Computing Approximate $\ell_p$ Sensitivities

Swati Padmanabhan  
Massachusetts Institute of Technology  
pswt@mit.edu

David P. Woodruff  
Carnegie Mellon University  
dwoodruf@cs.cmu.edu

Qiuyi (Richard) Zhang  
Google Research  
qiuyiz@google.com

Abstract

Recent works in dimensionality reduction for regression tasks have introduced the notion of sensitivity, an estimate of the importance of a specific datapoint in a dataset, offering provable guarantees on the quality of the approximation after removing low-sensitivity datapoints via subsampling. However, fast algorithms for approximating sensitivities, which we show is equivalent to approximate regression, are known for only the $\ell_2$ setting, in which they are popularly termed leverage scores. In this work, we provide the first efficient algorithms for approximating $\ell_p$ sensitivities and related summary statistics of a given matrix. In particular, for a given $n \times d$ matrix, we compute $\alpha$-approximation to its $\ell_1$ sensitivities at the cost of $O(n/\alpha)$ sensitivity computations. For estimating the total $\ell_p$ sensitivity (i.e. the sum of $\ell_p$ sensitivities), we provide an algorithm based on importance sampling of $\ell_p$ Lewis weights, which computes a constant factor approximation to the total sensitivity at the cost of roughly $O(\sqrt{d})$ sensitivity computations. Furthermore, we estimate the maximum $\ell_1$ sensitivity, up to a $\sqrt{d}$ factor, using $O(d)$ sensitivity computations. We generalize all these results to $\ell_p$ norms for $p > 1$. Lastly, we experimentally show that for a wide class of matrices in real-world datasets, the total sensitivity can be quickly approximated and is significantly smaller than the theoretical prediction, demonstrating that real-world datasets have low intrinsic effective dimensionality.

1 Introduction

Many modern large-scale machine learning datasets comprise tall matrices $A \in \mathbb{R}^{n \times d}$ with $d$ features and $n \gg d$ training examples, often making it computationally expensive to run them through basic algorithmic primitives. Therefore, many applications spanning dimension reduction, privacy, and fairness aim to estimate the importance of a given datapoint as a first step. Such importance estimates afford us a principled approach to focus our attention on a subset of only the most important examples.

In view of this benefit, several sampling approaches have been extensively developed in the machine learning literature. One of the simplest such methods prevalent in practice is uniform sampling. Prominent examples of this technique include variants of stochastic gradient descent methods \cite{1–4} such as \cite{5–14} developed for the “finite-sum minimization” problem

$$\min_{\theta \in \mathcal{X}} \sum_{i=1}^{n} f_i(\theta). \tag{1.1}$$

where, in the context of empirical risk minimization, each $f_i : \mathcal{X} \to \mathbb{R}_{\geq 0}$ in (1.1) measures the loss incurred by the $i$-th data point from the training set, and in generalized linear models, each $f_i$ represents a link function applied to a linear predictor evaluated at the $i$-th data point. The aforementioned algorithms randomly sample a function $f_i$ and make progress via gradient estimation techniques.

However, uniform sampling can lead to significant information loss when the number of important training examples is relatively small. Hence, importance sampling methods that assign higher sampling probabilities to more important examples have emerged both in theory [15, 16] and practice [17–21]. One such technique is based on sensitivity scores, defined [15] for each coordinate $i \in [n]$ as:

$$\sigma_i \overset{\text{def}}{=} \max_{x \in \mathcal{X}} \frac{f_i(x)}{\sum_{j=1}^{n} f_j(x)}.$$  \hspace{1cm} (1.2)

Sampling each $f_i$ independently with probability $p_i \propto \sigma_i$ and scaling the sampled row with $1/p_i$ preserves the objective in (1.1) in expectation for every $x \in \mathcal{X}$. Further, one gets a $(1 + \varepsilon)$-approximation to this objective by sampling $\mathcal{O}(p / (\varepsilon n))$ functions [22, Theorem 1.1], where $\mathcal{S} = \sum_{i=1}^{n} \sigma_i$ is called the total sensitivity, and $d$ is an associated VC dimension.

Numerous advantages of sensitivity sampling over $\ell_p$ Lewis weight sampling [23] were highlighted in the recent work [24], prominent among which is that a sampling matrix built using the $\ell_p$ sensitivities of $A$ is significantly smaller than one built using its $\ell_p$ Lewis weights in many important cases, e.g., when the total sensitivity $\mathcal{S}$ is small for $p > 2$ (which is seen in many structured matrices, such as combinations of Vandermonde, sparse, and low-rank matrices as studied in [25] as well as those in many naturally occurring datasets (cf. Section 4)). Additionally, sensitivity sampling has found use in constructing $\varepsilon$-approximators for shape-fitting functions, also called strong coresets [15, 16, 26–28], clustering [16, 29, 30], logistic regression [31, 32], and least squares regression [33, 34], Despite their benefits, the computational burden of estimating these sensitivities has been known to be significant, requiring solving an expensive maximization problem for each datapoint, which is a limiting factor in their use [35]. A key exception is for $f_i(x) = (a_i^T x)^2$, when the sensitivity score of $f_i$ is exactly the leverage score of $a_i$, which can be computed quickly in $O(\log(n))$ sensitivity calculations, as opposed to the naive $O(n)$ sensitivity calculations (cf. Section 2).

### 1.1 Our contributions

In this work, we initiate a systematic study of algorithms for approximately computing the $\ell_p$ sensitivities of a matrix, generally up to a constant factor. We first define $\ell_p$ sensitivities:

**Definition 1.1.** [15] Given a full-rank matrix $A \in \mathbb{R}^{n \times d}$ with $n \geq d$ and a scalar $p \in (0, \infty)$, let $a_i$ denote the $i$’th row of matrix $A$. Then the vector of $\ell_p$ sensitivities of $A$ is defined as\footnote{From hereon, for notational conciseness, we omit stating $Ax \neq 0$ in the problem constraint.} (cf. Section 2 for the notation)

$$\sigma_p(a_i) \overset{\text{def}}{=} \max_{x \in \mathbb{R}^d, Ax \neq 0} \frac{|a_i^T x|^p}{\|Ax\|_p^p} \text{ for all } i \in [n],$$  \hspace{1cm} (1.3)

and the total $\ell_p$ sensitivity is defined as $\mathcal{S}_p(A) \overset{\text{def}}{=} \sum_{i=1}^{n} \sigma_p(a_i)$.

These $\ell_p$ sensitivities yield sampling probabilities for $\ell_p$ regression, a canonical optimization problem that captures least squares regression, maximum flow, and linear programming, in addition to appearing in applications like low rank matrix approximation [36], sparse recovery [37], graph-based semi-supervised learning [38–40], and data clustering [41]. However, algorithms for approximating $\ell_p$ sensitivities were known for only $p = 2$ (when they are called “leverage scores”). As is typical in large-scale machine learning, we assume $n \gg d$. We now state our contributions, which are three-fold.

1. **Approximation algorithms.** We design algorithms for three prototypical computational tasks associated with $\ell_p$ sensitivities of a matrix $A \in \mathbb{R}^{n \times d}$. Specifically, for an approximation parameter $\alpha > 1$, we can simultaneously estimate all $\ell_p$ sensitivities up to an additive error of $O\left(\frac{\varepsilon}{\alpha} \mathcal{S}_p(A)\right)$ via $O(n/\alpha)$ individual sensitivity calculations, reducing our runtime by sacrificing some accuracy. We limit most of our results in the main text to $\ell_1$ sensitivities, but extend these to all $p \geq 1$ in Appendix C.

To state our results for the $\ell_p$ setting, we use $\text{nnz}(A)$ to denote the number of non-zero entries of the matrix $A$, $\omega$ for the matrix multiplication constant, and we introduce the following piece of notation.

**Definition 1.2 (Notation for $\ell_p$ Results).** We introduce the notation $\text{LP}(m, d, p)$ to denote the cost of approximating one $\ell_p$ sensitivity of an $m \times d$ matrix up to an accuracy of a given constant factor.
Theorem 1.3 (Estimating all sensitivities; informal version of Theorem 3.1 and Theorem C.2). Given a matrix $A \in \mathbb{R}^{n \times d}$ and $1 \leq \alpha \ll n$, there exists an algorithm (Algorithm 1) which returns a vector $\tilde{\sigma} \in \mathbb{R}^{\geq 0}$ such that, with high probability, for each $i \in [n]$, we have

$$\sigma_1(a_i) \leq \tilde{\sigma}_i \leq O(\sigma_1(a_i) + \frac{\alpha}{n} \|S_1(A)\|).$$

Our algorithm’s runtime is $\tilde{O}(\text{nnz}(A) + \frac{\alpha}{n} \cdot d^3)$. Moreover, for $p \geq 1$, we can compute (via Algorithm 6) an estimate $\tilde{\sigma} \in \mathbb{R}^{\geq 0}$ satisfying $\sigma_p(a_i) \leq \tilde{\sigma}_i \leq O(\alpha^{p-1} \sigma_p(a_i) + \frac{\alpha}{n} \|S_p(A)\|)$ with high probability for each $i \in [n]$, at a cost of $O(\text{nnz}(A) + \frac{\alpha}{n} \cdot \text{LP}(O(d^{\max(1,p/2)}, d, p))$.

Two closely related properties arising widely in algorithms research are the total sensitivity $\mathcal{S}_1(A)$ and the maximum sensitivity $\|\sigma_1(A)\|_\infty$. As stated earlier, one important feature of the total sensitivity is that it determines the total sample size needed for function approximation, thus making its efficient estimation crucial for fast $\ell_1$ regression. Additionally, it was shown in [24] that the sample complexity of sensitivity sampling is much lower than that of Lewis weight sampling in many practical applications such as when the total sensitivity is small; therefore, an approximation of the total sensitivity can be used as a fast test for whether or not to proceed with sensitivity sampling (involving the costly task of calculating all sensitivities).

While the total sensitivity helps us characterize the whole dataset (as explained above), the maximum sensitivity captures the importance of the most important datapoint and is used in, e.g., experiment design and in reweighting matrices for low coherence [42]. Additionally, it captures the maximum extent to which a datapoint can influence the objective function and is therefore used in differential privacy [43]. While one could naively estimate the total and maximum sensitivities by computing all $n$ sensitivities using Theorem 1.3, we give faster algorithms that, strikingly, have no polynomial dependence on $n$. In the assumed regime of $n \gg d$, this is a significant runtime improvement.

Theorem 1.4 (Estimating the total sensitivity; informal version of Theorem 3.5). Given a matrix $A \in \mathbb{R}^{n \times d}$, a scalar $p \geq 1$, and a small constant $\gamma < 1$, there exists an algorithm (Algorithm 2) which returns a positive scalar $\tilde{s}$ such that, with a probability of at least 0.99, we have

$$\mathcal{S}_p(A) \leq \tilde{s} \leq (1 + O(\gamma)) \mathcal{S}_p(A).$$

Our algorithm’s runtime is $\tilde{O}\left(\text{nnz}(A) + \frac{1}{\gamma^2} \cdot d^{p/2 - 1} \cdot \text{LP}(O(d^{\max(1,p/2)}, d, p))\right)$.

