sqrt{nd})$-Lipschitz continuous.*

Proof By the definition of 1-norm,

$$\|\boldsymbol{Ax} - \boldsymbol{b}\|_1 = \sum_{i=1}^{n} |\boldsymbol{A}_{i,:}\boldsymbol{x} - \boldsymbol{b}_i|.$$

We then set $f_i(\boldsymbol{x}) = n \cdot |\boldsymbol{A}_{i,:}\boldsymbol{x} - \boldsymbol{b}_i|$ and the result follows from Lemma 15. ■

We can then incorporate our objective into the routine from Theorem 23, along with our initialization of $\boldsymbol{x}_0$.

Lemma 25 *Let $\mathcal{A}$ be a $\textsc{Time}_{\mathcal{A}}(L,\sigma)$-minimizer, along with objective $\|\boldsymbol{Ax} - \boldsymbol{b}\|_1$ such that $\boldsymbol{A}$ is IRB and $\boldsymbol{x}_0 = \boldsymbol{A}^T\boldsymbol{b}$, then the routine from Theorem 23 produces $\boldsymbol{x}_T$ satisfying $f(\boldsymbol{x}_T) - f(\boldsymbol{x}^*) \leq O(\epsilon)$ in total running time*

$$\sum_{t=0}^{T-1} \textsc{Time}_{\mathcal{A}}\left(O\left(\frac{nd2^t}{\Delta}\right), O\left(\frac{n\Delta}{d \cdot f(\boldsymbol{x}^*)^2 2^t}\right)\right).$$

Proof Lemma 24 implies that we can apply Theorem 23 where $G = O(\sqrt{nd})$, and Lemma 14 gives $\Theta = O(\frac{d}{n}f(x^*)^2)$. We then obtain $\beta_0 = O(\frac{\Delta}{nd})$ and $\sigma_0 = O(\frac{n\Delta}{df(x^*)^2})$ and substitute these values in the running time of Theorem 23. ■

A.2. Applying Katyusha Accelerated SGD

Now that we have shown how our initialization can be plugged into the smoothing construction of Allen-Zhu and Hazan (2016), we simply need an efficient $\textsc{Time}_{\mathcal{A}}(L,\sigma)$-*minimizer* to obtain all the necessary pieces to prove our primary result.

Theorem 26 (Corollary 3.8 in Allen Zhu (2017)) *There is a routine that is a* $\text{TIME}_{\mathcal{A}}(L,\sigma)$*-minimizer where* $\text{TIME}_{\mathcal{A}}(L,\sigma) = d \cdot O(n + \sqrt{nL/\sigma})$.

We can then precondition the matrix to give our strong bounds on the initialization distance of $\boldsymbol{x}_0$ from the optimal $\boldsymbol{x}^*$, which allows us to apply the smoothing reduction and Katyusha accelerated gradient descent more efficiently.

Theorem 1 *Given* $\boldsymbol{A} \in \mathbb{R}^{n\times d}$, $\boldsymbol{b} \in \mathbb{R}^n$, *assume* $\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2$ *is either 0 or bounded below by* n^{-c} *for some constant* $c > 0$. *Then for any* $\epsilon > 0$, *there is a routine that outputs* $\tilde{\boldsymbol{x}}$ *such that with high probability,*[13]

$$\|\boldsymbol{A}\tilde{\boldsymbol{x}} - \boldsymbol{b}\|_1 \leq (1+\epsilon)\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_1$$

with a runtime of $O\left(nnz(\boldsymbol{A})\log^2 n + d^{2.5}\epsilon^{-2}\log^{1.5} n\right)$ *whenever* $n \geq d\epsilon^{-2}\log n$, *and a runtime of* $O\left(\sqrt{n}d^2\epsilon^{-1}\log n + nd^{\omega-1}\log n\right)$ *when* $n \leq d\epsilon^{-2}\log n$.

Proof Once again, by preconditioning with Lemma 12 and error $O(\epsilon)$ we obtain a matrix $\tilde{\boldsymbol{A}}\boldsymbol{U} \in \mathbb{R}^{N\times d}$ and a vector $\tilde{\boldsymbol{b}} \in \mathbb{R}^n$ in time $O(nnz(\boldsymbol{A})\log n + d^{\omega-1}\min\{d\epsilon^{-2}\log n, n\})$. We utilize the routine in Theorem 26 as the $\text{TIME}_{\mathcal{A}}(L,\sigma)$*-minimizer* for Lemma 25, and plug the time bounds in to achieve an absolute error of $O(\delta)$ in the preconditioned objective function with the following running time:

$$\begin{aligned} d\cdot O\left(\sum_{t=0}^{T-1} N + \sqrt{N\cdot\left(\frac{Nd2^t}{\Delta}\right)\left(\frac{d\cdot f(x^*)^2 2^t}{N\Delta}\right)}\right) &= d\cdot O\left(N\log\frac{\Delta}{\delta} + \frac{d\sqrt{N}\cdot f(x^*)}{\Delta}\sum_{t=0}^{T-1} 2^t\right) \\ &= d\cdot O\left(N\log\frac{\Delta}{\delta} + \frac{d\sqrt{N}\cdot f(x^*)}{\delta}\right). \end{aligned}$$

To achieve our desired relative error of ϵ we need to set $\delta = O(\epsilon f(\boldsymbol{x}^*))$. Technically, this means that the input to gradient descent will require at least a constant factor approximation to $f(\boldsymbol{x}^*)$. We will show in Appendix D.2 that we can assume that we have such an approximation at the cost of a factor of $\log n$ in the running time. We assume that $f(\boldsymbol{x}^*) \geq n^{-c}$ for some fixed constant[14] c in order to upper bound $\log\frac{\Delta}{\delta}$, which gives a runtime of $O\left(dN\log(n\epsilon^{-1}) + \frac{d\sqrt{N}\cdot}{\epsilon}\right)$.

Here, we used the fact that $N = O(d\epsilon^{-2}\log n)$, but can also assume that computationally, $N \leq n$, as will be addressed in Appendix D.1. This gives a runtime of

$$O\left(\min\{d^{2.5}\epsilon^{-2}\sqrt{\log n}, nd\log(n/\epsilon) + \sqrt{n}d^2\epsilon^{-1}\}\right),$$

which, combined with our preconditioning runtime (where Υ is a lower order term if we assume the ϵ is at most polynomially small in n) and the factor $\log n$ overhead from estimating $f(\boldsymbol{x}^*)$ which we address in Appendix D.2, gives the desired runtime. Furthermore, since the error in our preconditioning was $O(\epsilon)$, by Lemma 12 we have achieved a solution with $O(\epsilon)$ relative error in the original problem. ■

13. Throughout this paper, we let "with high probability" mean that ξ is the failure probability and our runtime has dependence on $\log(1/\xi)$, which we ignore for ease of notation.
14. Note that if $f(\boldsymbol{x}^*) = 0$, then our initialization $\boldsymbol{x}_0 = \boldsymbol{A}^T\boldsymbol{b}$ will be equal to $\boldsymbol{x}^*$.

Appendix B. Preconditioning with Lewis Weights and Rotation

In this section, we show how to precondition a given matrix $\boldsymbol{A} \in \mathbb{R}^{n \times d}$ into a "good" matrix, primarily using techniques by Cohen and Peng (2015), and will ultimately prove Lemma 12. Recall that our overall goal was to efficiently transform $\boldsymbol{A}$ into a matrix $\tilde{\boldsymbol{A}}$ such that the ℓ_1 norm is approximately maintained for all $\boldsymbol{x}$, along with $\tilde{\boldsymbol{A}}$ being isotropic and having all row norms approximately equal.

Accordingly, our preconditioning will be done in the following two primary steps:

1. We sample $N = O(d\epsilon^{-2} \log d)$ rows from $\boldsymbol{A}$ according to Lewis weights (Cohen and Peng, 2015) to construct a matrix $\tilde{\boldsymbol{A}} \in \mathbb{R}^{N \times d}$. The guarantees of Cohen and Peng (2015) ensure that for all $x \in \mathbb{R}^d$, $||\tilde{\boldsymbol{A}}x||_1 \approx_{1+\epsilon} \|\boldsymbol{A}x\|_1$ with high probability. We then further show that this sampling scheme gives $\tau_i(\tilde{\boldsymbol{A}}) = O(d/N)$ for all $1 \leq i \leq N$ with high probability.

2. We then find an invertible matrix $\boldsymbol{U}$ such that $\tilde{\boldsymbol{A}}\boldsymbol{U}$ still has the two necessary properties from Lewis weight sampling and is also isotropic.

