We consider the $\ell_1$ minimization problem $\min_x \|Ax - b\|_1$ in the overconstrained case, commonly known as the Least Absolute Deviations problem, where there are far more constraints than variables. More specifically, we have $A \in \mathbb{R}^{n \times d}$ for $n \gg d$. Many important combinatorial problems, such as minimum cut and shortest path, can be formulated as $\ell_1$ regression problems [CMMP13]. We follow the general paradigm of preconditioning the matrix and solving the resulting problem with gradient descent techniques, and our primary insight will be that these methods are actually interdependent in the context of this problem. The key idea will be that preconditioning from [CP15] allows us to obtain an isotropic matrix with fewer rows and strong upper bounds on all row norms. We leverage these conditions to find a careful initialization, which we use along with smoothing reductions in [AH16] and the accelerated stochastic gradient descent algorithms in [All17] to achieve $\epsilon$ relative error in about $\|A\| + nd^{\omega-1} + \sqrt{nd^2} \epsilon^{-1}$ time with high probability. Moreover, we can also assume $n \leq O(d \epsilon^{-2} \log n)$ from preconditioning. This improves over the previous best result using gradient descent for $\ell_1$ regression [YCRM16], and is comparable to the best known running times for interior point methods [LS15].

Finally, we also show that if our original matrix $A$ is approximately isotropic and the row norms are approximately equal, we can avoid using fast matrix multiplication and prove a running time of about $\|A\| + sd^{1.5} \epsilon^{-2} + d^2 \epsilon^{-2}$, where $s$ is the maximum number of non-zeros in a row of $A$. 

---

*Georgia Institute of Technology. email:ddurfee@gatech.edu
†Georgia Institute of Technology. email:kevinlai@gatech.edu
‡Georgia Institute of Technology. email:sawlani@gatech.edu
1 Introduction

The unconstrained over-determined $\ell_1$ regression problem, popularly known as the Least Absolute Deviations problem, is defined as follows:

$$\min_{x \in \mathbb{R}^d} \|Ax - b\|_1,$$

where $A \in \mathbb{R}^{n \times d}$, $b \in \mathbb{R}^n$ and $n \gg d$. Compared to Least Squares ($\ell_2$) regression, the $\ell_1$ 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 $\ell_1$ regression problems \cite{CMMP13}, and high accuracy $\ell_1$ regression can be used to solve general linear programs.

This problem can be formulated as a linear program \cite{PK97, CDS01}, and has traditionally been solved by generic methods for solving linear programs, such as the interior-point method (IPM) \cite{PK97, Por97, MM13b, LS15}. Although interior point and simplex methods significantly beat the theoretical guarantees of stochastic gradient descent (SGD) algorithms for optimization \cite{NV08, NJLS09}, SGD techniques still held interest for regression problems because of their practical efficiency and simplicity. Yet even combining acceleration techniques introduced by Nesterov \cite{Nes83}, originally designed for ‘smooth’ objectives, with methods extending acceleration to certain classes of non-smooth functions \cite{Nes05b, Nes05a, Nes07, AH16} did not yield SGD algorithms for $\ell_1$ regression that provably ran in poly($d$) time.

In order to narrow this gap, sketching techniques involving randomized linear algebra were introduced. These looked to find a low-distortion embedding of $A$ into a smaller subspace, after which popular techniques for $\ell_1$ regression could be applied on the reduced matrix. Efforts to build these sampled matrices or “coresets” have been made using random sampling \cite{Cla05}, fast matrix multiplication \cite{SW11}, and ellipsoidal rounding \cite{DDH+09, CDMI+13}. All of these methods produce coresets of size poly($d, \epsilon^{-1}$) $\times$ $d$ in time $O(n \cdot \text{poly}(d))$. \cite{MM13a} and \cite{WZ13} improve these techniques to produce similar coresets in $O(n \text{nnz}(A) + \text{poly}(d))$ time.

An alternative to using sketching as a preprocessing step is using preconditioners, which provide efficiently sampled matrices with additional desirable properties. In particular, we use the preconditioning technique of \cite{CP15}, which utilizes a Lewis change of density \cite{Lew78} to sample rows of $A$ with probability proportional to their Lewis weights such that the sampled matrix preserves $\ell_1$ distances. Lewis weights also lead to nice $\ell_2$ conditions, which are required by SGD, so Lewis weights are a natural way to sample a matrix when solving $\ell_1$ regression with SGD methods. Recently, \cite{YCRM16} achieved faster gradient descent methods for $\ell_1$ regression, in part by applying these preconditioning techniques. The key difference in our work will be to leverage special properties from Lewis weights preconditioning to achieve better guarantees for SGD, comparable to those given by IPMs. In this way, our work is the first real $\ell_1$ analog of the work in the $\ell_2$ regression in \cite{AMT10}.

Table 1 compares the running time of our algorithm to the fastest gradient descent methods \cite{Cla05, Nes09, YCRM16}, interior point methods \cite{MM13b, LS15}, and multiplicative weights update methods \cite{CMMP13}. Notice that in terms of dependence on $n$ and $d$, our accelerated algorithm improves upon all gradient descent methods and is comparable to the current best IPM \cite{LS15}. In fact, there are parameter regimes where our running time bounds are better than \cite{LS15}. While these regimes are small, we believe they are significant, as discussed at the end of Section 1.1.

\footnote{For instance, one can determine if $\{x | Ax - b, x \geq 0\}$ is feasible by writing an objective of the form $\alpha(\|Ax - b\|_1 + \|x\|_1 + \|x - \beta I\|_1)$ where $\alpha$ and $\beta$ are sufficiently large polynomials in the input size.}
Running time $O(\tilde{\Omega} \log d \log \frac{1}{\epsilon})$ to improve the runtime to $\tilde{O}(\sqrt{d} \log(1/\epsilon))$. These properties can be further applied to smoothing reductions from $\text{AH16}$ and weights row sampling $\text{CP15}$, which will allow us to assume $n = o(d^2)$.

In this paper, we use the same underlying preconditioning method, i.e., Lewis weights row sampling $\text{CP15}$, which will allow us to assume $n \leq O(de^{\frac{1}{2}} \log n)$. All running times are to find a solution with $\epsilon$ relative error, with constant failure probability.

### 1.1 Our Results

As mentioned above, our techniques for solving the $\ell_1$ regression problem will follow the general paradigm of preconditioning and then using gradient descent methods on the resulting problem. However, instead of looking for ways to improve the preconditioning or gradient descent, we show that in some sense improvements in the two are not independent of one another. The crucial idea in this paper is that preprocessing techniques can give us some strong properties in addition to the low-dimensional embedding, and we are able to effectively leverage these properties into faster running times for $\ell_1$ regression.

In particular, preconditioning the given matrix-vertex pair $[A \ b]$ using Lewis weights $\text{CP15}$ to get $[\tilde{A} \ \tilde{b}]$ has the critical property that all the leverage scores of the new matrix are approximately equal, in addition to the fact that $\|Ax - b\|_1 \approx \|\tilde{A}x - \tilde{b}\|_1$ for every $x \in \mathbb{R}^d$. Since rotations of a matrix do not change its leverage scores, we are free to rotate $\tilde{A}$ to place it into isotropic position. As a result, our analysis of the $\ell_1$ regression problem can now assume that the input matrix is both isotropic and has strong upper bounds on every row norm.

These assumptions on the input matrix yield two essential properties, (1) a careful choice of initial vector can be shown to be close to optimal, and (2) 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 an objective with $\epsilon$ relative error, leading to a total running time of $O(\text{nnz}(A) \log n + d^3 \epsilon^{-2})$, which already beats the previous best SGD-based results for $\ell_1$-regression $\text{YCRM16}$. These properties can be further applied to smoothing reductions from $\text{AH16}$ and accelerated SGD algorithms in $\text{All17}$ to improve the runtime to $\tilde{O}(\text{nnz}(A) + nd^{2.5} \epsilon^{-2})$, or if we plug in the number of rows, $O(de^{-2} \log n)$, after sampling by Lewis weights we can also obtain $\tilde{O}(\text{nnz}(A) + nd^{2.5} \epsilon^{-2})$ running time. Algorithm $\text{IPM}$ gives the basic framework of our $\ell_1$ solver.

---

2 $\tilde{O}$ hides terms polylogarithmic in $d$ and $n$.

3 $O(n^\omega)$ denotes the running time of the fastest algorithm to multiply two $n \times n$ matrices. ($\omega \approx 2.373$ is the current best value. $\text{WH12}$ $\text{DS13}$ $\text{LG14}$).