In Algorithm 4 (presented in Appendix B.2), we give an alternate method to estimate $\mathcal{S}_1(A)$ based on recursive leverage score sampling, without $\ell_p$ Lewis weight calculations, which we believe to be of independent interest. All of our algorithms utilize approximation properties of $\ell_p$ Lewis weights and subspace embeddings and recent developments in fast computation of these quantities.

Theorem 1.5 (Estimating the maximum sensitivity; informal version of Theorem 3.7 and Theorem C.3). Given a matrix $A \in \mathbb{R}^{n \times d}$, there exists an algorithm (Algorithm 3) which returns a scalar $\tilde{s} > 0$ such that

$$\|\sigma_1(A)\|_\infty \leq \tilde{s} \leq O(\sqrt{d} \|\sigma_1(A)\|_\infty).$$

Our algorithm’s runtime is $\tilde{O}(\text{nnz}(A) + d^{p+1})$. Furthermore, via Algorithm 7, this result holds for all $p \in [1, 2]$ and for $p > 2$, this guarantee costs $\tilde{O}(\text{nnz}(A) + d^{p/2} \cdot \text{LP}(O(d^{p/2}), d, p))$.

(2) Hardness results. In the other direction, while it is known that $\ell_p$ sensitivity calculation reduces to $\ell_p$ regression, we show the converse. Specifically, we establish hardness of computing $\ell_p$ sensitivities in terms of the cost of solving multiple corresponding $\ell_p$ regression problems, by showing that a $(1 + \varepsilon)$-approximation to $\ell_p$-sensitivities solves $\ell_p$ multiregression up to an accuracy factor of $1 + \varepsilon$.

Theorem 1.6 (Leave-One-Out $\ell_p$ Multiregression Reduces to $\ell_p$ Sensitivities; informal). Suppose that we are given an sensitivity approximation algorithm $A$, which for any matrix $A' \in \mathbb{R}^{n' \times d}$ and accuracy parameter $\varepsilon' \in (0, 1)$, computes $(1 \pm \varepsilon')\mathcal{S}_p(A')$ in time $T(n', d', \text{nnz}(A'), \varepsilon')$.

Given a matrix $A \in \mathbb{R}^{n \times d}$ with $n \geq d$, let $OPT_i := \min_{y \in \mathbb{R}^{d-1}} \|A_{\cdot i} - y \|_2$ and $y^*_i := \arg\min_{y \in \mathbb{R}^{d-1}} \|A_{\cdot i} - y \|_2$ for all the $i \in [d]$. Then, there exists an algorithm that takes $A \in \mathbb{R}^{n \times d}$ and computes $(1 \pm \varepsilon)OPT_i$ for all $i$ in time $T(n + d, d, \text{nnz}(A), \varepsilon)$.

\textsuperscript{2}Please see Lemma D.2 for the formal version with a small additive error.
For $p \neq 2$, approximating all $\ell_p$ sensitivities to a high accuracy would therefore require improving $\ell_p$ multi-regression algorithms with $\Omega(d)$ instances, for which the current best high-accuracy algorithms take $\text{poly}(d) \text{nnz}(A)$ time [44, 45]. More concretely, our hardness results imply that in general one cannot compute all sensitivities as quickly as leverage scores unless there is a major breakthrough in multi-response regression. In order to show this, we introduce a reduction algorithm that efficiently regresses columns of $A$ against linear combinations of other columns of $A$, by simply adding a row to $A$ capturing the desired linear combination and then computing sensitivities, which will reveal the cost of the corresponding regression problem. We can solve $\Theta(d)$ of these problems by augmenting the matrix to increase the rows and columns by at most a constant factor. We defer these details to Appendix A.

**1.2 Related work**

Introduced in the pioneering work of [15], sensitivity sampling falls under the broad framework of “importance sampling”. It has found use in constructing $\epsilon$-approximators of numerical integrands of several function classes in numerical analysis, statistics (e.g., in the context of VC dimension), and computer science (e.g., in coresets for clustering). The results from [15] were refined and extended by [16], with more advances in the context of shape-fitting problems in [26–28], clustering [16, 29, 30], logistic regression [31, 32], least squares regression [33, 34, 47], principal component analysis [48], reinforcement learning [49, 50], and pruning of deep neural networks [51–53]. Sampling algorithms for $\ell_p$ subspace embeddings have also been extensively studied in functional analysis [54–59] as well as in theoretical computer science [23, 60, 44, 45].

Another notable line of work in importance sampling uses Lewis weights to determine sampling probabilities. As we explain in Section 2, Lewis weights are closely related to sensitivities. Many modern applications of Lewis weights in theoretical computer science we introduced in [23], which gave input sparsity time algorithms for approximating Lewis weights and used them to obtain fast algorithms for solving $\ell_p$ regression. They have subsequently been used in row sampling algorithms for data pre-processing [33, 61, 62, 60, 23], computing dimension-free strong coresets for $k$-median and subspace approximation [63], and fast tensor factorization in the streaming model [64]. Lewis weights are also used for $\ell_1$ regression in: [65] for stochastic gradient descent pre-conditioning, [66] for quantile regression, and [67] to provide algorithms for linear algebraic problems in the sliding window model. Lewis weights have also become widely used in convex geometry [68], randomized numerical linear algebra [23, 42, 69, 64, 70], and machine learning [71–73].

**2 Notation and preliminaries**

We use boldface uppercase and lowercase letters for matrices and vectors respectively. We denote the $i$th row vector of a matrix $A$ by $a_i$. Given a matrix $M$, we use $\text{Tr}(M)$ for its trace, $\text{rank}(M)$ for its rank, $M^\dagger$ for its Moore-Penrose pseudoinverse, and $|M|$ for the number of its rows. When $x$ is an integer, we use $[x]$ for the set of integers $1, 2, \ldots, x$. For two positive scalars $x$ and $y$ and a scalar $\alpha \in (0, 1)$, we use $x \approx_{\alpha} y$ to denote $(1 - \alpha)y \leq x \leq (1 + \alpha)y$; we sometimes use $x \approx y$ to mean $(1 - c)y \leq x \leq (1 + c)y$ for some appropriate universal constant $c$. We use $\tilde{O}$ to hide dependence on $\alpha^{(1)}$. We acknowledge that there is some notation overloading between $p$ (when used to denote the scalar in, for example, $\ell_p$ norms) and $p_i$ (when used to denote some probabilities) but the distinction is clear from context. We defer the proofs of facts stated in this section to Appendix A.

**Notation for sensitivities.** We start with a slightly general definition of sensitivities: $\sigma_p^B(a_i)$ to denote the $\ell_p$ sensitivity of $a_i$ with respect to $B$; when computing the sensitivity with respect to the
While $\sigma$ when referring to the vector of $\ell_p$ sensitivities of all the rows of a matrix, we omit the subscript in the argument: so, $\sigma^B_p(A)$ is the vector of $\ell_p$ sensitivities of rows of the matrix $A$, each computed with respect to the matrix $B$ (as in Equation (2.1)), and analogously, $\sigma_p(A)$ is the vector of $\ell_p$ sensitivities of the rows of matrix $A$, each computed with respect to itself. Similarly, the total sensitivity is the sum of $\ell_p$ sensitivities of the rows of $A$ with respect to matrix $B$, for which we use the following notation:

$$\sigma^B_p(A) := \sum_{i \in [n]} \sigma^B_p(a_i)$$

$$\sigma_p(A) := \sum_{i \in [n]} \sigma_p(a_i),$$

where, analogous to the rest of the notation regarding sensitivities, we omit the superscript when the sensitivities are being computed with respect to the input matrix itself.

While $\sigma^B_p(a_i)$ could be infinite, by appending $a_i$ to $B$, the generalized sensitivity becomes once again contained in $[0, 1]$. Specifically, a simple rearrangement of the definition gives us the identity $\sigma^B_{p, a}(a_i) = 1/(1 + \sigma^B_p(a_i))$. We now define leverage scores, which are a special type of sensitivities and also amenable to efficient computation [60], which makes them an ideal “building block” in algorithms for approximating sensitivities.

**Leverage scores are $\ell_2$ sensitivities.** The leverage score of the $i^{th}$ row $a_i$ of $A$ is $\tau_i(A) \overset{\text{def}}{=} a_i^T(A^T A)^{-1} a_i$. The $i^{th}$ leverage score is also the $i^{th}$ diagonal entry of the orthogonal projection matrix $A(A^T A)^{-1} A^T$. Since the eigenvalues of an orthogonal projection matrix are zero or one, it implies $\tau_i(A) = 1 | A(A^T A)^{-1} A^T 1_i | \leq 1$ for all $i \in [n]$ and $\sum_{i=1}^n \tau_i(A) \leq d$ (see [74]). It turns out that $\tau_i(A) = \sigma_2(a_i)$ (see [75]). This may be verified by applying a change of basis to the variable $y = Ax$ in the definition of sensitivity and working out that the maximizer is the vector parallel to $y = A^T a_i$. This also gives an explicit formula for the total $\ell_2$ sensitivity: $\Sigma_2(A) = \text{rank}(A)$.

The fastest algorithm for constant factor approximation to leverage scores is by [60], as stated next.

**Fact 2.1 ([60, Lemma 7]).** Given a matrix $A \in \mathbb{R}^{n \times d}$, we can compute constant-factor approximations to all its leverage scores in time $O(\text{nnz}(A) \log(n) + d^\omega \log(d))$.

In this paper, the above is the result we use to compute leverage scores (to a constant accuracy). Indeed, since the error accumulates multiplicatively across different steps of our algorithms, a high-accuracy algorithm for leverage score computation does not serve any benefit over this constant-accuracy one. Leverage scores approximate $\ell_2$ sensitivities up to a distortion factor of $\sqrt{n}$, as seen from the below known fact we prove, for completeness, in Appendix A.

**Fact 2.2 (Crude sensitivity approximation via leverage scores).** Given a matrix $A \in \mathbb{R}^{n \times d}$, let $\sigma_i(a_i)$ denote the $i^{th}$ $\ell_1$ sensitivity of $A$ with respect to $A$, and let $\tau_i(A)$ denote its $i^{th}$ leverage score with respect to $A$. Then we have $\sqrt{\tau_i(A)} n \leq \sigma_i(a_i) \leq \sqrt{\tau_i(A)}$.