The matrix $\tilde{\boldsymbol{A}}\boldsymbol{U}$ then has all the prerequisite properties to run our ℓ_1 minimization algorithms, and it only becomes necessary to show that running an ℓ_1-minimization routine on $\tilde{\boldsymbol{A}}\boldsymbol{U}$ will help us find an approximate solution to the original problem.

In Appendix B.1, we show that Lewis weight sampling gives a matrix with approximately equal leverage scores. In Appendix B.2, we find the invertible matrix $\boldsymbol{U}$ that makes $\tilde{\boldsymbol{A}}\boldsymbol{U}$ isotropic while preserving other properties. In Appendix B.3, we show that an approximate solution with respect to the preconditioned matrix will give an approximate solution with respect to the original matrix. Finally, we prove our primary preconditioning result, Lemma 12, in Appendix B.4.

Before we do this, the following facts are useful.

Fact 27 (Foster's theorem (Foster, 1953)) *For a matrix $\boldsymbol{A} \in \mathbb{R}^{n \times d}$,*

$$\sum_{i=1}^{n} \tau_i(\boldsymbol{A}) = d.$$

Fact 28 (Lemma 2 in Cohen et al. (2015)) *Given a matrix $\boldsymbol{A}$, for all rows i,*

$$\tau_i(\boldsymbol{A}) = \min_{\boldsymbol{A}^T \boldsymbol{x} = \boldsymbol{A}_{i,:}^T} \|\boldsymbol{x}\|_2^2 .$$

B.1. Lewis Weight Sampling gives Approximately Equal Leverage Scores

In this section, we prove that sampling according to Lewis weights gives a matrix with approximately equal leverage scores. This proof will largely rely on showing that, up to row rescaling, the sampled matrix $\tilde{\boldsymbol{A}}$ is such that $\tilde{\boldsymbol{A}}^T\tilde{\boldsymbol{A}}$ is spectrally close to $\boldsymbol{A}^T\boldsymbol{A}$. This proof will boil down to a standard application of matrix concentration bounds for sampling according to leverage scores. Our primary lemma in this section will then mostly follow from the following lemma, which will be proven at the end of this section.

Lemma 29 *Given a matrix $\boldsymbol{A}$ that is sampled according to Theorem 11 with error ϵ and gives matrix $\tilde{\boldsymbol{A}}$, then*

$$\tilde{\boldsymbol{A}}^T\tilde{\boldsymbol{A}} \approx_{O(1)} \frac{1}{h(n,\epsilon)} \boldsymbol{A}^T \overline{\boldsymbol{W}}^{-1} \boldsymbol{A}$$

with high probability.

Using this, we can prove our key lemma.

Lemma 30 *Given a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times d}$ that is sampled according to Theorem 11 and gives matrix $\tilde{\boldsymbol{A}}$, then for all rows i of $\tilde{\boldsymbol{A}}$,*

$$\tau_i(\tilde{\boldsymbol{A}}) \approx_{O(1)} \frac{d}{N}$$

with high probability.

Proof Lemma 29 implies that

$$\tau_i(\tilde{\boldsymbol{A}}) = \tilde{\boldsymbol{A}}_{i,:}\left(\tilde{\boldsymbol{A}}^T\tilde{\boldsymbol{A}}\right)^{-1}\tilde{\boldsymbol{A}}_{i,:}^T \approx_{O(1)} h(n,\epsilon)\cdot\tilde{\boldsymbol{A}}_{i,:}\left(\boldsymbol{A}^T\overline{\boldsymbol{W}}^{-1}\boldsymbol{A}\right)^{-1}\tilde{\boldsymbol{A}}_{i,:}^T$$

with high probability. Theorem 11 implies that every row i of $\tilde{\boldsymbol{A}}$ is simply some row j of $\boldsymbol{A}$, scaled by $\frac{1}{\boldsymbol{p}_j}$. Therefore, for any row i of $\tilde{\boldsymbol{A}}$ we must have

$$\tau_i(\tilde{\boldsymbol{A}}) \approx_{O(1)} h(n,\epsilon)\cdot\tilde{\boldsymbol{A}}_{i,:}\left(\boldsymbol{A}^T\overline{\boldsymbol{W}}^{-1}\boldsymbol{A}\right)^{-1}\tilde{\boldsymbol{A}}_{i,:}^T = h(n,\epsilon)\cdot\frac{\boldsymbol{A}_{j,:}}{\boldsymbol{p}_j}\left(\boldsymbol{A}^T\overline{\boldsymbol{W}}^{-1}\boldsymbol{A}\right)^{-1}\frac{\boldsymbol{A}_{j,:}^T}{\boldsymbol{p}_j}.$$

From Definition 7 we have

$$\overline{\boldsymbol{w}}_j^2 = \boldsymbol{A}_{j,:}\left(\boldsymbol{A}^T\overline{\boldsymbol{W}}^{-1}\boldsymbol{A}\right)^{-1}\boldsymbol{A}_{j,:}^T$$

which along with the fact that $\boldsymbol{p}_j \approx_{O(1)} \overline{\boldsymbol{w}}_j \cdot h(n,\epsilon)$ reduces the leverage score to

$$\tau_i(\tilde{\boldsymbol{A}}) \approx_{O(1)} h(n,\epsilon)\cdot\frac{\overline{\boldsymbol{w}}_j^2}{\boldsymbol{p}_j^2} \approx_{O(1)} \frac{1}{h(n,\epsilon)}.$$

Finally Fact 27 gives us that the sum of Lewis weights must be d because they are leverage scores of $\overline{\boldsymbol{W}}^{-1/2}\boldsymbol{A}$, which implies $\frac{1}{h(n,\epsilon)} \approx_{O(1)} \frac{d}{N}$ by our definition of $N = \sum_i \boldsymbol{p}_i$. ■

It now remains to prove Lemma 29. The proof follows similarly to the proof of Lemma 4 in Cohen et al. (2015), except that their leverage score sampling scheme draws each row without replacement, and we need a fixed number of sampled rows with replacement. Accordingly, we will also use the following matrix concentration result from Harvey (2012), which is a variant of Corollary 5.2 in Tropp (2012):

Lemma 31 (Harvey (2012)) *Let $\boldsymbol{Y}_1 \ldots \boldsymbol{Y}_k$ be independent random positive semidefinite matrices of size $d\times d$. Let $\boldsymbol{Y} = \sum_{i=1}^k \boldsymbol{Y}_i$, and let $\boldsymbol{Z} = \mathbb{E}\left[\boldsymbol{Y}\right]$. If $\boldsymbol{Y}_i \preceq R\cdot\boldsymbol{Z}$ then*

$$Pr\left[\sum_{i=1}^k \boldsymbol{Y}_i \preceq (1-\epsilon)\,\boldsymbol{Z}\right] \le de^{\frac{-\epsilon^2}{2R}}$$

and

$$Pr\left[\sum_{i=1}^k \boldsymbol{Y}_i \succeq (1+\epsilon)\,\boldsymbol{Z}\right] \le de^{\frac{-\epsilon^2}{3R}}.$$

Proof [Proof of Lemma 29] First, we define $\overline{\boldsymbol{A}} \overset{\text{def}}{=} \overline{\boldsymbol{W}}^{-1/2}\boldsymbol{A}$. Then, by Definition 7, $\overline{\boldsymbol{w}}_i = \tau_i(\overline{\boldsymbol{A}})$. Since $\overline{\boldsymbol{W}}$ is the diagonal matrix of Lewis weights $\overline{\boldsymbol{w}}$, each row of $\overline{\boldsymbol{A}}$ is simply $\overline{\boldsymbol{A}}_{i,:} = \overline{\boldsymbol{w}}_i^{-1/2}\boldsymbol{A}_{i,:}$.