4 Interior Point Cutting Plane Methods
Input: Matrix $A \in \mathbb{R}^{n \times d}$, and vector $b \in \mathbb{R}^n$, along with error parameter $\epsilon > 0$.
1. Precondition $[A \; b]$ by Lewis weight sampling as in [CP15], along with a matrix rotation.
2. Initialize $x_0$ according to the exact or approximate corresponding $\ell_2$ minimizer.
3. Run a stochastic gradient descent algorithm on the preconditioned matrix with starting point $x_0$.

Algorithm 1: General structure of our algorithm

Theorem 1.1. Given $A \in \mathbb{R}^{n \times d}$, $b \in \mathbb{R}^n$, assume $\min_x \|Ax - b\|_2$ is either 0 or bounded below by some polynomial in $n$. Then for any $\epsilon > 0$, there is a routine that outputs $\tilde{x}$ such that with high probability\(^5\)

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

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

Note that our $\ell_1$ solver will always run the Lewis weights preconditioning, which is essentially a specific row sampling procedure of the original matrix. Thus, 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 can carry out the rest of our linear algebraic manipulations in time dependent on $n$ rather than $d\epsilon^{-2}\log n$\(^6\).

Since the theoretical running bounds of fast matrix multiplication are difficult to achieve in practice, it would be ideal for the 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. Moreover, this method preserves the rowsparsity of $A$. The primary difficulty of this routine will be accounting for the fact that $A$ is only approximately isotropic in our computations of the $x_0$ initialization, and will require utilizing efficient $\ell_2$ regression solvers that do not rely on fast matrix multiplication.

Theorem 1.2. Let $[A \; b]$ be an $n \times d$ matrix, $n \times 1$ vector pair, such that $[A \; b]^T[A \; b] \approx_{O(1)} I$ and for each row $i$ of $[A \; b]$, $\| [A \; b]_i \|_2^2 \approx_{O(1)} d/n$. Assume $\min_x \|Ax - b\|_2 \geq n^{-c}$ and $\|b\|_2 \leq n^c$ for some constant $c > 0$. Then for any $\epsilon > 0$, there is a routine that computes $\tilde{x}$ such that with high probability

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

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

Note that if $s \leq d^{0.87} \ll d^{2-1.5}$, then our algorithm beats fast matrix multiplication.

\(^5\)For ease of notation, we ignore factors of $\log(1/\xi)$ in the running times throughout this paper, where $\xi$ is the failure probability

\(^6\)We also note that the “$n$” used by Katyusha is the time required to compute a full gradient, which we can also do in $O(n)$ time.
Comparison to \[\text{LS15}\] and conjectures

Our algorithm achieves comparable bounds to \[\text{LS15}\], who have the current best IPM for \(\ell_1\) regression. When combined with Lewis weights row sampling, \[\text{LS15}\] have the best overall bounds for \(\ell_1\) regression. In the case of Lewis weights sampling, \[\text{LS15}\] can assume \(\text{nnz}(A) = O(d^2\epsilon^{-2}\log n)\), and plugging in \(n = O(d\epsilon^{-2}\log n)\) for our result gives us both equivalent running times of about \(\tilde{O}(d^2\epsilon^{-2})\). Our algorithm outperforms \[\text{LS15}\] in the parameter regime where \(n \ll d\epsilon^{-2}\log n\) and \(\text{nnz}(A) = \Theta(nd)\), for which the leading order term in \[\text{LS15}\] is \(\tilde{O}(nd\epsilon^{-1}\log(1/\epsilon))\), while our leading order term is \(O(\sqrt{n}d\epsilon^{-1}\log n)\), so our bound is better for \(n \geq d/\sqrt{\epsilon}\). Finally, \[\text{LS15}\] uses preprocessing techniques from \[\text{Vai89}\], which in turn require that \(\omega \leq 2.4\), while we only need \(\omega \leq 2.5\).

The fact that our algorithm achieves \(\tilde{O}(\text{nnz}(A) + nd\epsilon^{-1}\sqrt{n}d\epsilon^{-1})\) running time is significant because it suggests that \(\text{nnz}(A) + \text{poly}(d)\epsilon^{-1}\) algorithms are possible for \(\ell_1\) regression. In fact, we believe a more careful analysis of our sampling and SGD might immediately yield such an algorithm. “Sketch and solve” approaches tend to have \(\epsilon^{-2}\) dependence in their running times for \(\ell_1\) regression due to the their use of standard concentration bounds \[\text{ACW16, PSW17}\], even though \(\text{nnz}(A) + \text{poly}(d)\log(1/\epsilon)\) bounds are possible for \(\ell_2\) regression using iterative refinement methods such as conjugate gradient. As such, we conjecture that \(\text{nnz}(A) + \text{poly}(d)\log(1/\epsilon)\) running times are achievable for \(\ell_1\) regression.

1.2 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 \(\ell_1\) 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. In Section 5, we show that row sampling using Lewis weights \[\text{CP15}\], along with matrix rotation, suffices to give us a matrix satisfying our precondition requirements. Section 6 has some technical details from Section 3.

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 \(A_i:\) denote the \(i^{th}\) row of a matrix \(A\), and we use \(\text{nnz}(A)\) to denote the number of non-zero elements in \(A\). \(A^\dagger\) refers to the Moore-Penrose pseudoinverse of \(A\). When \(A\) has linearly-independent columns, \(A^\dagger = (A^TA)^{-1}A^T\). Also, we assume that the input \(A\) has full rank.

**Definition 2.1 (\(\ell_p\)-norm).** The \(\ell_p\) norm of a vector \(v \in \mathbb{R}^n\) is defined as

\[
\|v\|_p \overset{\text{def}}{=} \left(\sum_{i=1}^{n} v_i^p\right)^{1/p}.
\]

Accordingly, the \(\ell_p\) norm of a matrix \(A \in \mathbb{R}^{n \times d}\) is defined as

\[
\|A\|_p \overset{\text{def}}{=} \sup_{x \in \mathbb{R}^d, x \neq 0} \frac{\|Ax\|_p}{\|x\|_p}.
\]
**Definition 2.2** (Matrix approximation). We say that $A \approx_{\kappa} B$ if and only if 
\[
\frac{1}{\kappa} B \preceq A \preceq \kappa B.
\]
Here, $\preceq$ refers to the Löwner partial ordering of matrices, where we say that $A \preceq B$ if $B - A$ is positive semi-definite.

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

**Definition 2.3** (IRB). A matrix $A \in \mathbb{R}^{n \times d}$ with $n \geq d$ is said to be isotropic row-bounded (IRB) if the following hold:
1. $A^T A = I$,
2. For all rows of $A$, $\|A_i\|_2^2 \leq O(d/n)$.

**Definition 2.4**. Given a matrix $A$, we define the statistical leverage score of row $A_i$ to be
\[
\tau_i(A) \coloneqq A_i : (A^T A)^{-1} A_i^T = \left\| (A^T A)^{-1/2} A_i^T \right\|_2^2.
\]

**Definition 2.5**. For a matrix $A$, the $\ell_1$ Lewis weights $\overline{w}$ are the unique weights such that for each row $i$ we have
\[
\overline{w}_i = \tau_i \left( \overline{W}^{-1/2} A \right)
\]
or equivalently
\[
\overline{w}^2 = A_i : \left( A^T \overline{W}^{-1} A \right)^{-1} A_i^T
\]
where $\overline{W}$ is the diagonal matrix formed by putting the elements of $\overline{w}$ on the diagonal.

**Definition 2.6**. A function $f(x)$ 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 2.7**. A function $f(x)$ is $\sigma$-strongly convex if for any $x, y \in \mathbb{R}^d$,
\[
f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle + \frac{\sigma}{2} \|x - y\|_2^2.
\]

**Definition 2.8**. A function $f(x)$ 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 $\ell_1$ Regression

In this section, we describe how we achieve the bounds in Theorem 1.1. We first introduce the preconditioning technique from [CP15], which, along with rotating the matrix, will reduce our problem to $\ell_1$ minimization where the input matrix $A$ is isotropic and the norms of all its rows have strong upper bounds, i.e. it is IRB by Definition 2.3. We relegate the details and proof of this preconditioning procedure to Section 5.

In Sections 3.2 and 3.3 we prove that known stochastic gradient descent algorithms will run provably faster if we assume that $A$ is IRB. In particular, if $A$ is IRB, we can find an initialization $x_0$ that is close to the optimum $x^*$, 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 $\tilde{O}(nnz(A) + d^2 \epsilon^{-2})$. Finally, we take known smoothing techniques from [AH16] along with the Katyusha accelerated stochastic gradient descent from [All17] to achieve a runtime of $\tilde{O}(nnz(A) + d^{2.5} \epsilon^{-2})$. 


3.1 Preconditioning with Lewis weights

The primary tool in our preconditioning routine will be a sampling scheme by Lewis weights introduced in [CP15] that was shown to approximately preserve the $\ell_1$ norm. Specifically, we will use the combination of two primary theorems from [CP15] that approximately compute the Lewis weights of a matrix quickly and then sample accordingly while still approximately preserving $\ell_1$ norm distances with high probability.