In general, $\ell_p$ sensitivities of rows of a matrix suffer a distortion factor of at most $n^{1/2-1/p}$ from corresponding leverage scores, which follows from Hölder’s inequality. The $\ell_p$ generalizations of leverages scores are called $\ell_p$ Lewis weights [54, 23], which satisfy the recurrence $w_i = \tau_i(W^{1/2-1/p} A)$. However, unlike when $p = 2$, $\ell_p$ Lewis weights are not equal to sensitivities, and from an approximation perspective, they provide only a one-sided bound on the sensitivities $\sigma^A_p(a_i) \leq d^\max(0, p/2-1) w^A_p(a_i)$ [75, 76]. While we have algorithms for efficiently computing leverage scores [60] and Lewis weights [23, 77], extensions for utilizing these ideas for fast and accurate sensitivity approximation algorithms for all $p$ have been limited, which is the gap this paper aims to fill. A key regime of interest for many downstream algorithms is a small constant factor approximation to true sensitivities, such as for subsampling [78].

**Subspace embeddings.** We use sampling-based constant-approximation $\ell_p$ subspace embeddings that we denote by $S_p$. When the type of embedding is clear from context, we omit the subscript.
**Definition 2.3 (ℓ_p Subspace Embedding).** Given $A \in \mathbb{R}^{n \times d}$ (typically $n \gg d$), we call $S \in \mathbb{R}^{r \times n}$ (typically $d \leq r \ll n$) an $\epsilon$-approximate ℓ_p subspace embedding of $A$ if $\|SAx\|_p \approx \|Ax\|_p \forall x \in \mathbb{R}^d$.

When the diagonal matrix $S$ is defined as $S_{ii} = 1/\sqrt{p_i}$ with probability $p_i = \Omega(\tau_i(A))$ (and 0 otherwise), then with high probability $SA$ is an ℓ_2 subspace embedding for $A$ [33, 79]; further, $S$ has at most $O(d\epsilon^{-2})$ non-zero entries. For general $p$, we use Lewis weights to construct ℓ_p subspace embeddings fast [23, Theorem 7.1]. This fast construction has been made possible by the reduction of Lewis weight computation to a few (polynomial in $p$) leverage score computations for $p < 4$ [23] as well as $p \geq 4$ [77]. This efficient construction is the reason for our choice of this subspace embedding. We note that there has been an extensive amount of literature on designing ℓ_1 subspace embeddings for example using Cauchy [80] and exponential [81] random variables.

For $p \leq 2$, it is known that the expected sample complexity of rows is $\sum_{i=1}^{n} p_i = O(d)$ [34], and for $p > 2$, this is known to be at most $O(d^{p/2})$ [35]. While these bounds are optimal in the worst case, one important application of faster sensitivity calculations is to compute the minimal subspace embedding dimension for average case applications, particularly those seen in practice. This gap is captured by the total sensitivity $\tilde{\mathcal{S}}_p(A)$, which has been used to perform more sample efficient subspace embeddings [67, 27, 28], as sampling $\tilde{O}(\tilde{\mathcal{S}}_p(A)d/\epsilon^2)$ rows suffice to provide an ℓ_p subspace embedding.

**Accurate sensitivity calculations.** While leverage scores give a crude approximation, constant-factor approximations of ℓ_p sensitivities seem daunting since we must compute worst-case ratios over the input space. However, we can compute the cost of one sensitivity specific row by reducing it to ℓ_p regression since by scale invariance, we have $\frac{1}{\sigma_{i}^{(p)}(a_{i})} = \min_{x \in \mathbb{R}^d} \|Bx\|_p$.

This reduction allows us to use recent developments in fast approximate algorithms for ℓ_p regression [44, 25, 45], which utilize subspace embeddings to first reduce the number of rows of the matrix to $O(poly(d))$. For the specific case of $p = 1$, which is the focus of the main body of our paper, we may reduce to approximate ℓ_1 regression on a $O(d) \times d$ matrix, which is a form of linear programming.

**Fact 2.4.** Given an $n \times d$ matrix, the cost of computing $k$ of its ℓ_1 sensitivities is $\tilde{O}(\text{nnz}(A) + k \cdot d^2)$.

### 3 Approximating functions of sensitivities

We first present a constant-probability algorithm approximating the ℓ_1 sensitivities in **Algorithm 1**. Our algorithm is a natural one: Since computing the ℓ_1 sensitivities of all the rows simultaneously is computationally expensive, we instead hash α-sized blocks of random rows into smaller (constant-sized) buckets and compute the sensitivities of the smaller, $O(n/\alpha)$-sized matrix $P$ so generated, computing the sensitivities with respect to $SA$, the ℓ_1 subspace embedding of $A$. Running this algorithm multiple times gives our high-probability guarantee via the standard median trick.

**Theorem 3.1.** Given a full-rank matrix $A \in \mathbb{R}^{n \times d}$ and an approximation factor $1 < \alpha \ll n$, let $\sigma_1(a_i)$ be the ℓ_1 sensitivity of the $i^{th}$ row of $A$. Then there exists an algorithm that, in time $O(\text{nnz}(A) + \frac{n}{\alpha} \cdot d^2)$, returns $\tilde{\sigma} \in \mathbb{R}_{\geq 0}^n$ such that with high probability, for each $i \in [n]$,

$$\sigma_1(a_i) \leq \tilde{\sigma}_i \leq O(\sigma_1(a_i) + \frac{n}{\alpha} \tilde{\mathcal{S}}_1(A)).$$

**Proof Sketch of Theorem 3.1; full proof in Appendix B.1.** We achieve our guarantee via **Algorithm 1**. To see this, first note that $\|SAx\|_1 \approx \Theta(\|Ax\|_1)$ (since as per **Line 1**, $SA$ is an ℓ_1 subspace embedding of $A$). Combining this with the fact that every row in $A$ is mapped, via the designed randomness, to some row in the matrix $P$ helps establish the desired approximation guarantee. The runtime follows from using **Fact 2.4** to compute ℓ_1 sensitivities with respect to $SA$ of $|P| = O(n/\alpha)$ rows.

As seen in **Appendix C.1**, our techniques described above also generalize to the $p \geq 1$ case. Specifically, in **Theorem C.2**, we show that reducing ℓ_p sensitivity calculations by an $\alpha$ factor gives an approximation guarantee of the form $O(\alpha^{p-1} \sigma_p(a_i)) + \frac{n}{\alpha} \tilde{\mathcal{S}}_p(A))$.

**Remark 3.2.** The sensitivities returned by our algorithm are approximate, with relative and additive error, but are useful in certain settings as they preserve ℓ_p regression approximation guarantees while increasing total sample complexity by only a small poly($d$) factor compared to true sensitivities while still keeping it much smaller than that obtained via Lewis weights. To see this with a simple toy...
Algorithm 1 Approximating $\ell_1$-Sensitivities: Row-wise Approximation

**Input:** Matrix $A \in \mathbb{R}^{n \times d}$, approximation factor $\alpha \in (1, n)$

**Output:** Vector $\tilde{\sigma} \in \mathbb{R}^n_{>0}$ that satisfies, for each $i \in [n]$, with probability 0.9, that

$$\sigma_1(a_i) \leq \tilde{\sigma}_i \leq O(\sigma_1(a_i) + \tilde{S}_1(A) \frac{\alpha}{d})$$

1: Compute for $A$ an $\ell_1$ subspace embedding $SA \in O(d) \times d$ (cf. Definition 2.3)
2: Partition $A$ into $\frac{n}{\alpha}$ blocks $B_1, \ldots, B_{n/\alpha}$, each containing $\alpha$ randomly selected rows
3: for the $\ell^{th} \text{ block } B_i$, with $\ell \in \left[ \frac{n}{\alpha} \right]$ do
4: Sample $\alpha$-dimensional independent Rademacher vectors $r_1^{(\ell)}, \ldots, r_{100}^{(\ell)}$
5: For each $j \in [100]$, compute the row vectors $r_j^{(\ell)} B_i \in \mathbb{R}^d$
6: end for
7: Let $P \in \mathbb{R}^{100n \times d}$ be the matrix of all vectors from Line 5. Compute $\sigma_1^{SA}(P)$ using Fact 2.4
8: for each $i \in [n]$ do
9: Denote by $J$ the set of row indices in $P$ that $a_i$ is mapped to in Line 5
10: Set $\tilde{\sigma}_i = \max_{j \in J}(\sigma_1^{SA}(p_j))$
11: end for
12: Return $\tilde{\sigma}$

Example, consider the case $p > 2$. Note that by Theorem 1.5 of [24], the sample complexity using approximate sensitivities is $\tilde{O}(\alpha^{2n-2} \tilde{S}_p^{-2/2p}(A))$, using the $\alpha^p$ factor blowup via Theorem C.2, that using true sensitivities is $\tilde{O}(\tilde{S}_p^{-2/2p}(A))$, and that using $\ell_p$ Lewis weights is $O(d^{p/2})$. Suppose also that the total sensitivity $\tilde{S}_p(A) = d$. Further assume that $n = d^{10}$ and $\alpha = n^{10} = d^{1/p}$; then the dimension-dependence of sample complexities given by our approximate sensitivities, true sensitivities, and Lewis weights are, respectively, $O(d^4), O(d^2)$, and $O(d^{p/2})$ (ignoring small factors in the exponent that do not affect the asymptotic analysis). Thus, our approximate sensitivities provide a tradeoff in computational cost and sample complexity between using true sensitivities and Lewis weights.

### 3.1 Estimating the sum of $\ell_p$ sensitivities

In this section, we present a one-shot algorithm that provides a constant-approximation to the total $\ell_p$ sensitivity for all $p \geq 1$. This algorithm is based on importance sampling using $\ell_p$ Lewis weights. For our desired guarantees, we crucially need the approximation bounds of $\ell_p$ Lewis weights, as provided by Fact 3.3, efficient computation of $\ell_p$ Lewis weights as provided by [23] for $p < 4$ and by [77] for $p \geq 4$, and the fact that a random $\ell_p$ sampling matrix $S$ (cf. Fact 3.4) yields $SA$, a constant-approximation subspace embedding to $A$ [24, Theorem 1.8].