By construction of our random $\tilde{\boldsymbol{A}}$ in Theorem 11 we choose a row j of $\boldsymbol{A}$ with probability $\frac{\boldsymbol{p}_j}{N}$ and scale by $\frac{1}{\boldsymbol{p}_j}$. Therefore, if we let $\boldsymbol{Y}_i$ be the random variable

$$\boldsymbol{Y}_i = \begin{cases} \frac{\boldsymbol{A}_{j,:}\boldsymbol{A}_{j,:}^T}{\boldsymbol{p}_j^2}, & \text{with probability } \frac{\boldsymbol{p}_j}{N} \text{ for each } j \end{cases}$$

then,

$$\boldsymbol{Y} = \sum_{i=1}^{N} \boldsymbol{Y}_i = \sum_{i=1}^{N} \tilde{\boldsymbol{A}}_{i,:}\tilde{\boldsymbol{A}}_{i,:}^T = \tilde{\boldsymbol{A}}^T\tilde{\boldsymbol{A}}.$$

Furthermore, we can substitute $\overline{\boldsymbol{A}}_{j,:}\sqrt{\overline{\boldsymbol{w}}_i}$ for $\boldsymbol{A}_{j,:}$ and use the fact that $\boldsymbol{p}_j \approx_{O(1)} \overline{\boldsymbol{w}}_j \cdot h(n,\epsilon)$ to obtain

$$\frac{\boldsymbol{A}_{j,:}\boldsymbol{A}_{j,:}^T}{\boldsymbol{p}_j^2} \approx_{O(1)} \frac{\overline{\boldsymbol{A}}_{j,:}\overline{\boldsymbol{A}}_{j,:}^T}{\boldsymbol{p}_j \cdot h(n,\epsilon)}.$$

As a result, we have

$$\begin{aligned} \boldsymbol{Z} = \mathbb{E}\left[\sum_{i=1}^{N}\boldsymbol{Y}_i\right] &= \sum_{i=1}^{N}\mathbb{E}\left[\boldsymbol{Y}_i\right] \\ &\approx_{O(1)} \sum_{i=1}^{N}\sum_{j=1}^{n}\frac{\overline{\boldsymbol{A}}_{j,:}\overline{\boldsymbol{A}}_{j,:}^T}{N\cdot h(n,\epsilon)} \\ &= \frac{1}{h(n,\epsilon)}\sum_{j=1}^{n}\overline{\boldsymbol{A}}_{j,:}\overline{\boldsymbol{A}}_{j,:}^T = \frac{1}{h(n,\epsilon)}\boldsymbol{A}^T\overline{\boldsymbol{W}}^{-1}\boldsymbol{A}. \end{aligned}$$

In order to apply Lemma 31 we need to find R such that $\boldsymbol{Y}_i \preceq R\cdot\boldsymbol{Z}$, which by our construction of $\boldsymbol{Y}_i$ requires

$$\frac{\boldsymbol{A}_{j,:}\boldsymbol{A}_{j,:}^T}{\boldsymbol{p}_j^2} \preceq R\cdot\boldsymbol{Z}$$

for all j. We use our constant factor approximations of $\boldsymbol{Z}$ and $\frac{\boldsymbol{A}_{j,:}\boldsymbol{A}_{j,:}^T}{\boldsymbol{p}_j^2}$ to see that it also suffices to show

$$\frac{\overline{\boldsymbol{A}}_{j,:}\overline{\boldsymbol{A}}_{j,:}^T}{\boldsymbol{p}_j\cdot h(n,\epsilon)} \preceq \frac{R}{O(1)}\cdot\frac{1}{h(n,\epsilon)}\overline{\boldsymbol{A}}^T\overline{\boldsymbol{A}}.$$

Given that $\tau_j(\overline{\boldsymbol{A}}) = \overline{\boldsymbol{w}}_j$ and $\boldsymbol{p}_j \approx_{O(1)} \overline{\boldsymbol{w}}_j\cdot h(n,\epsilon)$, we have

$$\frac{\overline{\boldsymbol{A}}_{j,:}\overline{\boldsymbol{A}}_{j,:}^T}{\boldsymbol{p}_j\cdot h(n,\epsilon)} \preceq \frac{O(1)\overline{\boldsymbol{A}}_{j,:}\overline{\boldsymbol{A}}_{j,:}^T}{\tau_j(\overline{\boldsymbol{A}})\cdot h(n,\epsilon)^2}$$

which along with the fact (Equation 12 in the proof of Lemma 4 from Cohen et al. (2015)) that

$$\frac{\overline{\boldsymbol{A}}_{j,:}\overline{\boldsymbol{A}}_{j,:}^T}{\tau_j(\overline{\boldsymbol{A}})} \preceq \overline{\boldsymbol{A}}^T\overline{\boldsymbol{A}}$$

implies that

$$\boldsymbol{Y}_i \preceq \frac{O(1)}{h(n,\epsilon)}\boldsymbol{Z}.$$

By Theorem 11 we know that $h(n,\epsilon) \geq c\epsilon^{-2}\log n$ for some constant c. Plugging this in for R in Lemma 31 gives that

$$\boldsymbol{Y} \approx_{1+\epsilon} \boldsymbol{Z}$$

or, substituting our values of $\boldsymbol{Y}$ and $\boldsymbol{Z}$,

$$\tilde{\boldsymbol{A}}^T\tilde{\boldsymbol{A}} \approx_{O(1)} \frac{1}{h(n,\epsilon)}\boldsymbol{A}^T\overline{\boldsymbol{W}}^{-1}\boldsymbol{A}$$

with probability at least $1 - 2de^{-\frac{\epsilon^{-2}}{3R}} \geq 1 - 2de^{-\frac{c\log n}{O(1)}} \geq 1 - 2dn^{-c/O(1)}$. This implies that the statement in the lemma is true with high probability for c bigger than $O(1)$ (where the $O(1)$ comes from our $\boldsymbol{p}_i$ approximation of $\overline{\boldsymbol{w}}_i \cdot h(n,\epsilon)$) and our assumption on $n \geq d$. ■

B.2. Rotating the Matrix to Achieve Isotropic Position

Now that we have sampled by Lewis weights and achieved all leverage scores to be approximately equal, we will show that we can efficiently rotate the matrix into isotropic position while still preserving the fact that all leverage scores are approximately equal.

Lemma 32 *If $\boldsymbol{U} \in \mathbb{R}^{d\times d}$ is an invertible matrix and $\boldsymbol{U}^T\boldsymbol{U} = \left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-1}$ then*

1. $(\boldsymbol{A}\boldsymbol{U})^T(\boldsymbol{A}\boldsymbol{U}) = \boldsymbol{I}$.
2. *For all rows i, $\tau_i(\boldsymbol{A}) = \tau_i(\boldsymbol{A}\boldsymbol{U})$.*

Proof For the first condition, we see that

$$\boldsymbol{U}^T\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{U} = \boldsymbol{I} \iff \boldsymbol{A}^T\boldsymbol{A} = (\boldsymbol{U}^T)^{-1}\boldsymbol{U}^{-1} \iff (\boldsymbol{A}^T\boldsymbol{A})^{-1} = \boldsymbol{U}^T\boldsymbol{U}.$$

For the second condition, the ith row of $\boldsymbol{A}\boldsymbol{U}$ will be $\boldsymbol{A}_{i,:}\boldsymbol{U}$, which by the definition of leverage scores then gives,

$$\begin{aligned}
\tau_i(\boldsymbol{A}\boldsymbol{U}) &= \boldsymbol{A}_{i,:}\boldsymbol{U}\left((\boldsymbol{A}\boldsymbol{U})^T(\boldsymbol{A}\boldsymbol{U})\right)^{-1}(\boldsymbol{A}_{i,:}\boldsymbol{U})^T \\
&= \boldsymbol{A}_{i,:}\boldsymbol{U}\boldsymbol{U}^{-1}\left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-1}(\boldsymbol{U}^T)^{-1}\boldsymbol{U}^T\boldsymbol{A}_{i,:}^T \\
&= \boldsymbol{A}_{i,:}\left(\boldsymbol{A}^T\boldsymbol{A}\right)^{-1}\boldsymbol{A}_{i,:}^T \\
&= \tau_i(\boldsymbol{A}).
\end{aligned}$$

■

It is clear then that we want to rotate our matrix by $\boldsymbol{U}$ as above, so it only remains to efficiently compute such a $\boldsymbol{U}$.

Lemma 33 *Given a full rank matrix $\tilde{\boldsymbol{A}} \in \mathbb{R}^{N\times d}$, there is a routine* ROTATE *that can find an invertible $\boldsymbol{U}$ such that $\boldsymbol{U}^T\boldsymbol{U} = \left(\tilde{\boldsymbol{A}}^T\tilde{\boldsymbol{A}}\right)^{-1}$ in time $O(Nd^{\omega-1} + d^\omega)$.*

Proof Computing $\tilde{\boldsymbol{A}}^T\tilde{\boldsymbol{A}}$ can be done in $O(Nd^{\omega-1})$ time using fast matrix multiplication. Inverting $\tilde{\boldsymbol{A}}^T\tilde{\boldsymbol{A}}$, a $d \times d$ matrix that must have an inverse because $\tilde{\boldsymbol{A}}$ is full rank, will require $O(d^\omega)$ time. Finally, we perform a QR-decomposition of $\left(\tilde{\boldsymbol{A}}^T\tilde{\boldsymbol{A}}\right)^{-1}$ in $O\left(d^\omega\right)$ time to obtain our square invertible matrix $\boldsymbol{U}$.[15] ■

Lastly, we want to ensure that by rotating our matrix, we can still use an approximate solution to the rotated matrix to obtain an approximate solution of the original matrix.