**Theorem 3.1** (Theorem 2.3 and 6.1 from [CP15]). Given a matrix $A \in \mathbb{R}^{n \times d}$ with $\ell_1$ Lewis weights $\overline{p}$ and an error parameter $\epsilon > 0$, then for any function $h(n, \epsilon) \geq O(\epsilon^{-2} \log n)$, we can find sampling values

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

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

$$\|Ax\|_1 \approx_{1+\epsilon} \|A\|_1$$

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

In Section 5 we will show that this sampling scheme also ensures that each row of $A$ has approximately the same leverage score. Furthermore, we will obtain additional nice properties by rotating $A$ and showing that a solution to our reduced problem gives an approximate solution to the original problem, culminating in the following lemma.

**Lemma 3.2.** There is a routine that takes a matrix $A \in \mathbb{R}^{n \times d}$, a vector $b \in \mathbb{R}^n$ and $\epsilon > 0$, then produces a matrix $[A, \tilde{b}] \in \mathbb{R}^{N \times (d+1)}$ with $N = O(d\epsilon^{-2} \log n)$ and an invertible matrix $U \in \mathbb{R}^{d \times d}$ such that matrix $A U$ is IRB and if $\tilde{x}_U^*$ minimizes $\|AUx - \tilde{b}\|_1$, then for any $\tilde{x}$ such that

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

we must have

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

with high probability.

Furthermore, the full running time is $O(nnz(A) \log n + d\omega^{-1} \min\{d\epsilon^{-2} \log n, n\} + \Upsilon)$ 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 $A$ are already in the same form as $A U$, and simply find solutions with small relative error in that objective function.

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

To demonstrate the usefulness of the properties of our preconditioned $A$, we consider standard stochastic gradient descent and the bounds on its running time. We let $x^* = \arg\min_x \|Ax - b\|_1$.

**Theorem 3.3** ([RS86]). Given a function $f$ and $x_0$ such that $\|x_0 - x^*\|_2 \leq R$ and $L$ is an upper bound on $\|\nabla(n \cdot [A_i, x - b_i])\|_2$ for all $i$, then projected subgradient descent ensures that after $t$ steps:

$$f(x_t) - f(x^*) \leq O\left(\frac{RL}{\sqrt{t}}\right),$$

where $x^* = \arg\min_x f(x)$. 

Our assumptions on $A$ will give us bounds on the initialization distance $\|x_0 - x^*\|_2$ and the norm of the gradient.

**Lemma 3.4.** If $A$ is IRB, then by setting $x_0 = A^T b$ we have

$$\|x_0 - x^*\|_2^2 \leq O\left(\frac{d}{n}\right) \|Ax^* - b\|_1^2.$$ 

**Proof.**

\[
\|x^{(0)} - x^{(*)}\|_2^2 = \|A^T b - x^{(*)}\|_2^2 \\
= \|A^T \left(b - Ax^{(*)}\right)\|_2^2 \quad \text{by assumption } A^T A = I \\
= \left|\sum_i A_{i,:} \left(b - Ax^{(*)}\right)\right|_2^2 \\
\leq \|Ax^* - b\|_1^2 \cdot \max_i \|A_{i,:}\|_2^2 \quad \text{by convexity of } \|\cdot\|_2, \text{ also shown in Lemma 4.8} \\
= O\left(\frac{d}{n}\right) \|Ax^* - b\|_1^2. \\
\]

**Lemma 3.5.** If $A$ is IRB, then $\|\nabla(n \cdot |A_{i,:} x - b_i|)\|_2^2 \leq O(nd)$ for all $i$.

**Proof.** We see that $\nabla(n \cdot |A_{i,:} x - b_i|) = n \cdot A_{i,:} \text{sgn}(A_{i,:} x - b_i)$, and $\|A_{i,:}\|_2 \leq O(d/n)$ for all $i$ by our assumption that $A$ is IRB. This then implies our desired inequality.

These bounds, particularly the initialization distance, are stronger than the bounds for general $A$, and together will give our first result that improves upon the runtime in [YCRM16] by using our preconditioning.

**Theorem 3.6.** Given $A \in \mathbb{R}^{n \times d}$, we can find $\tilde{x} \in \mathbb{R}^d$ using preconditioning and stochastic gradient descent such that

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

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

**Proof.** By preconditioning with Lemma 3.2 and error $O(\epsilon)$ we obtain an $N \times d$ matrix $\tilde{A} U$ in time $O(nnz(A) \log n + d^2 \epsilon^{-2})$.

By Theorem 3.3 we then need to run $O(d^2 \epsilon^{-2})$ iterations of standard stochastic gradient descent to achieve absolute error of $O(\epsilon \cdot f(x^*))$ 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 $f(x^*)$. We will show in Section 6 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 $\tilde{x}$ with $O(\epsilon)$ relative error to the optimal objective function value in time $O(nnz(A) \log n + d^3 \epsilon^{-2})$. Adding the factor $\log n$ overhead from estimating $f(x^*)$ gives the desired runtime.
3.3 Smoothing Reductions and Katyusha Accelerated SGD

We now further examine whether our strong initialization distance bound will allow us to improve the running time with 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 [Nes05b, Nes07, DBW12, OG12, AH16] 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 certain error and runtime guarantees. Accordingly, we will first show how our initialization distance fits into the reduction of [AH16], then apply Katyusha’s accelerated gradient descent algorithm in [All17] to their framework.

Smoothing the Objective Function and Adding Strong Convexity

As before, we let $x^* = \arg\min_x \|Ax - b\|_2$. For clarity, we will borrow some of the notation from [AH16] to more clearly convey their black-box reductions.

**Definition 3.7.** Function $f(x)$ is $(L, \sigma)$-smooth-sc if it is both $L$-smooth and $\sigma$-strongly-convex.

**Definition 3.8.** An algorithm $A(f(x), x_0)$ is a $\text{Time} A(L, \sigma)$-minimizer if $f(x)$ is $(L, \sigma)$-smooth-sc and $\text{Time} A(L, \sigma)$ is the time it takes $A$ to produce $x'$ such that $f(x') - f(x^*) \leq f(x_0) - f(x^*)$ for any starting $x_0$.

Allen-Zhu and Hazan assume access to efficient $\text{Time} A(L, \sigma)$-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 3.9** (Theorem C.2 from [AH16]). 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 $x_0$ be a starting vector such that $f(x_0) - f(x^*) \leq \Delta$ and $\|x_0 - x^*\|_2 \leq \Theta$. Then there is a routine that takes as input a $\text{Time} A(L, \sigma)$-minimizer, $A$, 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

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

It is then straightforward to show that our objective function fits the necessary conditions to utilize Theorem 3.9.

**Lemma 3.10.** If $A$ is IRB, then the function $\|Ax - b\|_1$ can be written as $\frac{1}{n} \sum_{i=1}^{n} f_i(x)$ such that each $f_i(\cdot)$ is $O(\sqrt{nd})$-Lipschitz continuous.

**Proof.** By the definition of 1-norm,

$$\|Ax - b\|_1 = \sum_{i=1}^{n} |A_i; x - b_i|.$$

We then set $f_i(x) = n \cdot |A_i; x - b_i|$ and the result follows from Lemma 3.5. 

\[\square\]
We can then incorporate our objective into the routine from Theorem 3.9 along with our initialization of \( x_0 \).

**Lemma 3.11.** Let \( A \) be an \( \text{TIME}_{A}(L, \sigma)\)-minimizer, along with objective \( \| Ax - b \|_1 \) such that \( A \) is IRB and \( x_0 = A^T b \), then the routine from Theorem 3.9 produces \( x_T \) satisfying \( f(x_T) - f(x^*) \leq O(\epsilon) \) in total running time

\[
\sum_{t=0}^{T-1} \text{TIME}_{A} \left( O \left( \frac{nd2^t}{\Delta} \right), O \left( \frac{n\Delta}{d \cdot f(x^*)^{2^t}} \right) \right).
\]

**Proof.** Lemma 3.10 implies that we can apply Theorem 3.9 where \( G = O(\sqrt{nd}) \), and Lemma 3.4 gives \( \Theta = O \left( \frac{d}{n} f(x^*)^2 \right) \). We then obtain \( \beta_0 = O \left( \frac{\Delta}{nd} \right) \) and \( \sigma_0 = O \left( \frac{n\Delta}{d f(x^*)^2} \right) \) and substitute these values in the running time of Theorem 3.9. \( \square \)

**Applying Katyusha Accelerated SGD**

Now that we have shown how our initialization can be plugged into the smoothing construction of [AH16], we simply need an efficient \( \text{TIME}_{A}(L, \sigma)\)-minimizer to obtain all the necessary pieces to prove our primary result.