Algorithm 2 Approximating the Sum of $\ell_p$-Sensitivities for $p \geq 1$

**Input:** Matrix $A \in \mathbb{R}^{n \times d}$, approximation factor $\gamma \in (0.01, 1)$, and a scalar $p \geq 1$

**Output:** A positive scalar that satisfies, with probability 0.99, that

$$\tilde{S}_p(A) \leq \hat{s} \leq (1 + O(\gamma))\tilde{S}_p(A)$$

1: Compute all $\ell_p$ Lewis weights of $A$, denoting the $i^{th}$ weight by $w_p(a_i)$
2: Define the sampling vector $v \in \mathbb{R}^n_{>0}$ by $v_i = \frac{w_p(a_i)}{d}$ for all $i \in [n]$
3: Sample $m = O(d^{1-p/2})$ rows with replacement, picking the $i^{th}$ row with a probability of $v_i$
4: Construct a random $\ell_p$ sampling matrix $S_p A$ with probabilities $\{v_i\}_{i=1}^n$ (cf. Fact 3.4)
5: For each sampled row $i_j, j \in [m], \text{ compute } r_j = \frac{\sigma_p^{S_p A}(a_{i_j})}{v_{i_j}}$
6: Return $\frac{1}{m} \sum_{j=1}^m r_j$

**Fact 3.3** (Lewis Weights Bounds Sensitivities, [42, 82]). Given a matrix $A \in \mathbb{R}^{n \times d}$ and $p \geq 1$, the $\ell_p$ sensitivity $\sigma_p^A(a_i)$ and $\ell_p$ Lewis weight $w_p(a_i)$ of the row $a_i$ satisfy, for all $i \in [n],$

$$\sigma_p^A(a_i) \leq d^{\max(0, p/2 - 1)} w_p(a_i).$$
Fact 3.4 (Sampling via Lewis Weights [23, 35]). Given \( A \in \mathbb{R}^{n \times d} \) and \( p > 0 \). Consider a random diagonal \( \ell_p \) sampling matrix \( S \in \mathbb{R}^{n \times n} \) with sampling probabilities \( \{p_i\} \) proportional to the \( \ell_p \) Lewis weight of \( A \), i.e., for each \( i \in [n] \), the \( i \)th diagonal entry is independently set to be

\[
S_{i,i} = \begin{cases} 
1/p_i^{1/p} & \text{with probability } p_i \\
0 & \text{otherwise}
\end{cases}
\]

Then, with high probability, \( S \) with \( O(\varepsilon^{-2}d^{\max(1,p/2)}(\log d)^2 \log(d/\varepsilon)) \) rows is an \( \ell_p \) subspace embedding for \( A \) (cf. Definition 2.3).

Theorem 3.5. Given a matrix \( A \in \mathbb{R}^{n \times d} \) and an approximation factor \( \gamma \in (0, 1) \), there exists an algorithm, which returns a positive scalar \( \hat{s} \) such that, with a probability 0.99, we have

\[ \mathcal{E}_p(A) \leq \hat{s} \leq (1 + O(\gamma))\mathcal{E}_p(A) . \]

Our algorithm’s runtime is \( \tilde{O}(mnz(A) + \frac{\gamma}{\varepsilon} \cdot d^{p/2-1}L\tilde{p}(O(d^{\max(1,p/2)}, d, p)) \).

Proof. Without loss of generality, we may assume \( A \) to be full rank. Then, its Lewis weights satisfy \( \sum_{i=1}^{n} w_p(a_i) = d \). Per Line 2 of Algorithm 2, our sampling distribution is chosen to be \( v_i = \frac{w_p(a_i)}{m} \) for all \( i \in [n] \). We sample from \( A \) rows with replacement, with row \( i \) picked with a probability of \( v_i \). From the definition of \( v_i \) in Line 2 and \( r_j \) in Line 5 and the fact that \( S_pA \) is a constant factor \( \ell_p \) subspace embedding of \( A \), our algorithm’s output satisfies the following unbiasedness condition:

\[ \mathbb{E} \left( \frac{1}{m} \sum_{j[m]} r_j \right) = \mathbb{E} \left( \frac{1}{m} \sum_{j=1}^{m} \sum_{i=1}^{n} \frac{\sigma_{p,A}(a_{ij})}{v_j} \right) = \mathcal{E}_p(A). \]

By independence, we also have the following bound on the variance of \( \frac{1}{m} \sum_{j[m]} r_j \):

\[ \text{Var} \left( \frac{1}{m} \sum_{j[m]} r_j \right) \leq \frac{1}{m} \sum_{i=1}^{n} \frac{\sigma_{p,A}(a_i)^2}{v_i} = \frac{d}{m} \sum_{i=1}^{n} \frac{\sigma_{p,A}(a_i)^2}{w_p(a_i)}, \]

with the final step holding by the choice of \( v_i \) in Line 2. In the case that \( p \geq 2 \), we have

\[ \text{Var} \left( \frac{1}{m} \sum_{j[m]} r_j \right) \leq \frac{d}{m} \sum_{i=1}^{n} \frac{\sigma_{p,A}(a_i)^2}{w_p(a_i)} \leq \frac{d}{m} \sum_{i=1}^{n} \sigma_{p,A}(a_i) \cdot d^{p/2-1} \leq \frac{2d^{p/2}}{m} \mathcal{E}_p(A), \]

where the first inequality uses Fact 3.3. Therefore, applying Chebyshev’s inequality on \( \frac{1}{m} \sum_{j[m]} r_j \) (as defined in Line 5) with \( m = O \left( \frac{d^{p/2}}{\mathcal{E}_p(A)^2} \right) \) gives us a \( \gamma \)-multiplicative accuracy in approximating \( \mathcal{E}_p(A) \) (with the desired constant probability). For \( p \geq 2 \), we additionally have [24, Theorem 1.7] the lower bound \( \mathcal{E}_p(A) \geq d \), which when plugged into the value of \( m \) gives the claimed sample complexity. A similar argument may be applied for the case \( p < 2 \); specifically, we have that

\[ \text{Var} \left( \frac{1}{m} \sum_{j[m]} r_j \right) \leq \frac{d}{m} \sum_{i=1}^{n} \frac{\sigma_{p,A}(a_i)^2}{w_p(a_i)} \leq \frac{d}{m} \sum_{i=1}^{n} \sigma_{p,A}(a_i) \leq \frac{2d}{m} \mathcal{E}_p(A), \]

where the second step is by \( w_p(a_i) \geq \sigma_{p}(a_i) \) from Fact 3.3. For \( p < 2 \), we also have \( \mathcal{E}_p(A) \geq d^{p/2} \) from [24, Theorem 1.7]. Applying Chebyshev’s inequality with this fact gives a sample complexity of \( m = O \left( d^{1-p/2}/\gamma^2 \right) \). This completes the correctness guarantee.

Runtime. We first compute all \( \ell_p \) Lewis weights of \( A \in \mathbb{R}^{n \times d} \) up to a constant multiplicative accuracy, the cost of which is \( O \left( \frac{1}{1-1/p^2} \log(\log(n)) \right) \) leverage score computations for \( p < 4 \) [23] and \( O(p^3 \log(np)) \) leverage score computations for \( p \geq 4 \) [77]. Next, we compute \( m = O(d^{1-p/2}) \) sensitivities with respect to \( S_pA \). From [23, 35], \( S_p \) has, with high probability, \( O(d^{p/2} \log d) \) rows when \( p > 2 \) and \( O(d \log d) \) rows when \( p \leq 2 \). Summing the cost of computing these \( m \) sensitivities and that of computing the Lewis weights gives the claimed runtime. \( \square \)

Remark 3.6. We present in Appendix B.2 an alternate algorithm, Algorithm 4, for estimating the total \( \ell_p \) sensitivity for \( p = 1 \). Algorithm 4 uses recursive computations of leverage scores, in contrast to Algorithm 2 which uses Lewis weights in a one-shot manner, and may be of independent interest.
3.2 Approximating the maximum $\ell_1$ sensitivity

In this section, we present an algorithm that approximates $\|\sigma_1(A)\|_\infty = \max_{i \in [\ell]} \sigma_1(a_i)$, the maximum of the $\ell_1$ sensitivities of the rows of a matrix $A$. As alluded to in Section 1, a first approach to this problem would be to simply estimate all $n$ sensitivities and compute their maximum. To do better than this, one idea, inspired by the random hashing approach of Algorithm 1, is as follows.

If the matrix has a large number of high-sensitivity rows, then, intuitively, the appropriately scaled maximum sensitivity of a uniformly sampled subset of these rows should approximate $\|\sigma_1(A)\|_\infty$. Specifically, assume that the matrix has at least $\alpha$ rows of sensitivity at least $\|\sigma_1(A)\|_\infty / \sqrt{\alpha}$; uniformly sample $n/\alpha$ rows and return $\tilde{\sigma}_1$, the (appropriately scaled) maximum of $\sigma_1^{RT}(a_i)$, for the sampled rows $i$. Then, $\|\sigma_1(A)\|_\infty \leq \tilde{\sigma}_1 \leq O(\sqrt{\alpha} \|\sigma_1(A)\|_\infty)$ with a constant probability. Here the upper bound guarantee is without any condition on the number of rows with large $\ell_1$ sensitivities.

In the other case, if the matrix does not have too many high-sensitivity rows, we could estimate the maximum sensitivity by hashing rows into small buckets via Rademacher combinations of uniformly sampled blocks of $\alpha$ rows each (just like in Algorithm 1). Then, $\tilde{\sigma}_2$, the scaled maximum of the $\ell_1$ sensitivities of these rows satisfies $\|\sigma_1(A)\|_\infty \leq \tilde{\sigma}_2 \leq O(\sqrt{\alpha} \|\sigma_1(A)\|_\infty)$. Here, it is the lower bound that comes for free (i.e., without any condition on the number of rows with large $\ell_1$ sensitivities).

Despite the above strategies working for each case, there is no way to combine these approaches without knowledge of whether the input matrix has a enough “high-sensitivity” rows or not. We therefore avoid this approach and instead present Algorithm 3, where we make use of $S_{\infty}A$, an $\ell_\infty$ subspace embedding of $A$. Our algorithm hinges on the recent development [75, Theorem 1.3] on efficient construction of such an embedding with simply a subset of $\tilde{O}(d)$ rows of $A$.

**Algorithm 3 Approximating the Maximum of $\ell_1$-Sensitivities.**

**Input:** Matrix $A \in \mathbb{R}^{n \times d}$

**Output:** Scalar $\bar{s} \in \mathbb{R}_{\geq 0}$ that satisfies $\|\sigma_1(A)\|_\infty \leq \bar{s} \leq C \cdot \sqrt{d} \cdot \|\sigma_1(A)\|_\infty$

1: Compute, for $A$, an $\ell_\infty$ subspace embedding $S_{\infty}A \in \mathbb{R}^{O(d \log^\epsilon(d)) \times d}$ such that $S_{\infty}A$ is a subset of the rows of $A$ [75, Theorem 1.3]

2: Compute, for $A$, an $\ell_1$ subspace embedding $S_1A \in \mathbb{R}^{O(d) \times d}$

3: Return $\sqrt{d} \|\sigma_1^{S_1A}(S_{\infty}A)\|_\infty$

**Theorem 3.7 (Approximating the Maximum of $\ell_1$ Sensitivities).** Given a matrix $A \in \mathbb{R}^{n \times d}$, there exists an algorithm, which in time $\tilde{O}(nnz(A) + d^{\omega+1})$, outputs a positive scalar $\bar{s}$ that satisfies

$$\Omega(\|\sigma_1(A)\|_\infty) \leq \bar{s} \leq O(\sqrt{d} \|\sigma_1(A)\|_\infty).$$