Lemma 34 *Given a matrix-vector pair $\boldsymbol{A} \in \mathbb{R}^{n\times d}, \boldsymbol{b} \in \mathbb{R}^n$, another matrix-vector pair $\tilde{\boldsymbol{A}} \in \mathbb{R}^{N\times d}$, $\tilde{\boldsymbol{b}} \in \mathbb{R}^N$, and an invertible matrix $\boldsymbol{U} \in \mathbb{R}^{d\times d}$,*

$$\|[\boldsymbol{A},\boldsymbol{b}]\boldsymbol{x}\|_1 \approx_{1+\epsilon} \|[\tilde{\boldsymbol{A}},\tilde{\boldsymbol{b}}]\boldsymbol{x}\|_1 \forall \boldsymbol{x} \in \mathbb{R}^{d+1} \iff \|[\boldsymbol{A}\boldsymbol{U},\boldsymbol{b}]\boldsymbol{y}\|_1 \approx_{1+\epsilon} \|[\tilde{\boldsymbol{A}}\boldsymbol{U},\tilde{\boldsymbol{b}}]\boldsymbol{y}\|_1 \forall \boldsymbol{y} \in \mathbb{R}^{d+1}.$$

Proof This follows immediately from the fact that for any $\boldsymbol{x}$ satisfying the LHS, there exists a $\boldsymbol{y}$ satisfying the RHS, and vice versa. Specifically $\boldsymbol{y}_{[1,d]} = \boldsymbol{U}^{-1}\boldsymbol{x}_{[1,d]}$ and $\boldsymbol{y}_{d+1} = \boldsymbol{x}_{d+1}$, and equivalently $\boldsymbol{U}\boldsymbol{y}_{[1,d]} = \boldsymbol{x}_{[1,d]}$ and $\boldsymbol{y}_{d+1} = \boldsymbol{x}_{d+1}$. ■

B.3. Translating between Preconditioned and Original Matrix Solutions

Our preconditioning combination of Lewis weights and rotating the matrix gives our desired conditions, specifically an IRB matrix, but it remains to be seen that we can take a solution to this preconditioned matrix and translate it back into an approximate solution of the original matrix. In the following lemma we will show that this is in fact true.

Lemma 35 *Given a matrix-vector pair $\boldsymbol{A} \in \mathbb{R}^{n\times d}, \boldsymbol{b} \in \mathbb{R}^n$, another matrix-vector pair $\tilde{\boldsymbol{A}} \in \mathbb{R}^{N\times d}$, $\tilde{\boldsymbol{b}} \in \mathbb{R}^N$, and an invertible matrix $\boldsymbol{U} \in \mathbb{R}^{d\times d}$; if*

$$\left\|[\tilde{\boldsymbol{A}}\ \tilde{\boldsymbol{b}}]\boldsymbol{y}\right\|_1 \approx_{1+\epsilon} \left\|[\boldsymbol{A}\ \boldsymbol{b}]\boldsymbol{y}\right\|_1$$

for all $\boldsymbol{y} \in \mathbb{R}^{d+1}$, and if $\tilde{\boldsymbol{x}}^*_U$ minimizes $\|\tilde{\boldsymbol{A}}\boldsymbol{U}\boldsymbol{x} - \tilde{\boldsymbol{b}}\|_1$, then for any $\tilde{\boldsymbol{x}}$ such that

$$\|\tilde{\boldsymbol{A}}\boldsymbol{U}\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{b}}\|_1 \le (1+\delta)\|\tilde{\boldsymbol{A}}\boldsymbol{U}\tilde{\boldsymbol{x}}^*_U - \tilde{\boldsymbol{b}}\|_1$$

we must have

$$\|\boldsymbol{A}(\boldsymbol{U}\tilde{\boldsymbol{x}}) - \boldsymbol{b}\|_1 \le (1+\epsilon)^2(1+\delta)\|\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b}\|_1$$

with high probability.

Proof By assumption we have

$$\left\|[\tilde{\boldsymbol{A}}\ \tilde{\boldsymbol{b}}]\boldsymbol{y}\right\|_1 \approx_{1+\epsilon} \left\|[\boldsymbol{A}\ \boldsymbol{b}]\boldsymbol{y}\right\|_1$$

for all $\boldsymbol{y} \in \mathbb{R}^{d+1}$, and we can then use Lemma 34 to obtain

$$\left\|[\tilde{\boldsymbol{A}}\boldsymbol{U}\ \tilde{\boldsymbol{b}}]\boldsymbol{y}\right\|_1 \approx_{1+\epsilon} \left\|[\boldsymbol{A}\boldsymbol{U}\ \boldsymbol{b}]\boldsymbol{y}\right\|_1$$

15. For an invertible matrix $\boldsymbol{M} \in \mathbb{R}^{d\times d}$, it is easy to see that $\boldsymbol{M}(\boldsymbol{M}^T\boldsymbol{M})^{-1/2}$ is an orthonormal basis for $\boldsymbol{M}$. We can compute $(\boldsymbol{M}^T\boldsymbol{M})^{-1}$ using Schur decomposition in $O(d^\omega)$ time, and by careful analysis of that algorithm, we can also compute $(\boldsymbol{M}^T\boldsymbol{M})^{-1/2}$ in the same amount of time.

for all $\boldsymbol{y} \in \mathbb{R}^{d+1}$. By fixing $\boldsymbol{y}$ to be $\begin{pmatrix} \boldsymbol{x} \\ -1 \end{pmatrix}$, we get

$$\|\tilde{\boldsymbol{A}}\boldsymbol{x} - \tilde{\boldsymbol{b}}\|_1 \approx_{1+\epsilon} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_1 \qquad \forall\, \boldsymbol{x} \in \mathbb{R}^d, \tag{2}$$

$$\|\tilde{\boldsymbol{A}}\boldsymbol{U}\boldsymbol{x} - \tilde{\boldsymbol{b}}\|_1 \approx_{1+\epsilon} \|\boldsymbol{A}\boldsymbol{U}\boldsymbol{x} - \boldsymbol{b}\|_1 \qquad \forall\, \boldsymbol{x} \in \mathbb{R}^d. \tag{3}$$

(2) gives

$$\|\boldsymbol{A}\boldsymbol{U}\tilde{\boldsymbol{x}} - \boldsymbol{b}\|_1 \le (1+\epsilon)\|\tilde{\boldsymbol{A}}\boldsymbol{U}\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{b}}\|_1.$$

Using our initial assumption and defining $\tilde{\boldsymbol{x}}^* \stackrel{\text{def}}{=} \boldsymbol{U}\tilde{\boldsymbol{x}}^*_{\boldsymbol{U}}$ then gives us

$$\|\boldsymbol{A}\boldsymbol{U}\tilde{\boldsymbol{x}} - \boldsymbol{b}\|_1 \le (1+\epsilon)(1+\delta)\|\tilde{\boldsymbol{A}}\tilde{\boldsymbol{x}}^* - \tilde{\boldsymbol{b}}\|_1.$$

Notice that if $\tilde{\boldsymbol{x}}^*_{\boldsymbol{U}}$ minimizes $\|\tilde{\boldsymbol{A}}\boldsymbol{U}\boldsymbol{x} - \tilde{\boldsymbol{b}}\|_1$, then $\tilde{\boldsymbol{x}}^*$ must minimize $\|\tilde{\boldsymbol{A}}\boldsymbol{x} - \tilde{\boldsymbol{b}}\|_1$ because $\boldsymbol{U}$ is invertible. Therefore, $\|\tilde{\boldsymbol{A}}\tilde{\boldsymbol{x}}^* - \tilde{\boldsymbol{b}}\|_1 \le \|\tilde{\boldsymbol{A}}\boldsymbol{x}^* - \tilde{\boldsymbol{b}}\|_1$ and we have

$$\|\boldsymbol{A}\boldsymbol{U}\tilde{\boldsymbol{x}} - \boldsymbol{b}\|_1 \le (1+\epsilon)(1+\delta)\|\tilde{\boldsymbol{A}}\boldsymbol{x}^* - \tilde{\boldsymbol{b}}\|_1.$$

Finally, applying (3) gives

$$\|\boldsymbol{A}\boldsymbol{U}\tilde{\boldsymbol{x}} - \boldsymbol{b}\|_1 \le (1+\epsilon)^2(1+\delta)\|\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b}\|_1.$$

■

B.4. Proof of Lemma 12

We now have all the necessary pieces to prove our primary preconditioning lemma, which we will now restate and prove.