**Theorem 3.12** (Corollary 3.8 in [AH17]). There is a routine that is a \( \text{TIME}_{A}(L, \sigma)\)-minimizer where \( \text{TIME}_{A}(L, \sigma) = d \cdot O (n + \sqrt{nL}/\sigma) \).

We can then precondition the matrix to give our strong bounds on the initialization distance of \( x_0 \) from the optimal \( x^* \), which allows us to apply the smoothing reduction and Katyusha accelerated gradient descent more efficiently.

**Theorem 1.1.** Given \( A \in \mathbb{R}^{n \times d} \), \( b \in \mathbb{R}^n \), assume \( \min_x \| Ax - b \|_2 \) is either 0 or bounded below by some polynomial in \( n \). Then for any \( \epsilon > 0 \), there is a routine that outputs \( \tilde{x} \) such that with high probability\(^7\)

\[
\| A \tilde{x} - b \|_1 \leq (1 + \epsilon) \| A x^* - b \|_1
\]

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

**Proof.** Once again, by preconditioning with Lemma 3.2 and error \( O(\epsilon) \) we obtain a matrix \( \tilde{A} \in \mathbb{R}^{N \times d} \) and a vector \( \tilde{b} \in \mathbb{R}^n \) in time \( O(\text{nnz}(A) \log n + d^{\epsilon^{-1}} \min \{ d \epsilon^{-2} \log n, n \}) \). We utilize the routine in Theorem 3.12 as the \( \text{TIME}_{A}(L, \sigma)\)-minimizer for Lemma 3.11 and plug the time bounds in to achieve an absolute error of \( O(\delta) \) in the preconditioned objective function with the following running time:

\[
d \cdot O \left( \sum_{t=0}^{T-1} N + \sqrt{N \cdot \frac{(N d 2^t)}{\Delta}} \left( \frac{d \cdot f(x^*)^{2^t}}{N \Delta} \right) \right) = d \cdot O \left( N \log \frac{\Delta}{\delta} + \frac{d \sqrt{N} \cdot f(x^*)}{\Delta} \sum_{t=0}^{T-1} 2^t \right) = d \cdot O \left( N \log \frac{\Delta}{\delta} + \frac{d \sqrt{N} \cdot f(x^*)}{\delta} \right).
\]

To achieve our desired relative error of \( \epsilon \) we need to set \( \delta = O(\epsilon f(x^*)) \). Technically, this means that the input to gradient descent will require at least a constant factor approximation to \( f(x^*) \).

\(^7\)For ease of notation, we ignore factors of \( \log(1/\xi) \) in the running times throughout this paper, where \( \xi \) is the failure probability.
We will show in Section 6 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 \( f(x^*) \) is at most polynomially small, which gives a runtime of \( O\left( dN \log (n \epsilon^{-1}) + \frac{d \sqrt{N}}{\epsilon} \right) \).

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 Section 5.4. This gives a runtime of

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

which, combined with our preconditioning runtime (where \( \Upsilon \) is a lower order term if we assume the \( \epsilon \) is at most polynomially small in \( n \)) and the factor \( \log n \) overhead from estimating \( f(x^*) \), gives the desired runtime. Furthermore, since the error in our preconditioning was \( O(\epsilon) \), by Lemma 3.2 we have achieved a solution with \( O(\epsilon) \) relative error in the original problem.

\[\Box\]

### 4 Row-Sparsity Bounds for \( \ell_1 \) Regression

In this section, we explain how to avoid using matrix multiplication and achieve row-sparsity-preserved \( \ell_1 \) minimization in the case where our given matrix \( A \in \mathbb{R}^{n \times d} \) and vector \( b \in \mathbb{R}^n \) are such that \( [A \ b] \) and for each row \( i \) of \( [A \ b] \), \( \| [A \ b]_i \|_2^2 \approx O(1) \) \( d/n \). Notice that these conditions imply \( A^T A \approx O(1) I \) and \( \|A_i\|_2^2 \leq O(d/n) \), which were properties of the preconditioned matrix \( \tilde{A} U \) generated in Lemma 3.2. Lemma 3.2 used fast matrix multiplication and rotated \( A \), which removed sparsity guarantees for \( \tilde{A} U \). We thus need to avoid Lemma 3.2 which creates two new complications: (1) the row count of matrix \( A \) has not been reduced from \( n \) to \( O(d \epsilon^{-2} \log n) \), and (2) \( \tilde{A} \) is only approximately isotropic.

Section 4.1 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 3.1. In particular, if we uniformly sample \( N = O(d \epsilon^{-2} \log d) \) rows of \( [A \ b] \), this yields a smaller matrix \( [\tilde{A} \ \tilde{b}] \) such that \( \tilde{A}^T \tilde{A} \approx O(1) I \) and \( \|\tilde{A}_i\|_2^2 \leq O(d/N) \) for each row \( i \).

Section 4.2 then describes how to finding a good initialization point using conjugate gradient methods when \( \tilde{A} \) is only approximately isotropic. Finally, Section 4.3 shows how to use the reduction in \[4H16] \) and the Katyusha stochastic gradient descent algorithm from \[4l17] \) to achieve a total running time of \( \tilde{O}(nnz(A) + sd^{1.5} \epsilon^{-2} + d^2 \epsilon^{-2}) \), where \( s \) is the maximum number of non-zeros in a row of \( A \).

#### 4.1 Uniform Sampling of \( A \)

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

**Lemma 4.1.** Suppose we are given a matrix \( A \in \mathbb{R}^{n \times d} \) such that \( A^T A \approx O(1) I \) and \( \|A_i\|_2^2 \approx O(1) \) \( d/n \). If we uniformly sample \( N = O(d \epsilon^{-2} \log n) \) rows independently and rescale each row by \( n/N \) to obtain matrix \( \tilde{A} \), then with high probability the following properties hold:

1. \( \|Ax\|_1 \approx_{1+\epsilon} \|\tilde{A}x\|_1 \) for all \( x \in \mathbb{R}^d \).
2. \( \tilde{A}^T \tilde{A} \approx O(1) \left( \frac{n}{N} \right) I. \)

\[\text{Note that if } f(x^*) = 0, \text{ then our initialization } x_0 = A^T b \text{ will be equal to } x^*. \]
3. $\|\tilde{A}_i\|^2_2 \approx_{O(1)} dn/N^2$ for all rows $i \in \{1, 2, \ldots, N\}$.

To prove Lemma 4.1, we need the following lemma, which states the key fact that the conditions on $A$ ensure approximately uniform Lewis weights.

**Lemma 4.2** (Almost-uniform leverage scores imply almost-uniform Lewis weights). Consider a matrix $A \in \mathbb{R}^{n \times d}$ such that $A^T A \approx_{O(1)} I$ and $\|A_i\|^2_2 \approx_{O(1)} d/n$. Let $\bar{w}$ denote the $\ell_1$ Lewis weights for $A$. Then for each row $i$, we have $\bar{w}_i \approx_{O(1)} d/n$.

**Proof of Lemma 4.1.** Note that by Lemma 4.2 we have $p_i = N/n \approx_{O(1)} d \cdot O(e^{-2 \log n}) / n \approx_{O(1)} \bar{w}_i \cdot O(e^{-2 \log n})$.

Thus, if we use $p_i = N/n$ for each $i$ in Theorem 3.1, we get the first property while avoiding the cost of computing $p_i$’s stated in Theorem 3.1.

The second property follows from Lemma 5.4 in Section 5. Specifically, we have

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

which then implies that

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

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

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

Furthermore, Lemma 5.3 in Section 5 shows that $\tau_i(\tilde{A}) \approx_{O(1)} d/N$. Factoring this into the equation gives us

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

Now, to prove Lemma 4.2 we need the following definition and lemma.

**Definition 4.3** (Definition 5.1 of $\alpha$-almost Lewis weights for $\ell_1$ from [CP15]). For a matrix $A$, an assignment of weights $w$ is $\alpha$-almost Lewis if

$$A_i:(A^T W^{-1} A)^{-1} A_i:T \approx_{\alpha} w_i^2,$$

where $W$ is the diagonal matrix form of $w$.