**Proof Sketch of Theorem 3.7; full proof in Appendix B.3.** We achieve our guarantee via Algorithm 3.

Define $x^*$ and $a_{x^*}$ as: $x^* = \arg \max_{x \in \mathbb{R}^d} \|x\|_1 \frac{x^T S_1Ax}{\|Ax\|_1}$. Since $S_1A$ is an $\ell_1$ subspace embedding of $A$, we have, for any $x \in \mathbb{R}^d$, that $\|S_1Ax\|_1 = \Theta(\|Ax\|_1)$. If $S_{\infty}A$ contains the row $a_{x^*}$, then $\|S_1Ax\|_1 = \Theta(\|Ax\|_1)$ implies $\|\sigma_1^{S_1A}(S_{\infty}A)\|_\infty = \Theta(1) \|\sigma_1(A)\|_\infty$. In the other case, suppose $a_{x^*}$ is not included in $S_{\infty}A$. Then we observe that

$$\|\sigma_1^{S_1A}(S_{\infty}A)\|_\infty \leq \max_{x \in \mathbb{R}^d} \frac{x^T S_1Ax}{\|S_1Ax\|_1} \geq \max_{x \in \mathbb{R}^d} \frac{\|S_\inftyAx\|_\infty}{\|S_1Ax\|_1} = \Theta(1) \max_{x \in \mathbb{R}^d} \frac{\|S_\inftyAx\|_\infty}{\|Ax\|_1},$$

where the second step is by choosing a specific vector in the numerator and the third step uses $\|S_1Ax\|_1 = \Theta(\|Ax\|_1)$. We further have,

$$\max_{x \in \mathbb{R}^d} \frac{\|S_\inftyAx\|_\infty}{\|Ax\|_1} \geq \frac{\|S_\inftyAx\|_\infty}{\|Ax\|_1} \geq \frac{\|Ax\|_\infty}{\sqrt{d} \|Ax\|_1} \geq \frac{\|a_{x^*}\|_1}{\sqrt{d} \|Ax^*\|_1} = \frac{1}{\sqrt{d}} \|\sigma_1(A)\|_\infty, \tag{3.3}$$

where the first step is by choosing $x = x^*$, the second step is by the guarantee of $\ell_\infty$ subspace embedding, and the final step is by definition of $\sigma_1(a_{x^*})$. Combining Inequality (3.2) and Inequality (3.3) gives the claimed lower bound on $\|\sigma_1^{S_1A}(S_{\infty}A)\|_\infty$. The runtime follows from the cost of computing $S_{\infty}A$ from [75] and that of computing $\tilde{O}(d)$ of $\ell_1$ sensitivities with respect to $S_1A$. \(\square\)
4 Numerical experiments

We demonstrate our fast sensitivity approximations on multiple real-world datasets in the UCI Machine Learning Dataset Repository [46], such as the wine and fires datasets, for different $p$ and varying approximation parameters $\alpha$. We focus on the wine dataset here, with full details in Appendix E.

**Experimental Setup.** For each dataset and each $p$, we first apply the Lewis weight subspace embedding to compute a smaller matrix $SA$. Then, we run our sensitivity computation Algorithm 1 and measure the average and maximum absolute log ratio $|\log(\sigma_{\text{approx}}/\sigma_{\text{true}})|$ to capture the relative error of the sensitivity approximation. Note that this is stricter than our guarantees, which give only an $O(\alpha)$ additive error; therefore we have no real upper bound on the maximum absolute log ratio due to small sensitivities. We plot an upper bound of $\log(\alpha^p)$, which is the worst case additive error approximation although it does provide relative error guarantees with respect to the average sensitivity. We compare our fast algorithm for computing the total sensitivity with the naive brute-force method by approximating all sensitivities and then averaging.

**Analysis.** In practice, we found most real-world datasets have easy-to-approximate sensitivities with total $\ell_p$ sensitivity about 2-5 times lower than the theoretical upper bound. Figure 1 shows that the average absolute log ratios for approximating the $\ell_p$ sensitivities are much smaller than those suggested by the theoretical upper bound, even for large $\alpha$. Specifically, when $\alpha = 40$, we incur only a 16x accuracy deterioration in exchange for a 40x faster algorithm. This is significantly better than the worst-case accuracy guarantee which for $p = 3$ would be $\alpha^p = 40^3$.

More importantly, we find that empirical total sensitivities are much lower than the theoretical upper bounds suggest, often by a factor of at least 5, especially for $p > 2$. This implies that real-world structure can be utilized for improved data compression and justifies the importance of sensitivity estimation for real-world datasets. Furthermore, our novel algorithm approximates the total sensitivity up to an overestimate within a factor of 1.3, in less than a quarter of the time of the brute-force algorithm. Our observation generalizes to other real-world datasets (see Appendix E).

<table>
<thead>
<tr>
<th>$p$</th>
<th>Total Sensitivity Upper Bound</th>
<th>Brute-Force/True</th>
<th>Approximation</th>
<th>Brute-Force Runtime (s)</th>
<th>Approximate Runtime (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>14</td>
<td>5.2</td>
<td>6.4</td>
<td>667</td>
<td>105</td>
</tr>
<tr>
<td>1.5</td>
<td>14</td>
<td>11.6</td>
<td>14.2</td>
<td>673</td>
<td>131</td>
</tr>
<tr>
<td>2.5</td>
<td>27.1</td>
<td>13.8</td>
<td>14.9</td>
<td>693</td>
<td>209</td>
</tr>
<tr>
<td>3</td>
<td>52.4</td>
<td>7.2</td>
<td>8.9</td>
<td>686</td>
<td>192</td>
</tr>
</tbody>
</table>

Table 1: Runtime comparison for computing total sensitivities for the wine dataset, which has matrix shape (177, 14).

**Acknowledgements**

We are very grateful to: Sagi Perel, Arun Jambulapati, and Kevin Tian for helpful discussions about related work; Taisuke Yasuda for his generous help discussing results in $\ell_p$ sensitivity sampling; and our NeurIPS reviewers for their time, effort, and many constructive suggestions. Swati gratefully acknowledges funding for her student researcher position from Google Research and research
assistantship at UW from the NSF award CCF-1749609 (via Yin Tat Lee). Part of this work was
done while D. Woodruff was at Google Research. D. Woodruff also acknowledges support from
a Simons Investigator Award.
References


A Omitted proofs: general technical results

Fact A.1 (Crude sensitivity approximation via leverage scores). Given a matrix $A \in \mathbb{R}^{n \times d}$, let $\sigma_1(a_i)$ denote the $i$th $\ell_1$ sensitivity of $A$ with respect to $A$, and let $\tau_i(A)$ denote its $i$th leverage score with respect to $A$. Then we have $\sqrt{\frac{\tau_i(A)}{n}} \leq \sigma_1(a_i) \leq \sqrt{\tau_i(A)}$.

Proof. The proof of this claim follows from a simple application of standard norm inequalities, and we present one here for completeness. For any $u \in \mathbb{R}^n$, we have $\|u\|_2 \leq \|u\|_1 \leq \sqrt{n}\|u\|_2$. Therefore, for any $x \in \mathbb{R}^d$, we have

$$\frac{|x^\top a_i|}{\sqrt{n}\|Ax\|_2} \leq \frac{|x^\top a_i|}{\|Ax\|_1} \leq \frac{|x^\top a_i|}{\|Ax\|_2}. \tag{A.1}$$

Let $x_\sigma$ be the vector that realizes the $i$th leverage score, and $x_\tau$ be the vector that realizes the $i$th sensitivity (i.e., they are the maximizers of sensitivity). Then, we may conclude from Inequality (A.1) that

$$\sqrt{\frac{|x_\tau^\top a_i|^2}{n\|Ax_\tau\|_2^2}} \leq \frac{|x_\tau^\top a_i|}{\|Ax_\tau\|_1} \leq \frac{|x_\tau^\top a_i|}{\|Ax_\tau\|_2} \leq \sqrt{\frac{|x_\sigma^\top a_i|^2}{\|Ax_\sigma\|_2^2}},$$

which gives the claimed result. □

Fact A.2. Given an $n \times d$ matrix, the cost of computing $k$ of its $\ell_1$ sensitivities is $O(\text{nnz}(A) + k \cdot d^2)$.

Proof. Given an $n \times d$ matrix $A$, we first compute $SA$, its $\ell_1$ subspace embedding of size $O(d) \times d$, and compute the $i$th sensitivity of $A$ as follows.

$$\sigma_1^A(a_i) = \max_{x \in \mathbb{R}^d} \frac{|a_i^\top x|}{\|Ax\|_1} = \max_{x \in \mathbb{R}^d} \frac{|a_i^\top x|}{\Theta(\|SAx\|_1)} \approx O(1) \sigma_1^{SA}(a_i),$$

where the second step is by the definition of $\ell_1$ subspace embedding (cf. Definition 2.3). To compute $\sigma_1^{SA}(a_i)$, we need to compute $\min_{a_i^\top x = 1} \|SAx\|_1$, which, by introducing a new variable for each of the rows of $SAx$, can be transformed into a linear program with $O(d)$ variables and $d$ constraints. To see the claim on runtime, the cost of computing the subspace embedding is $\text{nnz}(A)$. Having computed this once, we can then solve $k$ linear programs, each at cost $d^2$ (which is the current fastest LP solver [83]). Putting together these two costs gives the claimed runtime. □

Lemma A.3. Given positive numbers $a_1, a_2, \ldots, a_m$, let $r := \frac{\max_{i \in [m]} a_i}{\min_{i \in [m]} a_i}$ and $A^{(\text{true})} := \sum_{i \in [m]} a_i$. Suppose we sample, uniformly at random with replacement, a set $S \in [m]$ of these numbers, and let $A^{(\text{est})} := \frac{m}{|S|} \sum_{i \in S} a_i$. Then, if $|S| \geq 10 \left(\frac{(1 + r)^2}{\gamma^2} \log \left(\frac{1}{\delta}\right)\right)$ for a large enough absolute constant $C$, we can ensure with at least a probability of $1 - \delta$, that $|A^{(\text{est})} - A^{(\text{true})}| \leq \gamma A^{(\text{true})}$.