Lemma 36 *There is a routine that takes a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times d}$, a vector $\boldsymbol{b} \in \mathbb{R}^n$ and $\epsilon > 0$, then produces a matrix $[\tilde{\boldsymbol{A}}, \tilde{\boldsymbol{b}}] \in \mathbb{R}^{N\times(d+1)}$ with $N = O(d\epsilon^{-2}\log n)$ and an invertible matrix $\boldsymbol{U} \in \mathbb{R}^{d\times d}$ such that matrix $\tilde{\boldsymbol{A}}\boldsymbol{U}$ is IRB and if $\tilde{\boldsymbol{x}}^*_{\boldsymbol{U}}$ minimizes $\|\tilde{\boldsymbol{A}}\boldsymbol{U}\boldsymbol{x} - \tilde{b}\|_1$, then for any $\tilde{\boldsymbol{x}}$ such that*

$$\|\tilde{\boldsymbol{A}}\boldsymbol{U}\tilde{\boldsymbol{x}} - \tilde{\boldsymbol{b}}\|_1 \le (1+\delta)\|\tilde{\boldsymbol{A}}\boldsymbol{U}\tilde{\boldsymbol{x}}^*_{\boldsymbol{U}} - \tilde{\boldsymbol{b}}\|_1,$$

holds for some $\delta > 0$, we must have

$$\|\boldsymbol{A}(\boldsymbol{U}\tilde{\boldsymbol{x}}) - \boldsymbol{b}\|_1 \le (1+\epsilon)^2(1+\delta)\|\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b}\|_1$$

with high probability.

Furthermore, the full running time is $O(nnz(\boldsymbol{A})\log n + d^{\omega-1}\min\{d\epsilon^{-2}\log n, n\} + \Upsilon)$ where $\Upsilon = \min\{d\epsilon^{-2}\log n, (d\epsilon^{-2}\log n)^{1/2+o(1)} + n\log^2 n\}$.

Proof From Theorem 11 we have that

$$\left\|[\tilde{\boldsymbol{A}}\ \tilde{\boldsymbol{b}}]\boldsymbol{y}\right\|_1 \approx_{1+\epsilon} \left\|[\boldsymbol{A}\ \boldsymbol{b}]\boldsymbol{y}\right\|_1$$

for all $\boldsymbol{y} \in \mathbb{R}^{d+1}$ with high probability. Lemma 35 then gives

$$\|\boldsymbol{A}(\boldsymbol{U}\tilde{\boldsymbol{x}}) - \boldsymbol{b}\|_1 \leq (1+\epsilon)^2(1+\delta)\,\|\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b}\|_1$$

by our assumption on $\tilde{x}$.

Lemma 29 and the assumption that $\boldsymbol{A}$ is full rank imply that $\tilde{\boldsymbol{A}}$ is full rank with high probability. Our use of Rotate to generate $\boldsymbol{U}$, such that $\boldsymbol{U}^T\boldsymbol{U} = (\tilde{\boldsymbol{A}}^T\tilde{\boldsymbol{A}})^{-1}$, along with Lemma 32, gives $(\tilde{\boldsymbol{A}}\boldsymbol{U})^T\tilde{\boldsymbol{A}}\boldsymbol{U} = \boldsymbol{I}$ and also that $\tau_i(\tilde{\boldsymbol{A}}\boldsymbol{U}) = \tau_i(\tilde{\boldsymbol{A}})$ for all i. Fact 28 gives $\tau_i(\tilde{\boldsymbol{A}}) \leq \tau_i([\tilde{\boldsymbol{A}}\ \tilde{\boldsymbol{b}}])$, which along with Lemma 30, implies $\tau_i(\tilde{\boldsymbol{A}}\boldsymbol{U}) \leq O(d/N)$ for all i. Finally, by Definition 6 and the fact that $(\tilde{\boldsymbol{A}}\boldsymbol{U})^T\tilde{\boldsymbol{A}}\boldsymbol{U} = \boldsymbol{I}$, we then have

$$\tau_i(\tilde{\boldsymbol{A}}\boldsymbol{U}) = \left\| \left(\tilde{\boldsymbol{A}}\boldsymbol{U}\right)_{i,:} \right\|_2.$$

The sampling of $\boldsymbol{A}$ is done according to the technique by Cohen and Peng (2015), which requires $O(nnz(\boldsymbol{A})\log n + d^\omega)$ time to obtain the sampling probabilities. Then the actual sampling requires $O(\min\{d\epsilon^{-2}\log n, (d\epsilon^{-2}\log n)^{1/2+o(1)} + n\log^2 n\})$-time according to Corollary 46 shown in Appendix D.1. Computing the invertible matrix $\boldsymbol{U}$ for input $\tilde{\boldsymbol{A}}$ takes $O(Nd^{\omega-1} + d^\omega)$ time from Lemma 33, and the number of rows of $\tilde{\boldsymbol{A}}$ is $N = O(d\epsilon^{-2}\log n)$. Finally, Lemma 42 and Corollary 43 in Appendix D.1 show that this computation time can also be bounded with $N \leq n$, which then gives our desired runtime. ■

Appendix C. Proofs from Section 4

In this section we provide the omitted proofs from Section 4.

C.1. Proof of Lemma 17

In this section we reduce the number of rows in $\boldsymbol{A}$ by uniform sampling while still preserving certain guarantees. Note that we will ultimately sample from $[\boldsymbol{A}\ \boldsymbol{b}]$, but for simplicity in notation, we will just use $\boldsymbol{A}$ here.

To prove Lemma 17, we need the following lemma, which states the key fact that the conditions on $\boldsymbol{A}$ ensure approximately uniform Lewis weights.

Lemma 37 (Almost-uniform leverage scores imply almost-uniform Lewis weights) *Consider a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times d}$ such that $\boldsymbol{A}^T\boldsymbol{A} \approx_{O(1)} \boldsymbol{I}$ and $\|\boldsymbol{A}_{i,:}\|_2^2 \approx_{O(1)} d/n$. Let $\overline{\boldsymbol{w}}$ denote the ℓ_1 Lewis weights for $\boldsymbol{A}$. Then for each row i, we have $\overline{\boldsymbol{w}}_i \approx_{O(1)} d/n$.*

Proof [Proof of Lemma 17] Note that by Lemma 37 we have

$$\boldsymbol{p}_i = \frac{N}{n} \approx_{O(1)} \frac{d\cdot O(\epsilon^{-2}\log n)}{n} \approx_{O(1)} \overline{\boldsymbol{w}}_i \cdot O(\epsilon^{-2}\log n).$$

Thus, if we use $\boldsymbol{p}_i = N/n$ for each i in Theorem 11, we get the first property while avoiding the cost of computing $\boldsymbol{p}_i$'s stated in Theorem 11.

The second property follows from Lemma 29 in Appendix B. Specifically, we have

$$\tilde{\boldsymbol{A}}^T\tilde{\boldsymbol{A}} \approx_{O(1)} \frac{1}{O(\epsilon^{-2}\log n)}\boldsymbol{A}^T\overline{\boldsymbol{W}}^{-1}\boldsymbol{A},$$

which then implies that

$$\tilde{\boldsymbol{A}}^T\tilde{\boldsymbol{A}} \approx_{O(1)} \frac{n}{d\cdot O(\epsilon^{-2}\log n)}\boldsymbol{A}^T\boldsymbol{A} \approx_{O(1)} \frac{n}{N}\boldsymbol{I}.$$

Let τ denote the leverage scores for $\boldsymbol{A}$. Now, for the third property, it follows from the definition of leverage scores and the second property that

$$\tau_i(\tilde{\boldsymbol{A}}) = \left\|\left(\tilde{\boldsymbol{A}}^T\tilde{\boldsymbol{A}}\right)^{-1/2}\tilde{\boldsymbol{A}}^T_{i,:}\right\|_2^2 \approx_{O(1)} \left\|\sqrt{N/n}\tilde{\boldsymbol{A}}^T_{i,:}\right\|_2^2.$$

Furthermore, Lemma 30 in Appendix B shows that $\tau_i(\tilde{\boldsymbol{A}}) \approx_{O(1)} d/N$. Factoring this into the equation gives us

$$\left\|\tilde{\boldsymbol{A}}^T_{i,:}\right\|_2^2 \approx_{O(1)} dn/N^2.$$

■

Now, to prove Lemma 37, we need the following definition and lemma.