**Lemma 4.4** (Definition 5.2 and Lemma 5.3 from [CP15]). Any set of $\alpha$-almost Lewis weights satisfy

$$\bar{w}_i \approx_{\alpha} w_i.$$

**Proof of Lemma 4.2**. We know that $A^T A \approx_{O(1)} I$ and for each row $i$, $\|A_i:\|^2_2 \approx_{O(1)} d/n$. Then,

$$\tau_i(A) = A_i:(A^T A)^{-1} A_i:T \approx_{O(1)} A_i:A_i:T \Rightarrow \tau_i(A) \approx_{O(1)} d/n.$$

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

$$A_i:(A^T W^{-1} A)^{-1} A_i:T = (d/n) A_i:(A^T A)^{-1} A_i:T \approx_{O(1)} d^2/n^2 = w_i^2.$$

Thus, $w$ is $O(1)$-almost Lewis. The result then follows by Lemma 4.4.

\[\square\]
4.2 Initialization using Approximate \( \ell_2 \) Minimizer

In this section, we describe how to find a good initialization \( x_0 \) for gradient descent even with our relaxed assumptions on \( A \). Previously, when we had \( A^TA = I \), we used \( x_0 = A^Tb = \arg\min_x \| Ax - b \|_2 \). It turns out that for \( A^TA \approx_{O(1)} I \), the \( \ell_2 \) minimizer \( x_0 = \arg\min_x \| Ax - b \|_2 \) is still a good initialization point. But finding an exact \( \ell_2 \) minimizer would take a prohibitive amount of time or would require matrix multiplication. However, an approximate \( \ell_2 \) minimizer suffices, and we can find such a point quickly using the conjugate gradient method.

For this section, we define \( x^* \defeq \arg\min_x \| Ax - b \|_1 \) and \( x_0 \defeq \arg\min_x \| Ax - b \|_2 \). Our main result is the following:

**Lemma 4.5.** Let \( A \in \mathbb{R}^{n \times d} \) be such that \( A^TA \approx_{O(1)} I \) and for each row \( i \) of \( A \), \( \| A_i \|_2^2 \leq O(d/n) \). Assume that \( \| b \|_2 \leq n^c \) and \( \| Ax_i - b \|_2 \geq n^{-c} \) for some constant \( c > 0 \).

\[
\| x_0 - x^* \|_2 \leq O(\sqrt{d/n}) \| Ax^* - b \|_1 .
\]

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

We prove Lemma 4.5 using the following two lemmas whose proofs are deferred until after the proof of Lemma 4.5. Lemma 4.6 is our equivalent statement to Lemma 4.4 where we show that the \( \ell_2 \) minimizer is close to the \( \ell_1 \) minimizer even when \( A \) is only approximately isotropic. Lemma 4.7 accounts for the fact that we cannot exactly find the \( \ell_2 \) minimizer, but will be able to obtain a very close estimate.

**Lemma 4.6 (Exact \( \ell_2 \)-minimizer is close to \( x^* \)).** Let \( A \in \mathbb{R}^{n \times d} \) be such that \( A^TA \approx_{O(1)} I \) and for each row \( i \) of \( A \), \( \| A_i \|_2^2 \leq O(d/n) \). Then

\[
\| x_0 - x^* \|_2 \leq O(\sqrt{d/n}) \| Ax^* - b \|_1 .
\]

**Lemma 4.7.** Let \( A \in \mathbb{R}^{n \times d} \) be such that \( A^TA \approx_{O(1)} I \) and for each row \( i \) of \( A \), \( \| A_i \|_2^2 \leq O(d/n) \). Assume that \( \| b \|_2 \leq n^c \) and \( \| Ax_i - b \|_2 \geq n^{-c} \). Then for any \( e > 0 \), conjugate gradient method can find an \( \tilde{x}_0 \) such that

\[
\| \tilde{x}_0 - x_0 \|_2 \leq \epsilon \| Ax_0 - b \|_2 .
\]

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

**Proof of Lemma 4.7.** We use conjugate gradient with \( e = \sqrt{d/n} \) to find an \( \tilde{x}_0 \) in \( O((t_{A^TA} + d) \log(n/e)) \) time by Lemma 4.7. Note that by definition of \( x_0 \) and by a standard norm inequality, we have:

\[
\| Ax_0 - b \|_2 \leq \| Ax^* - b \|_2 \leq \| Ax^* - b \|_1
\]

Then by the triangle inequality and Lemma 4.6, we have:

\[
\| \tilde{x}_0 - x^* \|_2 \leq \| \tilde{x}_0 - x_0 \|_2 + \| x_0 - x^* \|_2 \\
\leq \sqrt{d/n} \| Ax^* - b \|_1 + O(\sqrt{d/n}) \| Ax^* - b \|_1 .
\]

To prove Lemma 4.6, we use the following lemma:
Lemma 4.8. Let $v \in \mathbb{R}^n$ be a vector with $\|v\|_1 = 1$. Then, for a matrix $A \in \mathbb{R}^{n \times d}$,

$$\|A^Tv\|_2 \leq \max_i \|A_{i,:}\|_2.$$ 

Proof.

$$\|A^Tv\|_2 = \left\| \sum_i A_{i,:}v_i \right\|_2 \leq \max_i \|A_{i,:}\|_2,$$

where the inequality follows by the convexity of $\|\cdot\|_2$ and since $\sum_i |v_i| = 1$. \qed

Proof of Lemma 4.8. By our assumptions on $A$, we have $A^TA + B = I$ for some symmetric $B$ where $\|B\|_2 \leq O(1)$. Since $x_0 = \arg \min_x \|Ax - b\|_2$, we have $x_0 = A^Tb = (A^TA)^{-1}A^Tb$.

Let $v = (Ax^* - b)/\|Ax^* - b\|_1$.

$$\|x_0 - x^*\|_2 = \|(A^TA)^{-1}A^Tb - x^*\|_2 = \|(A^TA)^{-1}A^Tb - (A^TA)^{-1}A^TAx^*\|_2 = \|(A^TA)^{-1}A^T(b - Ax^*)\|_2.$$ 

Now note:

$$\|I + B(A^TA)^{-1}\|_2 \leq \|I\|_2 + \|B\|_2 \|(A^TA)^{-1}\|_2 \leq O(1).$$

Also, by Lemma 4.5 and the assumptions on $A$,

$$\|A^Tv\|_2 \leq O(\sqrt{d/n}).$$

Thus, we have:

$$\|x_0 - x^*\|_2 \leq O(\sqrt{d/n}) \|b - Ax^*\|_1.$$ 

To prove Lemma 4.7 we use the following theorem from [SV14]:

Theorem 4.9 (Theorem 9.1 from [SV14]). Given an symmetric positive definite matrix $M \in \mathbb{R}^{n \times n}$ and a vector $y \in \mathbb{R}^n$, the Conjugate Gradient method can find a vector $x$ such that $\|x - M^{-1}y\|_M \leq \delta \|M^{-1}y\|_M$ in time $O((t_M n + \sqrt{\kappa(M)})\log(1/\delta))$, where $t_M$ is the time required to multiply $M$ with a given vector and $\kappa(M)$ is the condition number of $M$.

Proof of Lemma 4.7. Let $M = A^TA$ and $y = A^Tb$. Then by Theorem 4.9 the conjugate gradient method finds a vector $\tilde{x}_0$ such that $\|\tilde{x}_0\|_{A^TA} \leq \delta \|(A^TA)^{-1}A^Tb\|_{A^TA}$ in time $O((t_{A^TA} + d)\log(1/\delta))$. Noting that $A^TA \approx O(1) I$, we get

$$\|\tilde{x}_0 - x_0\|_2 \leq O(\delta) \|x_0\|_2.$$
Next, we note:

\[ \|b\|_2 \geq \|Ax_0 - b\|_2 \geq \|Ax_0\|_2 - \|b\|_2 \]

\[ \implies \|x_0\|_2 \leq O(\|b\|_2). \]

Now, since we assume that \( \|b\|_2 \leq n^c \) and \( \|Ax_0 - b\|_2 \geq 1/n^c \) for some \( c \), we can set \( \delta = O(\epsilon/(n^c)) \) to get:

\[ \|\tilde{x}_0 - x_0\|_2 \leq \epsilon/n^c \leq \epsilon \|Ax_0 - b\|_2. \]

4.3 Proof of Theorem 1.2

In this section, we show how to use the matrix achieved in Section 4.1 and the initialization from Section 4.2 to achieve fast row-sparsity-preserving \( \ell_1 \) minimization. We will prove the following main theorem:

**Theorem 1.2.** Let \( [A \ b] \) be an \( n \times d \) matrix, \( n \times 1 \) vector pair, such that \( [A \ b]^T [A \ b] \approx_{O(1)} I \) and for each row \( i \) of \( [A \ b], \|[(A b)_i]\|_2^2 \approx_{O(1)} d/n \). Assume \( \min_x \|Ax - b\|_2 \geq n^{-c} \) and \( \|b\|_2 \leq n^c \) for some constant \( c > 0 \). Then for any \( \epsilon > 0 \), there is a routine that computes \( \tilde{x} \) such that with high probability

\[ \|A\tilde{x} - b\|_1 \leq (1 + \epsilon) \|Ax - b\|_1 \]

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

**Proof.** We first prove correctness. We apply Lemma 4.1 to \([A \ b]\) and rescale \([\tilde{A} \ \tilde{b}]\) by \( \sqrt{N/n} \). Note that rescaling does not change the relative error of our output \( \tilde{x} \). From this rescaling, we have \([\tilde{A} \ \tilde{b}]^T [\tilde{A} \ \tilde{b}] \approx_{O(1)} I \) and thus \( \tilde{A}^T \tilde{A} \approx_{O(1)} I \). This implies that \( \tau_i([\tilde{A} \ \tilde{b}]) \approx_{O(1)} \|([\tilde{A} \ \tilde{b}]_i)\|_2^2 \) and \( \tau_i(\tilde{A}) \approx_{O(1)} \|\tilde{A}_i\|_2^2 \). By Fact 5.2 we have \( \tau_i(\tilde{A}) \leq \tau_i([\tilde{A} \ \tilde{b}]) \), so we have \( \|\tilde{A}_i\|_2^2 \leq O(d/N) \) for all rows \( i \). As a result, we can find \( \tilde{x}_0 \) according to Lemma 4.5. The rest of the correctness follows exactly as in the proof of Theorem 1.1.