Proof. By construction, the expected value of a sample $a_i$ in $S$ is $\frac{1}{m} A^{(\text{true})}$. Denoting by $\mathcal{P}$ the uniform distribution over $a_1, a_2, \ldots, a_m$, and let $\mathcal{P}$ be the standard deviation of this distribution $\mathcal{P}$. Then, we can apply Bernstein’s inequality [84, Theorem 2.8.1] to see that the absolute error $|A^{(\text{est})} - A^{(\text{true})}|$ satisfies the following guarantee for all $t > 0$:

$$\Pr \left\{ |A^{(\text{est})} - A^{(\text{true})}| \geq t \right\} = \Pr \left\{ \left| \sum_{i \in S} \left( a_i - \frac{1}{m} A^{(\text{true})} \right) \right| \geq t \cdot \frac{|S|}{m} \right\}$$

$$\leq e \left\{ \frac{1}{2} t^2 \cdot \frac{|S|^2}{m^2} \right\} = \frac{1}{2} t^2 \cdot \frac{|S|}{m^2} \max_{i \in [m]} a_i,$$

$$= e \left\{ \frac{1}{2} t^2 \cdot |S| \right\} - \frac{1}{2} t^2 \cdot \frac{1}{m} \cdot \max_{i \in [m]} a_i \right\}. \tag{A.2}$$
Since each of the $a_i$s is positive, we note that

$$\sigma(P)^2 = \frac{1}{m} \sum_{i \in [m]} \left( a_i - \frac{A^{\text{true}}}{m} \right)^2 \leq \frac{1}{m} \sum_{i \in [m]} a_i^2 \leq \frac{1}{m} \max_{i \in [m]} a_i \cdot \sum_{i \in [m]} a_i = \frac{1}{m} \max_{i \in [m]} a_i \cdot \sigma(A^{\text{true}}).$$

(A.3)

Combining Inequality (A.2) and Inequality (A.3) for $t = \gamma A^{\text{true}}$ implies that

$$\Pr \left\{ |A^{\text{est}} - A^{\text{true}}| \geq \gamma A^{\text{true}} \right\} \leq e^{-\left( \frac{1}{2} \gamma A^{\text{true}} \cdot \max_{i \in [m]} a_i \right)} \leq \delta$$

(4.4)

holds for the choice of $|S| \geq 10 \left( \frac{m\gamma}{2}(1+\gamma)\frac{\max_{i \in [m]} a_i}{A^{\text{true}}} \log \left( \frac{1}{\delta} \right) \right)$ yields the claimed error guarantee.

\[ \square \]

B Omitted proofs: $\ell_1$ sensitivities

B.1 Estimating all $\ell_1$ sensitivities

Theorem 3.1. Given a full-rank matrix $A \in \mathbb{R}^{n \times d}$ and an approximation factor $1 < \alpha < n$, let $\sigma_1(a_i)$ be the $\ell_1$ sensitivity of the $i$th row of $A$. Then there exists an algorithm that, in time $\tilde{O}(\text{nzn}(A) + \frac{n}{\alpha} \cdot d^\omega)$, returns $\tilde{\sigma} \in \mathbb{R}^n_{\geq 0}$ such that with high probability, for each $i \in [n]$,

$$\sigma_1(a_i) \leq \tilde{\sigma_i} \leq O(\sigma_1(a_i) + \frac{n}{\alpha} \tilde{\sigma}_1(A)).$$

(3.1)

Proof. Note that in Line 2 of Algorithm 1, the rows of $A$ are partitioned into randomly created $n/\alpha$ blocks. Suppose $a_i$, the $i$th row of $A$, falls into the block $B_\ell$. Recall, that in Line 5, the rows from $B_\ell$ are mapped to those in $P$ with row indices in the set $J$. Then, we compute the $i$th $\ell_1$ sensitivity estimate as

$$\tilde{\sigma_i} = \max_{j \in J} \sigma_{SA}^1(p_j).$$

(3.1.3)

We use Equation (B.1) below to establish the claimed bounds. Let $x^* = \arg \max_{x \in \mathbb{R}^d} \frac{|a_i^\top x|}{\|SAx\|_1}$ be the vector that realizes the $i$th $\ell_1$ sensitivity of $A$. Further, let the $j$th row of $P$ be $r^{(f)}_{k,j} B_\ell$. Then, with a probability of at least $1/2$, we have

$$\sigma_{SA}^1(p_j) = \max_{x \in \mathbb{R}^d} \frac{|r^{(f)}_{k,j} B_\ell x|}{\|SAx\|_1} \geq \Theta(1) \frac{|r^{(f)}_{k,j} B_\ell x^*|}{\|SAx^*\|_1} \geq \Theta(1) \frac{|a_i^\top x^*|}{\|Ax^*\|_1} = \Theta(1) \sigma_1(a_i),$$

(3.2)

where the first step is by definition of $\sigma_{SA}^1(p_j)$, the second is by evaluating the function being maximized at a specific choice of $x$, the third step is by applying $\|SAx\|_1 = \Theta(\|Ax\|_1)$ from Equation (B.3), and the final step is by definition of $x^*$ and $\sigma_1(a_i)$. For the fourth step, we use the fact that $|r^{(f)}_{k,j} B_\ell x^*| = |r^{(f)}_{k,j} a_i^\top x^* + \sum_{j \neq i} r^{(f)}_{k,j}(B_\ell x^*)| \geq |a_i^\top x^*|$ with a probability of at least $1/2$ since the vector $r^{(f)}_{k,j}$ has coordinates that are $+1$ or $-1$ with equal probability. By a union bound over the $|J|$ independent rows that block $B_\ell$ is mapped to, we establish the claimed lower bound in Inequality (B.2) with a probability of at least $0.9$. To show an upper bound on $\sigma_{SA}^1(p_j)$, we observe that

$$\sigma_{SA}^1(p_j) = \max_{x \in \mathbb{R}^d} \frac{|r^{(f)}_{k,j} B_\ell x|}{\|SAx\|_1} \leq \max_{x \in \mathbb{R}^d} \frac{\|B_\ell x\|_1}{\|SAx\|_1} \leq \Theta(1) \max_{x \in \mathbb{R}^d} \frac{\|B_\ell x\|_1}{\|Ax\|_1} \leq \Theta(1) \sum_{j:a_j \in B_\ell} \max_{x \in \mathbb{R}^d} \frac{|a_j^\top x|}{\|Ax\|_1},$$

(3.3)

where the second step is by Hölder inequality and the fact that the entries of $r^{(f)}_{k,j}$ are all bounded between $-1$ and $+1$, and the third step is by Equation (B.1). The final term of the above chain of inequalities may be rewritten as $\Theta(1) \sum_{j:a_j \in \tilde{B}_\ell} \tilde{\sigma}_j(A)$, by the definition of $B_\ell$. Because $B_\ell$ is a group of $\alpha$ rows selected uniformly at random out of $n$ rows and contains the row $a_i$, we have:

$$\mathbb{E} \left\{ \sum_{j:a_j \in B_\ell, j \neq i} \sigma_1(a_j) \right\} = \frac{\alpha - 1}{n - 1} \sum_{j \neq i} \sigma_1(a_j).$$
Therefore, Markov inequality gives us that with a probability of at least 0.9, we have
\[ \sigma_1^{SA}(p_j) \leq \sigma_1(a_i) + \mathcal{G}_1(A)O(\frac{n}{\gamma}). \] (B.4)

The median trick then establishes the bounds in Inequality (B.2) and Inequality (B.4) with high probability. The claimed runtime is obtained by the cost of constructing \( SA \) plus \( O(n/\alpha) \) computations of \( \ell_1 \) sensitivities with respect to \( SA \), for which we may use Fact 2.4.

### B.2 Estimating the sum of \( \ell_1 \) sensitivities

In this section, we present a randomized algorithm that approximates the total sensitivity \( \mathcal{G}_1(A) \), up to a \( \gamma \)-multiplicative approximation for some \( \gamma \in (0, 1) \) that is not too small. This algorithm is significantly less general than Algorithm 2 (which holds for all \( p \geq 1 \)) but it uses extremely simple facts about leverage scores and what we think is a new recursion technique in this context, which we believe could be of potential independent interest.

**Theorem B.1.** Given a matrix \( A \in \mathbb{R}^{n \times d} \) and an approximation factor \( \gamma \in (0, 1) \) such that \( \gamma \geq \Omega\left(\frac{1}{n^2}\right) \), there exists an algorithm, which in time \( \tilde{O}\left(\text{nnz}(A) + \frac{\sigma^2}{\tau} \cdot \max\left(\sqrt{d}, \frac{1}{\tau}\right)\right) \), returns a positive scalar \( \hat{s} \) such that, with a probability of 0.99, we have
\[ \mathcal{G}_1(A) \leq \hat{s} \leq (1 + O(\gamma))\mathcal{G}_1(A). \]

**Algorithm 4 Approximating the Sum of \( \ell_1 \)-Sensitivities**

**Input:** Matrix \( A \in \mathbb{R}^{n \times d} \) and approximation factor \( \gamma \in (0, 1) \)

**Output:** A positive scalar that satisfies, with probability 0.99, that
\[ \mathcal{G}_1(A) \leq \hat{s} \leq (1 + O(\gamma))\mathcal{G}_1(A) \]

1. Compute \( SA \in \mathbb{R}^{O(d) \times d} \), a 1/2-approximate \( \ell_1 \) subspace embedding of \( A \) (cf. Definition 2.3)
2. Set \( \rho = O(\gamma/\log(\log(n + d))) \). Compute \( S'A \), a \( \rho \)-approximate \( \ell_1 \) subspace embedding of \( A \)
3. Construct a matrix \( A \) comprising only those rows of \( A \) with leverage scores at least \( 1/n^{10} \)
4. Let \( s \) be the output of Algorithm 5 with inputs \( \hat{A}, SA, S'A, n, \) and \( \rho \)
5. Return \( \hat{s} := (1 + \gamma)\left(s + \frac{1}{n^5}(|A| - |\hat{A}|)\right) \)

#### B.2.1 Proof sketch.

We achieve our guarantee via Algorithm 4 and Algorithm 5, which are based on the following key principles. The first is Fact 2.2: the \( \ell_1 \) sensitivity and \( \ell_1 \) leverage score of \( A \in \mathbb{R}^{n \times d} \) satisfy
\[ \sqrt{\tau_i(A)/n} \leq \sigma_1(a_i) \leq \sqrt{\tau_i(A)}. \] The second idea, derived from Bernstein’s inequality (and formalized in Lemma A.3), is: the sum of a set of positive numbers with values bounded between \( a \) and \( b \) can be approximated to a \( \gamma \)-multiplicative factor using \( \tilde{O}(\sqrt{(b/a)\gamma^{-2}}) \) uniformly sampled (and appropriately scaled) samples. The third principle is that \( \sigma_1(a_i) \approx \sigma_1^{SA}(a_i) \) when \( SA \) is a constant-approximation \( \ell_1 \) subspace embedding of \( A \).

Based on these ideas, we first divide the rows of \( A \) into \( B = \Theta(\log n) \) submatrices based on which of the \( B \) buckets \([1/2, 1], [1/4, 1/2], [1/8, 1/4], \ldots\) their leverage scores (computed with respect to \( A \)) fall into. Since each of these \( B \) submatrices comprises rows with leverage scores in the range \([a, 2a]\) for some \( a \), per Fact 2.2, we have \( \sqrt{\frac{a}{n}} \leq \sigma_1(a_i) \leq \sqrt{2a} \). Therefore, by Lemma A.3, the total sensitivity of this submatrix may be approximated by the appropriately scaled total sensitivity of \( O(\sqrt{n}) \) of its uniformly sampled rows. Our design choice to split rows based on leverage scores is based on the fact that leverage scores (unlike sensitivities) are efficiently computable (using [60]). Computing the total sensitivity of an \( \tilde{O}(\sqrt{n}) \)-sized matrix thus costs \( \tilde{O}(\text{nnz}(A) + \sqrt{n} \cdot d^\omega) \) under this scheme.