Definition 38 (Definition 5.1 of α-almost Lewis weights for ℓ_1 from Cohen and Peng (2015)) *For a matrix $\boldsymbol{A}$, an assignment of weights $\boldsymbol{w}$ is α-almost Lewis if*

$$\boldsymbol{A}_{i,:}(\boldsymbol{A}^T\boldsymbol{W}^{-1}\boldsymbol{A})^{-1}\boldsymbol{A}^T_{:,i} \approx_\alpha \boldsymbol{w}_i^2,$$

where $\boldsymbol{W}$ is the diagonal matrix form of $\boldsymbol{w}$.

Lemma 39 (Definition 5.2 and Lemma 5.3 from Cohen and Peng (2015)) *Any set of α-almost Lewis weights satisfy*

$$\overline{\boldsymbol{w}}_i \approx_\alpha \boldsymbol{w}_i.$$

Proof [Proof of Lemma 37] We know that $\boldsymbol{A}^T\boldsymbol{A} \approx_{O(1)} \boldsymbol{I}$ and for each row i, $\|\boldsymbol{A}_{i,:}\|_2^2 \approx_{O(1)} d/n$. Then,

$$\begin{aligned}\tau_i(\boldsymbol{A}) &= \boldsymbol{A}_{i,:}(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T_{i,:} \approx_{O(1)} \boldsymbol{A}_{i,:}\boldsymbol{A}^T_{i,:}\\ \implies \tau_i(\boldsymbol{A}) &\approx_{O(1)} d/n.\end{aligned}$$

That is, all of the leverage scores are approximately equal. Then we can show that $\boldsymbol{w} = (d/n)\mathbb{1}$, where $\mathbb{1}$ is the all ones vector. Then,

$$\boldsymbol{A}_{i,:}(\boldsymbol{A}^T\boldsymbol{W}^{-1}\boldsymbol{A})^{-1}\boldsymbol{A}^T_{:,i} = (d/n)\boldsymbol{A}_{i,:}(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T_{:,i} \approx_{O(1)} d^2/n^2 = \boldsymbol{w}_i^2.$$

Thus, $\boldsymbol{w}$ is $O(1)$-almost Lewis. The result then follows by Lemma 39. ■

C.2. Proof of Lemma 19 and Lemma 20

To prove Lemma 19, we use the following lemma:

Lemma 40 *Let $\boldsymbol{v} \in \mathbb{R}^n$ be a vector with $\|\boldsymbol{v}\|_1 = 1$. Then, for a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times d}$,*

$$\left\|\boldsymbol{A}^T\boldsymbol{v}\right\|_2 \le \max_i \|\boldsymbol{A}_{i,:}\|_2 .$$

Proof

$$\left\|\boldsymbol{A}^T\boldsymbol{v}\right\|_2 = \left\|\sum_i \boldsymbol{A}_{i,:}\boldsymbol{v}_i\right\|_2 \le \max_i \|\boldsymbol{A}_{i,:}\|_2 .$$

where the inequality follows by the convexity of $\|\cdot\|_2$ and since $\sum_i |\boldsymbol{v}_i| = 1$, ■

Proof [Proof of Lemma 19] By our assumptions on $\boldsymbol{A}$, we have $\boldsymbol{A}^T\boldsymbol{A} + \boldsymbol{B} = \boldsymbol{I}$ for some symmetric $\boldsymbol{B}$ where $\|\boldsymbol{B}\|_2 \le O(1)$. Since $\boldsymbol{x}_0 = \arg\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2$, we have $\boldsymbol{x}_0 = \boldsymbol{A}^\dagger \boldsymbol{b} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b}$.

$$\begin{aligned}
\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_2 &= \left\|(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b} - \boldsymbol{x}^*\right\|_2 \\
&= \left\|(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b} - (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{A}\boldsymbol{x}^*\right\|_2 \\
&= \left\|(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T(\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^*)\right\|_2 .
\end{aligned}$$

Let $\boldsymbol{v} = (\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b})/\left\|\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b}\right\|_1$.

$$\begin{aligned}
\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_2 &= \left\|(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{v}\right\|_2 \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^*\|_1 \\
&= \left\|(\boldsymbol{A}^T\boldsymbol{A} + \boldsymbol{B})(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{v}\right\|_2 \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^*\|_1 \\
&= \left\|(\boldsymbol{I} + \boldsymbol{B}(\boldsymbol{A}^T\boldsymbol{A})^{-1})\boldsymbol{A}^T\boldsymbol{v}\right\|_2 \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^*\|_1 \\
&\le \left\|\boldsymbol{I} + \boldsymbol{B}(\boldsymbol{A}^T\boldsymbol{A})^{-1}\right\|_2 \left\|\boldsymbol{A}^T\boldsymbol{v}\right\|_2 \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^*\|_1 .
\end{aligned}$$

Now note:

$$\begin{aligned}
\left\|\boldsymbol{I} + \boldsymbol{B}(\boldsymbol{A}^T\boldsymbol{A})^{-1}\right\|_2 &\le \|\boldsymbol{I}\|_2 + \|\boldsymbol{B}\|_2 \left\|(\boldsymbol{A}^T\boldsymbol{A})^{-1}\right\|_2 \\
&\le O(1).
\end{aligned}$$

Also, by Lemma 40 and the assumptions on $\boldsymbol{A}$,

$$\left\|\boldsymbol{A}^T\boldsymbol{v}\right\|_2 \le O(\sqrt{d/n}).$$

Thus, we have:

$$\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_2 \le O(\sqrt{d/n}) \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}^*\|_1 .$$

■

To prove Lemma 20, we use the following theorem from Sachdeva and Vishnoi (2014):

Theorem 41 (Theorem 9.1 from Sachdeva and Vishnoi (2014)) *Given an symmetric positive definite matrix $\boldsymbol{M} \in \mathbb{R}^{n \times n}$ and a vector $\boldsymbol{y} \in \mathbb{R}^n$, the Conjugate Gradient method can find a vector $\boldsymbol{x}$ such that $\left\|\boldsymbol{x} - \boldsymbol{M}^{-1}\boldsymbol{y}\right\|_{\boldsymbol{M}} \leq \delta \left\|\boldsymbol{M}^{-1}\boldsymbol{y}\right\|_{\boldsymbol{M}}$ in time $O((t_{\boldsymbol{M}} + n) \cdot \sqrt{\kappa(\boldsymbol{M})} \log(1/\delta))$, where $t_{\boldsymbol{M}}$ is the time required to multiply $\boldsymbol{M}$ with a given vector and $\kappa(\boldsymbol{M})$ is the condition number of $\boldsymbol{M}$.*

Proof [Proof of Lemma 20] Let $\boldsymbol{M} = \boldsymbol{A}^T\boldsymbol{A}$ and $\boldsymbol{y} = \boldsymbol{A}^T\boldsymbol{b}$. Then by Theorem 41, the conjugate gradient method finds a vector $\tilde{\boldsymbol{x}}_0$ such that $\|\tilde{\boldsymbol{x}}_0\|_{\boldsymbol{A}^T\boldsymbol{A}} \leq \delta \left\|(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b}\right\|_{\boldsymbol{A}^T\boldsymbol{A}}$ in time $O((t_{\boldsymbol{A}^T\boldsymbol{A}} + d)\log(1/\delta))$. Noting that $\boldsymbol{A}^T\boldsymbol{A} \approx_{O(1)} \boldsymbol{I}$, we get

$$\|\tilde{\boldsymbol{x}}_0 - \boldsymbol{x}_0\|_2 \leq O(\delta)\,\|\boldsymbol{x}_0\|_2\,.$$

Next, we note:

$$\begin{aligned} &\|\boldsymbol{b}\|_2 \geq \|\boldsymbol{A}\boldsymbol{x}_0 - \boldsymbol{b}\|_2 \geq \|\boldsymbol{A}\boldsymbol{x}_0\|_2 - \|\boldsymbol{b}\|_2 \\ \implies &\|\boldsymbol{x}_0\|_2 \leq O(\|\boldsymbol{b}\|_2). \end{aligned}$$

Now, since we assume that $\|\boldsymbol{b}\|_2 \leq n^c$ and $\|\boldsymbol{A}\boldsymbol{x}_0 - \boldsymbol{b}\|_2 \geq 1/n^c$ for some c, we can set $\delta = O(\epsilon/(n^c))$ to get:

$$\|\tilde{\boldsymbol{x}}_0 - \boldsymbol{x}_0\|_2 \leq \epsilon/n^c \leq \epsilon\,\|\boldsymbol{A}\boldsymbol{x}_0 - \boldsymbol{b}\|_2\,.$$