We now examine the running time and note that uniform sampling will take \( O(n\text{nnz}(A) + d^2 \epsilon^{-2} \log n) \) time to produce \([\tilde{A} \ \tilde{b}] \). By Lemma 4.5 we can then find \( \tilde{x}_0 \) in time \( O(d^2 \epsilon^{-2} \log n) \) because \( \tilde{A} \) is a \( d^{-2} \log^2 n \times d \) matrix, so \( t_{\tilde{A}^T \tilde{A}} = O(d^2 \epsilon^{-2} \log n) \). Finally from the analysis of Theorem 1.1 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 3.2 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 \( \tilde{A} \), which because of our uniform sampling is upper bounded by the maximum number of entries in any row of \( A \). Adding a runtime overhead of \( \log n \) for computing an approximation of the optimal objective, as in Section 6 gives the desired runtime.

5 Preconditioning with Lewis Weights and Rotation

In this section, we show how to precondition a given matrix \( A \in \mathbb{R}^{n \times d} \) into a “good” matrix, primarily using techniques from [CP15], and will ultimately prove Lemma 3.2. Recall that our overall goal was to efficiently transform \( A \) into a matrix \( \tilde{A} \) such that the \( \ell_1 \) norm is approximately maintained for all \( x \), along with \( \tilde{A} \) 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 $A$ according to Lewis weights as in [CP15] to construct a matrix $\tilde{A} \in \mathbb{R}^{N \times d}$. The guarantees of [CP15] ensure that for all $x \in \mathbb{R}^d$, 
$$\|\tilde{A}x\|_1 \approx 1 + \epsilon \|Ax\|_1$$
with high probability. We then further show that this sampling scheme gives $\tau_i(\tilde{A}) = O(d/N)$ for all $1 \leq i \leq N$ with high probability.

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

The matrix $\tilde{A}U$ then has all the prerequisite properties to run our $\ell_1$ minimization algorithms, and it only becomes necessary to show that running an $\ell_1$-minimization routine on $\tilde{A}U$ will help us find an approximate solution to the original problem.

In Section 5.1, we show that Lewis weight sampling gives a matrix with approximately equal leverage scores. In Section 5.2, we find the invertible matrix $U$ that makes $\tilde{A}U$ isotropic while preserving other properties. In Section 5.3, 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 3.2, in Section 5.5.

Before we do this, the following facts are useful.

**Fact 5.1** (Foster’s theorem [Fos53]). For a matrix $A \in \mathbb{R}^{n \times d}$,

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

**Fact 5.2** (Lemma 2 in [CLM+15]). Given a matrix $A$, for all rows $i$,

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

### 5.1 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 $\tilde{A}$ is such that $\tilde{A}^T \tilde{A}$ is spectrally close to $A^T A$. 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 Lemma 5.4 which will be proven at the end of this section.

**Lemma 5.3.** Given a matrix $A \in \mathbb{R}^{n \times d}$ that is sampled according to Theorem 3.1 and gives matrix $\tilde{A}$, then for all rows $i$ of $\tilde{A}$,

$$\tau_i(\tilde{A}) \approx O(1) h(n, \epsilon) \cdot \frac{d}{N}$$

with high probability.

**Proof.** Lemma 5.4 implies that

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

with high probability. Theorem 5.1 implies that every row $i$ of $\tilde{A}$ is simply some row $j$ of $A$, scaled by $\frac{1}{p_j}$. Therefore, for any row $i$ of $\tilde{A}$ we must have

$$\tau_i(\tilde{A}) \approx O(1) h(n, \epsilon) \cdot \tilde{A}_{i,:} \left( A^T W^{-1} A \right)^{-1} \tilde{A}_{i,:}^T = h(n, \epsilon) \cdot \frac{A_{j,:}}{p_j} \left( A^T W^{-1} A \right)^{-1} \frac{A_{j,:}^T}{p_j}.$$
From Definition 2.5 we have
\[ w_j^2 = A_{j,:} \left( A^T W^{-1} A \right)^{-1} A_{j,:}^T, \]
which along with the fact that \( p_j \approx O(1) \) \( w_j \cdot h(n, \epsilon) \) reduces the leverage score to
\[ \tau_i(\tilde{A}) \approx O(1) \frac{1}{h(n, \epsilon)} \]
Finally Fact 5.1 gives us that the sum of Lewis weights must be \( d \) because they are leverage scores of \( W^{-1/2} A \), which implies \( \frac{1}{h(n, \epsilon)} \approx O(1) \) \( h(n, \epsilon) \).

It is now necessary to prove the following key claim used in Lemma 5.3.

**Lemma 5.4.** Given a matrix \( A \) that is sampled according to Theorem 3.1 with error \( \epsilon \) and gives matrix \( \tilde{A} \), then
\[ \tilde{A}^T \tilde{A} \approx O(1) \frac{1}{h(n, \epsilon)} A^T W^{-1} A \]
with high probability.

The proof follows similarly to the proof of Lemma 4 in [CLM+15], 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 [Har12], which is a variant of Corollary 5.2 in [Tro12]:

**Lemma 5.5.** Let \( Y_1 \ldots Y_k \) be independent random positive semidefinite matrices of size \( d \times d \). Let \( Y = \sum_{i=1}^k Y_i \), and let \( Z = \mathbb{E}[Y] \). If \( Y_i \preceq R \cdot Z \) then
\[ \Pr \left[ \sum_{i=1}^k Y_i \preceq (1 - \epsilon) Z \right] \leq de^{-\frac{\epsilon^2}{2}} \]
and
\[ \Pr \left[ \sum_{i=1}^k Y_i \succeq (1 + \epsilon) Z \right] \leq de^{-\frac{\epsilon^2}{3}}. \]

**Proof of Lemma 5.4.** First, we define \( \tilde{A} \) \( \overset{\text{def}}{=} \) \( W^{-1/2} A \). Then, by Definition 2.5 \( w_i = \tau_i(\tilde{A}) \). Since \( W \) is the diagonal matrix of Lewis weights \( w_i \), each row of \( \tilde{A} \) is simply \( \tilde{A}_{i,:} = \frac{1}{p_{j}} \tilde{A}_{i,:} \).

By construction of our random \( \tilde{A} \) in Theorem 3.1 we choose a row \( j \) of \( A \) with probability \( \frac{p_j}{N} \) and scale by \( \frac{1}{p_j} \). Therefore, if we let \( Y_i \) be the random variable
\[ Y_i = \begin{cases} \frac{A_{j,:} A_{j,:}^T}{p_j^2}, & \text{with probability } \frac{p_j}{N} \text{ for each } j \end{cases} \]
then,
\[ Y = \sum_{i=1}^N Y_i = \sum_{i=1}^N \tilde{A}_{i,:} \tilde{A}_{i,:}^T = \tilde{A}^T \tilde{A}. \]

Furthermore, we can substitute \( \tilde{A}_{j,:} \) for \( A_{j,:} \) and use the fact that \( p_j \approx O(1) \) \( w_j \cdot h(n, \epsilon) \) to obtain
\[ \frac{A_{j,:} A_{j,:}^T}{p_j^2} \approx O(1) \frac{\tilde{A}_{j,:} \tilde{A}_{j,:}^T}{p_j \cdot h(n, \epsilon)}. \]
As a result, we have

\[ Z = \mathbb{E} \left[ \sum_{i=1}^{N} Y_i \right] = \sum_{i=1}^{N} \mathbb{E} [ Y_i ] \approx O(1) \sum_{i=1}^{N} \sum_{j=1}^{n} \frac{\overline{A}_{j:} \overline{A}_{j:}^T}{N \cdot h(n, \epsilon)} \]

\[ = \frac{1}{h(n, \epsilon)} \sum_{j=1}^{n} \overline{A}_{j:} \overline{A}_{j:}^T = \frac{1}{h(n, \epsilon)} A^T \overline{W}^{-1} A. \]