Applying the idea sketched above recursively further reduces the cost. Suppose, after dividing the input matrix \( A \) into \( B \) submatrices \( M_i \) and subsampling \( \hat{M}_i \) from these matrices, we try to recurse on each of these \( B \) submatrices \( \hat{M}_i \). A key requirement for this idea to work is that the sum of the \( \ell_1 \) sensitivities of the rows of \( \hat{M}_i \), in isolation, be close enough to the true sum of \( \ell_1 \) sensitivities of

---

19
Algorithm 5 Subroutine for Approximating the Sum of $\ell_1$-Sensitivities

Input: Matrices $\mathbf{M} \in \mathbb{R}^{r \times d}$, $\mathbf{S}\mathbf{A} \in \mathbb{R}^{O(d) \times d}$, $\mathbf{S}'\mathbf{A} \in \mathbb{R}^{O(d) \times d}$, scalars $n$ and $\rho \in (0, 1)$

Output: A positive scalar $\hat{s}$ that satisfies, with probability $0.99$, that

$$\mathbb{E}_{\mathbf{S}'}\mathbb{E}_{\mathbf{A}}\mathbf{M} \leq \hat{s} \leq (1 + O(\rho))\mathbb{E}_{\mathbf{S}'}\mathbb{E}_{\mathbf{A}}\mathbf{M}.$$

1. Set $B = \Theta(\log n)$, $D = O(\log(\log(n + d)))$, $\delta = 0.01$, and $b = O\left(\frac{D^4}{\ell_n} \max\left(\frac{D^4}{\ell_n}, \sqrt{d}\right)\right)$.
2. if the number of rows of $\mathbf{M}$ is at most $b$ then
3. Let $\mathbb{E}_{\mathbf{S}'}\mathbb{E}_{\mathbf{A}}\mathbf{M}$ be the sum of $\ell_1$ sensitivities of the rows of $\mathbf{M}$ with respect to $\mathbf{S}'\mathbf{A}$.
4. Using Fact 2.4 on $\mathbf{M}$, compute $\hat{s}$ such that $\mathbb{E}_{\mathbf{S}'}\mathbb{E}_{\mathbf{A}}\mathbf{M} \leq \hat{s} \leq (1 + O(\rho))\mathbb{E}_{\mathbf{S}'}\mathbb{E}_{\mathbf{A}}\mathbf{M}$.
5. Return $\hat{s}$.
6. else
7. Construct the matrix $\mathbf{C} := \left[\mathbf{M} \mathbf{S}\mathbf{A}\right]$. Set $r = O\left(\sqrt{|\mathbf{C}|}(1 + \rho)^{-2} \log(\delta^{-1})\right)$.
8. Compute $\tau(C)$, the vector of leverage scores of $\mathbf{C}$
9. Divide $\left[\frac{r}{r-1}, 1\right]$ into $B$ geometrically decreasing sub-intervals $[1/2, 1]$, $[1/4, 1/2]$, $\ldots$, etc.
10. for each bucket $i \in [B]$ created in the previous step do
11. Construct matrix $\mathbf{M}_i$ with those rows of $\mathbf{M}$ whose leverage scores fall in the $i$th bucket.
12. Construct matrix $\mathbf{M}_i^*$ composed of $r$ uniformly sampled (with replacement) rows of $\mathbf{M}_i$.
13. Compute $\hat{s}_i$, the output of Algorithm 5 with inputs $\mathbf{M}_i^*$, $\mathbf{S}\mathbf{A}$, $\mathbf{S}'\mathbf{A}$, $n$, and $\rho$.
14. end for
15. Return $\hat{s} := (1 + \rho) \sum_{i \in [B]} \left|\frac{\mathbf{M}_i}{\mathbf{M}_i^*}\right| \hat{s}_i$.
16. end if

those rows with respect to the original matrix $\mathbf{A}$. In general, this is not true. However, we can meet our requirement using the $\ell_1$ subspace embedding $\mathbf{S}\mathbf{A}$ of the (original) matrix $\mathbf{A}$. Specifically, if we vertically concatenate a matrix $\mathbf{M}_i$ with $\mathbf{S}\mathbf{A}$ and call this matrix $\mathbf{C}$, then for each row $\mathbf{M}_i[j] \in \mathbf{M}_i$, we have $\sigma^C(l)_{\mathbf{M}_i[j]} \approx \sigma^A(l)_{\mathbf{M}_i[j]}$. Further, $|\mathbf{C}| \leq O(\sqrt{n})$, and so in the step subsampling the rows of $\mathbf{M}_i$, we choose $O(n^{1/4})$ samples. Recursing this exponentially decreases the row dimension of the input matrix at each level, with $(\log n)^k$ matrices at the $k$th level of recursion.

B.2.2 Helper lemmas for proving Theorem B.1

For our formal proof, we need Definition B.2, Fact B.3, and Definition B.4.

Definition B.2 (Notation for recursion tree of Algorithm 5). Recall that $B = \Theta(\log(n))$ as in Line 1 of Algorithm 5. Then we have the following notation corresponding to Algorithm 5:

- We define a rooted $B$-ary tree $T$ corresponding to the execution of Algorithm 5.
- We use $T_j$ to denote the subtree starting from the root node with all the nodes up to and including those at the $j$th level. We denote the $i$th node at the $j$th level by $T_{(j,i)}$. Thus, every node is uniquely specified by its level in the tree and its index within that level.
- The root node $T_{(1,1)}$ at level 1 corresponds to the first call to Algorithm 5 from Algorithm 4.
- For the node $T_{(j,i)}$, we denote by $\mathbf{M}_{(j,i)}$ the input matrix to the corresponding recursive call; note that all the other inputs are global and remain the same throughout the recursive call.
- Analogously, we use $\hat{s}_{(j,i)}$ to denote our estimate of $\mathbb{E}_{\mathbf{S}'}\mathbb{E}_{\mathbf{A}}\mathbf{M}_{(j,i)}$, the sum of $\ell_1$ sensitivities of rows of the matrix $\mathbf{M}_{(j,i)}$ in the node $T_{(j,i)}$, defined with respect to the sketched matrix $\mathbf{S}'\mathbf{A}$.

Fact B.3. If event $A$ happens with probability $1 - \tau_1$ and event $B$ happens with probability $1 - \tau_2$ conditioned on event $A$, then the probability of both events $A$ and $B$ happening is at least $1 - \tau_1 - \tau_2$.

Definition B.4. We say that a node $N$ with matrix $\mathbf{M}$ satisfies the $(\hat{\rho}, \hat{\delta})$-approximation guarantee if $\hat{s}$, the output of Algorithm 5 on $\mathbf{M}$, satisfies the following guarantee for the true sum $\mathbb{E}_{\mathbf{S}'}\mathbb{E}_{\mathbf{A}}\mathbf{M}$ of $\mathbf{M}$:

$$\hat{s} \approx_{\hat{\rho}} \mathbb{E}_{\mathbf{S}'}\mathbb{E}_{\mathbf{A}}\mathbf{M}$$

with a probability at least $1 - \hat{\delta}$.
for some approximation parameter \( \hat{\rho} \in (0, 1) \) and an error probability parameter \( \hat{\delta} \in (0, 1) \).

**Lemma B.5** (Partitioning of M). In Algorithm 5, even though we ignore the range \([0, \frac{1}{m}]\) when creating the submatrices \( M_1, M_2, \ldots, M_b \) from \( M \), every row in \( M \) is part of exactly one of the submatrices \( M_1, M_2, \ldots, M_b \). In other words, the matrix \( M \) is partitioned into the matrices \( M_1, M_2, \ldots, M_b \), with no row missed out.

**Proof.** Our proof strategy is to show that for all row indices \( k \in \|M\| \), each of the leverage scores \( \tau_k(C) \) satisfy \( \tau_k(C) \geq \frac{1}{L} \). Therefore, creating buckets of rows with the range of leverage score starting at \( \frac{1}{L} \) (instead of 0) does not miss out any rows.

To this end, we note that for every row \( a_k \) in the input \( M \) to Algorithm 5, we have:

\[
\frac{1}{n} \leq \sqrt{\frac{x^T a_k}{|A|}} \leq \sigma_1^A(a_k) = \max_{x \in \mathbb{R}^d} \left| \frac{x^T a_k}{\|Ax\|_1} \right| \leq \max_{x \in \mathbb{R}^d} \frac{|x^T a_k|}{\|Cx\|_1} = 3 \sigma_1^C(a_k) \leq 3 \sqrt{\tau_k(C)},
\]

where the first step is because in Line 3 of Algorithm 4, we discard those rows \( a_k \) of \( A \) with leverage scores smaller than \( 1/n^{10} \) and there are \( n \) rows in \( A \), so every row \( a_k \) in the input matrix \( M \) to Algorithm 5 satisfies \( \tau_k(A) \geq \frac{1}{n^{10}} \); the second step is by Fact 2.2 applied to \( a_k \); the third step is by the definition of the \( \ell_1 \) sensitivity of \( a_k \) with respect to \( A \); the fourth step is by observing that \( \|Cx\|_1 = \|SAx\|_1 + \|Max\|_1 \leq 2\|Ax\|_1 + \|Ax\|_1 = 3\|Ax\|_1 \) since \( SA \) is a constant-factor (with the constant assumed 1/2) \( \ell_1 \) subspace embedding of \( A \) and because \( M \) is a submatrix of \( A \); the fifth step is by the definition of \( \ell_1 \) sensitivity of \( a_k \) with respect to \( C \); the final step is by Fact 2.2 applied to \( a_k \) within \( C \). Observing the first and last terms of this inequality chain, we have \( \tau_k(C) \geq \frac{1}{3\sqrt{n}} \) for all \( k \in \|C\| \). This finishes the proof of the claim. \( \square \)

**Lemma B.6.** For the recursion tree \( T \) corresponding to Algorithm 5, let the size \( b \) of the input matrices at the leaf nodes be \( b = \frac{5BD^3 \log(100B)}{\gamma} \cdot \left( \frac{2D^3 \log(100B)}{\gamma} + \sqrt{d \log(d)} \right) \). Then, the depth of the recursion tree is less than \( D := 1 + \log(\log(2n + 2d \log(d))) \).