■

C.3. Proof of Theorem 2

Theorem 2 *Let $\boldsymbol{A} \in \mathbb{R}^{n\times d}$ and $\boldsymbol{b} \in \mathbb{R}^n$ be such that the matrix-vector pair $[\boldsymbol{A}\ \ \boldsymbol{b}]$ satisfies $[\boldsymbol{A}\ \ \boldsymbol{b}]^T[\boldsymbol{A}\ \ \boldsymbol{b}] \approx_{O(1)} \boldsymbol{I}$ [16] and for each row i of $[\boldsymbol{A}\ \ \boldsymbol{b}]$, $\|[\boldsymbol{A}\ \ \boldsymbol{b}]_{i,:}\|_2^2 \approx_{O(1)} d/n$. Assume $\|\boldsymbol{b}\|_2 \leq n^c$ and $\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2$ is either 0 or bounded below by n^{-c} for some constant $c > 0$. Then for any $\epsilon > 0$, there is a routine that computes $\tilde{\boldsymbol{x}}$ such that with high probability,*

$$\|\boldsymbol{A}\tilde{\boldsymbol{x}} - \boldsymbol{b}\|_1 \leq (1+\epsilon)\min_{\boldsymbol{x}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_1$$

with a runtime of $O\left(nnz(\boldsymbol{A})\log^2 n + s \cdot d^{1.5}\epsilon^{-2}\log^{1.5} n + d^2\epsilon^{-2}\log^2 n\right)$, where s is the maximum number of entries in any row of $\boldsymbol{A}$.

Proof We first prove correctness. We apply Lemma 17 to $[\boldsymbol{A}\ \ \boldsymbol{b}]$ and rescale $[\tilde{\boldsymbol{A}}\ \ \tilde{\boldsymbol{b}}]$ by $\sqrt{N/n}$. Note that rescaling does not change the relative error of our output $\tilde{x}$. From this rescaling, we have $[\tilde{\boldsymbol{A}}\ \tilde{\boldsymbol{b}}]^T[\tilde{\boldsymbol{A}}\ \tilde{\boldsymbol{b}}] \approx_{O(1)} \boldsymbol{I}$ and thus $\tilde{\boldsymbol{A}}^T\tilde{\boldsymbol{A}} \approx_{O(1)} \boldsymbol{I}$. This implies that $\tau_i([\tilde{\boldsymbol{A}}\ \ \tilde{\boldsymbol{b}}]) \approx_{O(1)} \|[\tilde{\boldsymbol{A}}\ \ \tilde{\boldsymbol{b}}]_{i,:}\|_2^2$ and $\tau_i(\tilde{\boldsymbol{A}}) \approx_{O(1)} \|\tilde{\boldsymbol{A}}_{i,:}\|_2^2$. By Fact 28, we have $\tau_i(\tilde{\boldsymbol{A}}) \leq \tau_i([\tilde{\boldsymbol{A}}\ \ \tilde{\boldsymbol{b}}])$, so we have $\|\tilde{\boldsymbol{A}}_{i,:}\|_2^2 \leq O(d/N)$ for all rows i. As a result, we can find $\tilde{\boldsymbol{x}}_0$ according to Lemma 18. The rest of the correctness follows exactly as in the proof of Theorem 1.

We now examine the running time and note that uniform sampling will take $O(nnz(\boldsymbol{A}) + d^2\epsilon^{-2}\log n)$ time to produce $[\tilde{\boldsymbol{A}}\ \ \tilde{\boldsymbol{b}}]$. By Lemma 18, we can then find $\tilde{\boldsymbol{x}}_0$ in time $O(d^2\epsilon^{-2}\log n)$ because $\tilde{\boldsymbol{A}}$ is a $d\epsilon^{-2}\log n \times d$ matrix, so $t_{\tilde{\boldsymbol{A}}^T\tilde{\boldsymbol{A}}} = O(d^2\epsilon^{-2}\log n)$. Finally from the analysis of

16. As defined in the notation section, we say that $\boldsymbol{A} \approx_\kappa \boldsymbol{B}$ if and only if $\frac{1}{\kappa}\boldsymbol{B} \preceq \boldsymbol{A} \preceq \kappa\boldsymbol{B}$.

Theorem 1 we know that accelerated stochastic gradient descent requires $O(d^{2.5}\log^{1/2} n \cdot \epsilon^{-2})$ time. However, we note that the extra factor of d came from Theorem 26 where we substituted d for the time per iteration of stochastic gradient descent. This value can actually be upper bounded by the maximum number of entries in any row of $\tilde{\boldsymbol{A}}$, which because of our uniform sampling is upper bounded by the maximum number of entries in any row of $\boldsymbol{A}$. Adding a runtime overhead of $\log n$ for computing an approximation of the optimal objective, as in Appendix D.2, gives the desired runtime. ■

Appendix D. Secondary technical details for our main results

In this section, we address a couple assumptions that were made in the proofs of our main results. These assumptions were minor details, but we now include proofs for completeness. First, we always assumed that our row dimension after preprocessing was $O(\min(n, d\epsilon^{-2}\log n))$, and we will address this in Appendix D.1. Second, we required a constant factor approximation of the optimal objective value for which we give the procedure in Appendix D.2.

D.1. Simulated Sampling of A

In Lemma 12, our primary preconditioning lemma, we set N to be the minimum of n and $O(d\epsilon^{-2}\log n)$. However, all of our sampling above assumed that $O(d\epsilon^{-2}\log n)$ rows were sampled to achieve certain matrix concentration results. Accordingly, we will still assume that $O(d\epsilon^{-2}\log n)$ rows are sampled, but show that we can reduce the computational cost of any duplicate rows to $O(1)$, and hence the computation factor of N can be assumed to be $\min\{n, O(d\epsilon^{-2}\log n)\}$. The sampling procedure itself can be done in about $O(n)$ time. At the end of this section, we explain how the running time of Katyusha can be made to depend on n, rather than $d\epsilon^{-2}\log n$.

Ultimately, our proof of Lemma 12 will critically use the fact that $\tilde{\boldsymbol{A}}$ has $O(d\epsilon^{-2}\log n)$ rows in several places. The following lemmas will then show how we can reduce this computation for duplicate rows, allowing us to substitute n for $O(d\epsilon^{-2}\log n)$ in the running time when $n \ll d\epsilon^{-2}\log n$.

Lemma 42 *Let $\tilde{\boldsymbol{A}} \in \mathbb{R}^{N\times d}$ be a matrix with at most n unique rows, and for each unique row, we are given the number of copies in $\tilde{\boldsymbol{A}}$. Then computing $\tilde{\boldsymbol{A}}^T\tilde{\boldsymbol{A}}$ takes at most $O(nd^{\omega-1})$ time.*

Proof By definition

$$\tilde{\boldsymbol{A}}^T\tilde{\boldsymbol{A}} = \sum_i \tilde{\boldsymbol{A}}_{i,:}^T\tilde{\boldsymbol{A}}_{i,:}.$$

Therefore, if we have k copies of row $\tilde{\boldsymbol{A}}_{i,:}$, we know that they contribute $k\tilde{\boldsymbol{A}}_{i,:}^T\tilde{\boldsymbol{A}}_{i,:}$ to the summation. Accordingly, if we replaced all of them with one row $\sqrt{k}\tilde{\boldsymbol{A}}_{i,:}$, then this row would contribute an equivalent amount to the summation. As a result, we can combine all copies of unique rows to achieve an $n \times d$ matrix $\tilde{\boldsymbol{A}}'$ and compute $\tilde{\boldsymbol{A}}'^T\tilde{\boldsymbol{A}}'$ which will be equivalent to $\tilde{\boldsymbol{A}}^T\tilde{\boldsymbol{A}}$. ■

Corollary 43 *Let $[\tilde{\boldsymbol{A}}\ \tilde{\boldsymbol{b}}] \in \mathbb{R}^{N\times(d+1)}$ be a matrix with at most n unique rows, and for each unique row, we are given the number of copies in $[\tilde{\boldsymbol{A}}\ \tilde{\boldsymbol{b}}]$. Then computing $\tilde{\boldsymbol{A}}\boldsymbol{U}$ where $\boldsymbol{U} \in \mathbb{R}^{d\times d}$, and computing $\tilde{\boldsymbol{A}}^T\tilde{\boldsymbol{b}}$ takes $O(nd^{\omega-1})$ and $O(nd)$ time, respectively.*

Proof We can similarly use the fact that $\tilde{\boldsymbol{A}}_{i,:}\boldsymbol{U}$ is equivalent for all copies of $\tilde{\boldsymbol{A}}_{i,:}$ and combine k copies into the row $k\tilde{\boldsymbol{A}}_{i,:}$.