In order to apply Lemma 5.5 we need to find \( R \) such that \( Y_i \preceq R \cdot Z \), which by our construction of \( Y_i \) requires

\[ \frac{A_{j:} A_{j:}^T}{p_j^2} \preceq R \cdot Z \]

for all \( j \). We use our constant factor approximations of \( Z \) and \( \frac{A_{j:} A_{j:}^T}{p_j^2} \) to see that it also suffices to show

\[ \frac{\overline{A}_{j:} \overline{A}_{j:}^T}{p_j \cdot h(n, \epsilon)} \preceq \frac{R}{O(1)} \cdot \frac{1}{h(n, \epsilon)} \overline{A}^T \overline{A}. \]

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

\[ \frac{\overline{A}_{j:} \overline{A}_{j:}^T}{p_j \cdot h(n, \epsilon)} \preceq \frac{O(1) \overline{A}_{j:} \overline{A}_{j:}^T}{\tau_j(\overline{A}) \cdot h(n, \epsilon)^2} \]

which along with the fact (Equation 12 in the proof of Lemma 4 from [CLM+15]) that

\[ \frac{\overline{A}_{j:} \overline{A}_{j:}^T}{\tau_j(\overline{A})} \preceq \overline{A}^T \overline{A} \]

implies that

\[ Y_i \preceq \frac{O(1)}{h(n, \epsilon)} Z. \]

By Theorem 3.1 we know that \( h(n, \epsilon) \geq c \epsilon^{-2} \log n \) for some constant \( c \). Plugging this in for \( R \) in Lemma 5.5 gives that

\[ Y \approx_{1+\epsilon} Z \]

or, substituting our values of \( Y \) and \( Z \),

\[ \overline{A}^T \overline{A} \approx O(1) \frac{1}{h(n, \epsilon)} A^T \overline{W}^{-1} A \]

with probability at least \( 1 - 2de^{-\frac{c^2}{4n}} \geq 1 - 2de^{-\frac{c \log n}{O(1)}} \geq 1 - 2dn^{-c/O(1)} \). 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 \( p_i \) approximation of \( \overline{w}_i \cdot h(n, \epsilon) \)) and our assumption on \( n \geq d \). \( \Box \)
5.2 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 5.6. If $U$ is an invertible $d \times d$ matrix and $U^T U = (A^T A)^{-1}$ then

1. $(AU)^T (AU) = I$.
2. For all rows $i$, $\tau_i(A) = \tau_i(AU)$.

Proof. For the first condition, we see that

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

For the second condition, the $i$th row of $A U$ will be $A_i : U$, which by the definition of leverage scores then gives,

$$\tau_i(AU) = A_i : U ((AU)^T (AU))^{-1} (A_i : U)^T = A_i : U U^{-1} (A^T A)^{-1} (U^T)^{-1} U^T A_i^T = A_i : (A^T A)^{-1} A_i^T = \tau_i(A). \quad \square$$

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

Lemma 5.7. Given a full rank matrix $A \in \mathbb{R}^{N \times d}$, there is a routine Rotate that can find an invertible $U$ such that $U^T U = (A^T \tilde{A})^{-1}$ in time $O(Nd^{\omega-1} + d^\omega)$.

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

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 5.8. Given a matrix-vector pair $A \in \mathbb{R}^{N \times d}, b \in \mathbb{R}^N$, another matrix-vector pair $\tilde{A} \in \mathbb{R}^{N \times d}, \tilde{b} \in \mathbb{R}^N$, and an invertible matrix $U \in \mathbb{R}^{d \times d}$,

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

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

\footnote{For an invertible matrix $M \in \mathbb{R}^{d \times d}$, it is easy to see that $M(M^T M)^{-1/2}$ is an orthonormal basis for $M$. We can compute $(M^T M)^{-1}$ using Schur decomposition in $O(d^\omega)$ time, and by careful analysis of that algorithm, we can also compute $(M^T M)^{-1/2}$ in the same amount of time.}
5.3 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 5.9. Given a matrix-vector pair $A \in \mathbb{R}^{n \times d}$, $b \in \mathbb{R}^n$, another matrix-vector pair $\tilde{A} \in \mathbb{R}^{N \times d}$, $\tilde{b} \in \mathbb{R}^N$, and an invertible matrix $U \in \mathbb{R}^{d \times d}$; if

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

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

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

we must have

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

with high probability.

Proof. By assumption we have

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

for all $y \in \mathbb{R}^{d+1}$, and we can then use Lemma 5.8 to obtain

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

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

$$\| \tilde{A} x - \tilde{b} \|_1 \approx_{1+\epsilon} \| A x - b \|_1 \quad \forall x \in \mathbb{R}^d, \quad (2)$$

$$\| \tilde{A} U x - \tilde{b} \|_1 \approx_{1+\epsilon} \| A U x - b \|_1 \quad \forall x \in \mathbb{R}^d. \quad (3)$$

$\text{(2)}$ gives

$$\| A U \tilde{x} - b \|_1 \leq (1 + \epsilon) \| \tilde{A} U \tilde{x} - \tilde{b} \|_1.$$
5.4 Simulated Sampling of $A$

In Lemma 3.2 our primary preconditioning lemma, we set $N$ to be the minimum of $n$ and $O(d^{-2}\log n)$. However, all of our sampling above assumed that $O(d^{-2}\log n)$ rows were sampled to achieve certain matrix concentration results. Accordingly, we will still assume that $O(d^{-2}\log n)$ 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 $\min\{n, O(d^{-2}\log n)\}$. 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 $d^{-2}\log n$.

Ultimately, our proof of Lemma 3.2 will critically use the fact that $\tilde{A}$ has $O(d^{-2}\log n)$ 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 $O(d^{-2}\log n)$ in the running time when $n \ll d^{-2}\log n$.

**Lemma 5.10.** Let $\tilde{A}$ be an $N \times d$ matrix with at most $n$ unique rows, and for each unique row, we are given the number of copies in $\tilde{A}$. Then computing $\tilde{A}^T\tilde{A}$ takes at most $O(nd^{\omega-1})$ time.

**Proof.** By definition

$$\tilde{A}^T\tilde{A} = \sum_i \tilde{A}_i^T\tilde{A}_i.$$ 

Therefore, if we have $k$ copies of row $\tilde{A}_i$, we know that they contribute $k\tilde{A}_i^T\tilde{A}_i$ to the summation. Accordingly, if we replaced all of them with one row $\sqrt{k}\tilde{A}_i$, 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 $\tilde{A}'$ and compute $\tilde{A}'^T\tilde{A}'$ which will be equivalent to $\tilde{A}^T\tilde{A}$.

**Corollary 5.11.** Let $[\tilde{A} \; \tilde{b}]$ be an $N \times (d+1)$ matrix with at most $n$ unique rows, and for each unique row, we are given the number of copies in $[\tilde{A} \; \tilde{b}]$. Then computing $\tilde{A}U$ where $U$ is $d \times d$ matrix, and computing $\tilde{A}^T\tilde{b}$ takes $O(nd^{\omega-1})$ and $O(nd)$ time, respectively.

**Proof.** We can similarly use the fact that $\tilde{A}_i^T\tilde{U}$ is equivalent for all copies of $\tilde{A}_i$ and combine $k$ copies into the row $k\tilde{A}_i$.

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

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

**Theorem 5.12** (Theorem 1.1 in [FCT13]). 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 5.13.** Given a probability distribution $P = (p_1, ..., p_n)$ over a state space of size $n$, sampling $m$ items independently from $P$ 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 Theorem 5.12 for each step to achieve our desired running time.

Furthermore, because \( m \) is decreasing at each iteration, we can use the original preprocessing in Theorem 5.12 for each step to achieve our desired running time.

\[ \square \]

Corollary 5.14. Given \( A \), an \( n \times d \) matrix with a probability distribution over each row, we can produce \( \hat{A} \) according to the distribution that is \( O(de^{-2} \log n) \times d \) in \( O(\min\{de^{-2} \log n, (de^{-2} \log n)^{1/2+o(1)} + n \log^2 n\}) \)-time.

Finally, our application of Theorem 3.12 assumes that it is given an \( N \times d \) matrix, but we assumed that the computational cost could assume \( N = \min\{n, O(de^{-2} \log n)\} \). A closer examination of Algorithm 2 in [All17], which is the routine for Theorem 3.12, 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.