Analogously, we have $\tilde{\boldsymbol{A}}^T\tilde{\boldsymbol{b}} = \sum_i \tilde{\boldsymbol{A}}_{i,:}^T\tilde{\boldsymbol{b}}_i$, so we can combine duplicate rows. ■

Furthermore, we need to show that we can efficiently sample $O(d\epsilon^{-2}\log n)$ rows (ideally in $O(n)$-time) even when $O(d\epsilon^{-2}\log n) \gg n$. We will achieve this through known results on fast binomial distribution sampling.

Theorem 44 (Theorem 1.1 in Farach-Colton and Tsai (2015)) *Given a binomial distribution $B(n,p)$ for $n \in \mathbb{N}$, $p \in \mathbb{Q}$, drawing a sample from it takes $O(\log^2 n)$ time using $O(n^{1/2+\epsilon})$ space w.h.p., after $O(n^{1/2+\epsilon})$-time preprocessing for small $\epsilon > 0$. The preprocessing does not depend on p and can be used for any $p' \in \mathbb{Q}$ and for any $n' \leq n$.*

This result implies that sampling m items independently can be done more efficiently if $m \gg n$, where we are only concerned with the number of times each item in the state space is sampled.

Corollary 45 *Given a probability distribution $\mathcal{P} = (p_1, ..., p_n)$ over a state space of size n, sampling m items independently from $\mathcal{P}$ takes $O(m^{1/2+\epsilon} + n\log^2 n)$-time.*

Proof Note that sampling independently m times is equivalent to determining how many of each item is sampled by using the binomial distribution and updating after each item. More specifically, we can iterate over all $i \in [n]$ and draw $k_i \sim B(m, p_i)$, then update m to be $m - k_i$ and scale up each p_j (where $j > i$) by $(1-p_i)^{-1}$. It is straightforward to make the scaling up of each p_j efficient, and according to Theorem 44 we can obtain the binomial sample in $O(\log^2 n)$-time.

Furthermore, because m is decreasing at each iteration, we can use the original preprocessing in Theorem 44 for each step to achieve our desired running time. ■

Corollary 46 *Given a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times d}$, with a probability distribution over each row, we can produce a matrix $\tilde{\boldsymbol{A}} \in \mathbb{R}^{O(d\epsilon^{-2}\log n)\times d}$ according to the given distribution in time at most $O\left(\min(d\epsilon^{-2}\log n, (d\epsilon^{-2}\log n)^{1/2+o(1)} + n\log^2 n)\right)$.*

Finally, our application of Theorem 26 assumes that it is given an $N \times d$ matrix, but we assumed that the computational cost could assume $N = \min\{n, O(d\epsilon^{-2}\log n)\}$. A closer examination of Algorithm 2 in Allen Zhu (2017), which is the routine for Theorem 26, shows that the factor of N comes from a full gradient calculation, which can be done more quickly by combining rows in an equivalent manner to the lemma and corollary above.

D.2. Approximating the Optimal Objective Value

For ease of notation, we let $f^* = \min_{\boldsymbol{x}} \|\boldsymbol{Ax}-\boldsymbol{b}\|_1$ and $f_2^* = \min_{\boldsymbol{x}} \|\boldsymbol{Ax}-\boldsymbol{b}\|_2$ in this section. In our proof of both Theorem 16 and 1, we assumed access to a constant approximation of f^* with a runtime overhead of $\log n$. We will obtain access to this value by giving polynomially approximate upper and lower bounds on f^* and using our primary algorithm on $\log n$ guesses for f^* within this range. We start with the following lemma that gives upper and lower bounds on f^*:

Lemma 47 *Given a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times d}$ and a vector $\boldsymbol{b} \in \mathbb{R}^n$, if $\boldsymbol{x}_2^*$ minimizes $\|\boldsymbol{Ax}-\boldsymbol{b}\|_2$ then*

$$\|\boldsymbol{Ax}_2^* - \boldsymbol{b}\|_2 \leq \|\boldsymbol{Ax}^* - \boldsymbol{b}\|_1 \leq \sqrt{n}\,\|\boldsymbol{Ax}_2^* - \boldsymbol{b}\|_2 .$$

Proof By known properties of ℓ_1 and ℓ_2, for any $\boldsymbol{x} \in \mathbb{R}^n$, we have $\|\boldsymbol{x}\|_2 \leq \|\boldsymbol{x}\|_1 \leq \sqrt{n}\,\|\boldsymbol{x}\|_2$. Accordingly, we must have

$$\|\boldsymbol{A}\boldsymbol{x}_2^* - \boldsymbol{b}\|_2 \leq \|\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b}\|_2 \leq \|\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b}\|_1\,,$$

where the first inequality follows from $\boldsymbol{x}_2^*$ being the ℓ_2-minimizer. Similarly, we also have

$$\|\boldsymbol{A}\boldsymbol{x}^* - \boldsymbol{b}\|_1 \leq \|\boldsymbol{A}\boldsymbol{x}_2^* - \boldsymbol{b}\|_1 \leq \sqrt{n}\,\|\boldsymbol{A}\boldsymbol{x}_2^* - \boldsymbol{b}\|_2\,,$$

where the first inequality follows from $\boldsymbol{x}^*$ being the ℓ_1-minimizer. ■

Since $\boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}$, our initialization of $\boldsymbol{A}^T b$ is equal to $\boldsymbol{x}_2^*$. Then we can compute $\|\boldsymbol{A}\boldsymbol{x}_2^* - \boldsymbol{b}\|_2$ in $O(Nd)$ time.[17] Consequently, if we let f_2^* be the minimized objective function $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2$, we can compute polynomially close upper and lower bounds, f_2^* and $\sqrt{n}f_2^*$ respectively, for f^*.

Lemma 48 *In both variants of our primary algorithm for Theorem 16 and 1, we can run the respective algorithms with a constant approximation of f^* by running them $\log n$ times using different approximations of f^*, which we will denote by $\tilde{f}^*$. Furthermore, the runtime of each is independent of the choice of $\tilde{f}^*$.*

Proof We first examine the latter claim and note that the gradient descent portion of both algorithms take upper bounds on $\|\boldsymbol{x}_0 - \boldsymbol{x}^*\|_2$ as inputs. Therefore, given a certain $\tilde{f}^*$ we can input the upper bound $O(\sqrt{d/n})\tilde{f}^*$ and following the analysis of the proofs in Theorems 16 and 1, in runtime $O(d^3\epsilon^{-2})$ and $O\left(d^{2.5}\epsilon^{-2}\sqrt{\log n}\right)$ respectively, we are guaranteed that we achieve $\tilde{\boldsymbol{x}}$ such that $f(\tilde{\boldsymbol{x}}) - f^* \leq \epsilon \tilde{f}^*$ with high probability. However, note that this is only true if $f^* \leq \tilde{f}^*$. Otherwise, we are given no guarantee on the closeness of $f(\tilde{\boldsymbol{x}})$ to f^*.

The runtime of each algorithm is then not affected by our approximation of $\tilde{f}^*$, however, the closeness guarantees are affected. Accordingly, we will run the gradient descent procedure in each respective algorithm $\log n$ times with $\tilde{f}^* = f_2^* \cdot 2^i$ for $i = 0$ to $\log n$, and whichever iteration produces $\tilde{\boldsymbol{x}}$ that minimizes $f(\cdot)$ will be output. Lemma 47 implies that there must exist some i such that $f_2^* {\cdot} 2^i \leq f^* \leq f_2^* {\cdot} 2^{i+1}$. Therefore, when we run our algorithm with $\tilde{f}^* = f_2^* {\cdot} 2^{i+1}$, the algorithm will succeed with high probability. Thus, the overall success probability is at least as high as any individual run of the algorithm. Moreover, the output $\tilde{\boldsymbol{x}}$ is guaranteed to have $f(\tilde{\boldsymbol{x}}) - f^* \leq 2\epsilon f^*$. ■

17. Note that our sampled and rotated $\tilde{\boldsymbol{A}}\boldsymbol{U}$ from Lemma 12 loses any sparsity guarantees that $\boldsymbol{A}$ may have had.