5.5 Proof of Lemma 3.2

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

Lemma 3.2. There is a routine that takes a matrix \( A \in \mathbb{R}^{n \times d} \), a vector \( b \in \mathbb{R}^n \) and \( \epsilon > 0 \), then produces a matrix \( \hat{A}, \hat{b} \in \mathbb{R}^{N \times (d+1)} \) with \( N = O(de^{-2} \log n) \) and an invertible matrix \( U \in \mathbb{R}^{d \times d} \) such that matrix \( \hat{A}U \) is IRB and if \( \hat{x}_U^* \) minimizes \( \| \hat{A}U \hat{x} - \hat{b} \|_1 \), then for any \( \hat{x} \) such that

\[
\| \hat{A}U \hat{x} - \hat{b} \|_1 \leq (1 + \delta) \| \hat{A}U \hat{x}_U^* - \hat{b} \|_1,
\]

we must have

\[
\| A(U \hat{x}) - b \|_1 \leq (1 + \epsilon)^2(1 + \delta) \| A \hat{x}_U^* - b \|_1
\]

with high probability.

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

Proof. From Theorem 3.1 we have that

\[
\| [\hat{A} \hat{b}] y \|_1 \approx_{1+\epsilon} \| A b \|_1
\]

for all \( y \in \mathbb{R}^{d+1} \) with high probability. Lemma 5.9 then gives

\[
\| A(U \hat{x}) - b \|_1 \leq (1 + \epsilon)^2(1 + \delta) \| A \hat{x}_U^* - b \|_1
\]

by our assumption on \( \hat{x} \).

Lemma 5.4 and the assumption that \( A \) is full rank imply that \( \hat{A} \) is full rank with high probability. Our use of \textsc{Rotate} to generate \( U \), such that \( U^T U = (\hat{A}^T \hat{A})^{-1} \), along with Lemma 5.6, gives \( (\hat{A}U)^T \hat{A}U = I \) and also that \( \tau_i(\hat{A}U) = \tau_i(\hat{A}) \) for all \( i \). Fact 5.2 gives \( \tau_i(\hat{A}) \leq \tau_i([A \ b]) \), which

21
along with Lemma 5.3 implies \( \tau_i(\tilde{A}U) \leq O(d/N) \) for all \( i \). Finally, by Definition 2.4 and the fact that \((\tilde{A}U)^T \tilde{A}U = I\), we then have

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

The sampling of \( A \) is done according to [CP15], which requires \( O(nnz(A) \log n + d^2) \) time to obtain the sampling probabilities. Then the actual sampling requires \( O(\min\{de^{-2} \log n, (de^{-2} \log n)^{1/2+o(1)} + n \log^2 n\}) \)-time according to Corollary 5.11. Computing the invertible matrix \( U \) for input \( \tilde{A} \) takes \( O(Nd^{e-1} + d^2) \) time from Lemma 5.7 and the number of rows of \( \tilde{A} \) is \( N = O(d^{-2} \log n) \). Finally, Lemma 5.10 and Corollary 5.11 show that this computation time can also be bounded with \( N \leq n \), which then gives our desired runtime. \( \square \)

6 Approximating the Optimal Objective Value

For ease of notation, we let \( f^* = \min_x \| Ax - b \|_1 \) and \( f_2^* = \min_x \| Ax - b \|_2 \) in this section. In our proof of both Theorem 3.3 and 1.1 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 6.1.** Given a matrix \( A \in \mathbb{R}^{n \times d} \) and a vector \( b \in \mathbb{R}^n \), if \( x_2^* \) minimizes \( \| Ax - b \|_2 \) then

\[
\| Ax_2^* - b \|_2 \leq \| Ax^* - b \|_1 \leq \sqrt{n} \| Ax_2^* - b \|_2.
\]

**Proof.** By known properties of \( \ell_1 \) and \( \ell_2 \), for any \( x \in \mathbb{R}^n \), we have \( \| x \|_2 \leq \| x \|_1 \leq \sqrt{n} \| x \|_2 \). Accordingly, we must have

\[
\| Ax_2^* - b \|_2 \leq \| Ax^* - b \|_2 \leq \| Ax^* - b \|_1,
\]

where the first inequality follows from \( x_2^* \) being the \( \ell_2 \)-minimizer. Similarly, we also have

\[
\| Ax^* - b \|_1 \leq \| Ax_2^* - b \|_1 \leq \sqrt{n} \| Ax_2^* - b \|_2,
\]

where the first inequality follows from \( x^* \) being the \( \ell_1 \)-minimizer. \( \square \)

Since \( A^T A = I \), our initialization of \( A^T b \) is equal to \( x_2^* \). Then we can compute \( \| Ax_2^* - b \|_2 \) in \( O(Nd) \) time. Consequently, if we let \( f_2^* \) be the minimized objective function \( \| Ax - b \|_2 \), we can compute polynomially close upper and lower bounds, \( f_2^* \) and \( \sqrt{n} f_2^* \) respectively, for \( f^* \).

**Lemma 6.2.** In both variants of our primary algorithm for Theorem 3.3 and 1.1 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 \( \| x_0 - x^* \|_2 \) as inputs. Therefore, given a certain \( \tilde{f}^* \) we can input the upper bound \( O(\sqrt{d/n}) f^* \) and following the analysis of the proofs in Theorems 3.3 and 1.1 in runtime \( O(d^2 \epsilon^{-2}) \) and \( O\left( d^2 \epsilon^{-2} \sqrt{\log n} \right) \) respectively, we are guaranteed that we achieve \( \tilde{x} \) such that

\footnote{Note that our sampled and rotated \( \tilde{A}U \) from Lemma 5.2 loses any sparsity guarantees that \( A \) may have had.}
\[ f(\tilde{x}) - f^* \leq \epsilon \tilde{f}^* \] 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 \( f(\tilde{x}) \) 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 \( \tilde{f} = f^* \cdot 2^i \) for \( i = 0 \) to \( \log n \), and whichever iteration produces \( \tilde{x} \) that minimizes \( f(\cdot) \) will be output. Lemma 6.1 implies that there must exist some \( i \) such that \( f^* \cdot 2^i \leq f^* \leq f^* \cdot 2^{i+1} \). Therefore, when we run our algorithm with \( \tilde{f} = f^* \cdot 2^{i+1} \), 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 \( \tilde{x} \) is guaranteed to have \( f(\tilde{x}) - f^* \leq 2\epsilon f^* \).

7 Acknowledgements

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 on an earlier version of this manuscript.

References

[ACW16] Haim Avron, Kenneth L Clarkson, and David P Woodruff. Faster kernel ridge regression using sketching and preconditioning. arXiv preprint arXiv:1611.03220, 2016.

[AH16] Zeyuan Allen Zhu and Elad Hazan. Optimal black-box reductions between optimization objectives. In Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain, pages 1606–1614, 2016.

[All17] 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.

[AMT10] Haim Avron, Petar Maymounkov, and Sivan Toledo. Blendenpik: Supercharging lapack’s least-squares solver. SIAM Journal on Scientific Computing, 32(3):1217–1236, 2010.

[CDMI+13] 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.

[CDS01] Scott Shaobing Chen, David L. Donoho, and Michael A. Saunders. Atomic decomposition by basis pursuit. SIAM Rev., 43(1):129–159, January 2001.

[Cla05] Kenneth L. Clarkson. Subgradient and sampling algorithms for l1 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.
[CLM+15] 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.

[CMMP13] 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.

[CP15] Michael B. Cohen and Richard Peng. Lp 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.

[DBW12] John C Duchi, Peter L Bartlett, and Martin J Wainwright. Randomized smoothing for stochastic optimization. SIAM Journal on Optimization, 22(2):674–701, 2012.

[DDH+09] Anirban Dasgupta, Petros Drineas, Boulos Harb, Ravi Kumar, and Michael W. Mahoney. Sampling algorithms and coresets for lp regression. SIAM J. Comput., 38(5):2060–2078, February 2009.

[DS13] 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.

[FCT13] Martín Farach-Colton and Meng-Tsung Tsai. Exact Sublinear Binomial Sampling, pages 240–250. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[Fos53] F. G. Foster. On the stochastic matrices associated with certain queuing processes. Ann. Math. Statist., 24(3):355–360, 09 1953.

[Har12] 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.

[Lew78] D. Lewis. Finite dimensional subspaces of l_{lp}. Studia Mathematica, 63(2):207–212, 1978.

[LG14] 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.

[LS15] 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.

[MM13a] 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, 2013. ACM.
[Tro12] Joel A. Tropp. User-friendly tail bounds for sums of random matrices. *Found. Comput. Math.*, 12(4):389–434, August 2012. Available at http://arxiv.org/abs/1004.4389.

[Vai89] P. M. Vaidya. Speeding-up linear programming using fast matrix multiplication. In *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, SFCS ’89, pages 332–337, Washington, DC, USA, 1989. IEEE Computer Society.

[Wil12] 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.

[WZ13] David P. Woodruff and Qin Zhang. Subspace embeddings and \((\ell_p,\ell_p)\)-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.

[YCRM16] Jiyan Yang, Yin-Lam Chow, Christopher Ré, and Michael W. Mahoney. Weighted sgd for lp 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.