**Proof.** For now, let \( \beta \) be arbitrary but less than \( \beta^2 \leq n + d' \), where \( d' \) is the row dimension of the subspace embedding \( SA \). Then, the size of the input matrix to the nodes of the recursion tree as we increase in depth evolves as

\[
\beta \sqrt{d' + n}, \beta \sqrt{d' + \beta \sqrt{d' + n}}, \beta \sqrt{d' + \beta \sqrt{d' + \beta \sqrt{d' + n}}} \ldots
\]

Thus, this sequence evolves as

\[
x_{t+1} = \beta \sqrt{d' + x_t}, \text{ with } x_0 = n + d'.
\]

To see the limit of this sequence, we set \( x = \beta \sqrt{d' + x} \), and solve for this quadratic equation as

\[
x = \frac{\beta^2 + \sqrt{\beta^4 + 4\beta^2 d'}}{2} \leq \beta(\beta + \sqrt{d'}).
\]

We set the base case to be \( b^* := 7\beta(\beta + \sqrt{d'}) \), a constant times larger than this asymptotic limit. We now compute the depth \( t^* \) at which the recursion attains the value \( b^* \) is attained. To do so, we consider a sequence \( \{y_t\} \) which overestimates the sequence \( \{x_t\} \) of matrix sizes for a big range of \( t \). We define \( \{y_t\} \) as

\[
y_0 = n + d', \text{ and } y_{t+1} = \beta \sqrt{2y_t},
\]

and it can be checked that for \( t \) such that \( y_t \geq d' \), we have \( y_{t+1} \geq x_{t+1} \). Moreover, for all \( t \geq 1, y_t = \beta^{1+1/2+\cdots+1/2^{t-1}}(2y_0)^{1/2^t} = \beta^{2-2^{-1}}(2y_0)^{1/2^t} \). The limit of this sequence can easily be seen to \( \beta^2 \). We now split our analysis in two cases based on the limits of these two sequences.

**Case 1:** \( \beta^2 \geq d' \). Since the limit of the sequence \( \{y_t\} \) is \( \beta^2 \), it is easy to see that \( y_t \) is always greater \( d' \) (since \( y_t \) is monotonically decreasing as \( y_0 \geq \beta^2 \)). Thus, \( x_t \) will always be less than \( y_t \) in this case. We will now show that \( y_t \) (and thus \( x_t \)) will be less than \( b^* \) in \( t_0 = 2 \log \log(2y_0) \) steps. Solving the inequality for \( t \) as follows: \( y_t \leq \beta^2 (2y_0)^{1/2^t} \leq 4 \beta^2 \leq b^* \), it suffices to ensure that \( (2y_0)^{1/2^t} \leq 4 \), which happens if \( t \geq t_0 \). Thus, \( t^* \leq 2 \log \log 2y_0 \).
Case 2: $\beta^2 < d'$. In this case, the sequence $y_t$ will eventually go below $d'$ and thus the inequality $y_t \geq x_t$ might eventually break down. Let $t = t'$ denote the first time step at which $y_t \leq 5d'$ (which will be finite in this case). Then, by definition, $y_t \geq x_t$ for all $t < t'$. We will now upper bound the value of $t'$ as follows: $y_t \leq \beta^2 (y_0)^{1/2} \leq d'(y_0)^{1/2} \leq 4d'$. Thus, it suffices to ensure that $(2y_0)^{1/2} \leq 4$, which happens if $t \geq t_0 := 2\log \log (2y_0)$. Hence, $t' \leq t_0$. Moreover, the value of $y_t \leq 5d'$, implies that $y_{t-1} \leq y_{t-1} \leq \frac{(5d')^2}{2\beta^2}$. We will now show that $x_{t+1} \leq b^*$ using the evolution of $x_{t+1} = \sqrt{d' + x_t}$ as follows: First, $x_t \leq \beta \sqrt{d'} + \beta \sqrt{x_{t-1}} \leq \beta \sqrt{d'} + 5d' \leq 6d$; Then, $x_{t+1} \leq \beta \sqrt{d'} + \beta \sqrt{6d} \leq 7\beta \sqrt{d} \leq b^*$. Thus, $t^* \leq 1 + 2\log \log 2y_0).

Therefore, in both of these cases, we have that after at most $D = 1 + \log \log (2n + 2d')$ depth, the base case of $b^* = 7\beta (\beta + \sqrt{d'})$ is achieved at the leaf node. Recall that the algorithm sets $D = 2\log \log (n + d') + 1$ and $\rho = \gamma / D$ for $d' = d \log (d)$ which implies that $\beta := C(1+\rho) \log (\delta^{-1}) = CD(D+\gamma) \log (100B^D) \leq 2D^3 \log (100B)/\gamma^2$. Since Algorithm 5 sets $b = 50D^3 \log (100B)/\gamma^2 \left( 2D^3 \log (100B) + \sqrt{d \log (d)} \right)$, which is larger than $b^*$, we may conclude that the choice of $b$ implies that the recursion tree $T$ of Algorithm 5 has a depth less than $D = 1 + \log (\log (2n + 2d \log (d)))$. 

\[ \square \]

Lemma B.7. Suppose the root node $T(1,1)$ of Algorithm 5 satisfies a $(\hat{\rho}, \hat{\delta})$-approximation guarantee as defined in Definition B.4, and denote by $\tilde{s}$ the output of Algorithm 4. Then, with a probability of at least $1 - \hat{\delta}$, we have
\[ \frac{1}{1 + \gamma} \tilde{s} \approx \hat{\rho} + \hat{\delta} \mathbb{E}^\ell_1(A). \]

As a result of this lemma, in the final proof of correctness of Algorithm 4 (appearing later as proof of Theorem B.1), it suffices to show that $\tilde{s}$ satisfies a $(\hat{\rho}, \hat{\delta})$-approximation guarantee for some appropriate choices of these parameters.

Proof of Lemma B.7. Suppose the root node satisfies Definition B.4 with some $(\hat{\rho}, \hat{\delta})$-approximation guarantee. This means that there exists an $s$ such that the output of Algorithm 5 on the input matrix $\hat{A}$ satisfies
\[ \tilde{s} \approx \hat{\rho} \mathbb{E}^\ell_1(A) \] with a probability at least $1 - \hat{\delta}$. (B.7)

Since $S'A$ is, by design, a $\rho$-approximation $\ell_1$ subspace embedding for $A$, it means for all $x \in \mathbb{R}^d$ we have
\[ (1 - \rho)\|Ax\|_1 \leq \|S'A x\|_1 \leq (1 + \rho)\|Ax\|_1. \] (B.8)

Therefore, for every row $a_i \in A$, we have by the definition of $\ell_1$ sensitivity and Inequality (B.8) that the $\ell_1$ sensitivity of $a_i$ computed with respect to $S'A$ is a $\rho$-approximation of the $\ell_1$ sensitivity of $a_i$ with respect to $A$:
\[ \sigma^\ell_1(S'A(a_i)) = \max_{x \in \mathbb{R}^d} \frac{|a_i^\top x|}{\|S'A x\|_1} \approx \rho \max_{x \in \mathbb{R}^d} \frac{|a_i^\top x|}{\|Ax\|_1} = \sigma^\ell_1(A(a_i)). \]

Therefore, by linearity, this $\rho$-approximation factor transfers over in connecting the total sensitivity of $\hat{A}$ with respect to $S'A$ to the total sensitivity of $\hat{A}$ with respect to $A$:
\[ \mathbb{E}^\ell_1(S'A(\hat{A})) = \sum_{i \in [||\hat{A}||]} \sigma^\ell_1(S'A(a_i)) \approx \rho \sum_{i \in [||\hat{A}||]} \sigma^\ell_1(A(a_i)) = \mathbb{E}^\ell_1(A). \] (B.9)

Therefore, chaining Equation (B.7) and Equation (B.9) yields the following approximation guarantee:
\[ \tilde{s} \approx \hat{\rho} + \hat{\delta} \mathbb{E}^\ell_1(A) \] with a probability at least $1 - \hat{\delta}$. (B.10)

There is no change in the error probability above from Equation (B.7) because Equation (B.9) holds deterministically. Next, since in Line 3 of Algorithm 4, we dropped rows of $A$ that have leverage scores less than $\frac{1}{n^{\delta}}$, for each of the dropped rows $a_i$, we have
\[ \sigma^\ell_1(A(a_i)) \leq \sqrt{\tau_i(A)} \leq \frac{1}{n^\delta}. \] (B.11)
Therefore, the quantity we return, \( \hat{s} \), satisfies the following approximation guarantee:

\[
\frac{1}{1 + \gamma} \hat{s} := \tilde{s} + \frac{1}{n^3} (|A| - |\tilde{A}|) \geq (1 - (\hat{\rho} + \rho)) \mathbb{E}(\tilde{A}) + \sum_{i : \alpha_i \not\in \tilde{A}} \sigma_i^A (a_i) \geq (1 - (\hat{\rho} + \rho)) \mathbb{E}(A),
\]

where the first step is by definition of \( \tilde{s} \) in Algorithm 4; the second step is by Equation (B.10); the third step is by applying Inequality (B.11) to each of the rows not present in \( \tilde{A} \). In the other direction,

\[
\frac{1}{1 + \gamma} \hat{s} = \tilde{s} + \frac{1}{n^3} (|A| - |\tilde{A}|) \leq (1 + (\hat{\rho} + \rho)) \mathbb{E}(\tilde{A}) + \frac{1}{n^3} \leq (1 + (\hat{\rho} + \rho)) \mathbb{E}(A),
\]

where the first step is by definition of \( \tilde{s} \) in Algorithm 4; the second step is by the fact that \(|A| \leq n\); the final step is by combining the assumption \( \rho \geq \frac{1}{C^2} \) and the facts that all sensitivities are non-negative and that the total \( \ell_1 \) sensitivity is at least one. Combining Inequality (B.12) and Inequality (B.13) finishes the proof.

Lemma B.8 (Single-Level Computation in Recursion Tree). Consider a node \( N \) in the recursion tree corresponding to Algorithm 5, and let \( M \) be the input matrix for \( N \). Further assume that \( N \) is not a leaf node. Denote by \( N_1, N_2, \ldots, N_B \) (where \( B = \Theta(|\log(n)|) \)) is the number of recursive calls at each node in Algorithm 5) the children of node \( N \), with each node \( N_i \) with the input matrix \( \tilde{M}_i \) (as constructed in Algorithm 5). Suppose each of these child nodes \( N_i \) satisfies a \( (\tilde{\rho}, \tilde{\delta}) \)-approximation guarantee (cf. Definition B.4). Then \( N \) satisfies a \( (\tilde{\rho} + \rho, (\tilde{\delta} + \delta) \cdot B) \)-approximation guarantee.

Proof. We first reiterate the steps of Algorithm 5 that are crucial to this proof. As proved in Lemma B.5, Algorithm 5 partitions the rows of the matrix \( M \) in node \( N \) into \( B \) submatrices \( M_1, M_2, \ldots, M_B \) based on their leverage scores. Next, for each matrix \( M_i \), we uniformly sample with replacement \( O(\sqrt{|C|}(1 + \rho)^{-2} \log(\delta^{-1})) \) rows and term this set of rows as matrix \( \tilde{M}_i \), which we then pass to Algorithm 5 recursively in node \( N_i \) (recall that we assume that the node \( N \) is not a leaf node). In other words, each child node \( N_i \) has as its input matrix \( \tilde{M}_i \).

Since we assume that all the child nodes \( N_1, N_2, \ldots, N_B \) satisfy a \( (\tilde{\rho}, \tilde{\delta}) \)-approximation guarantee as in Definition B.4, it means that, for all \( i \in [B] \), the output \( \tilde{s}_i \) of Algorithm 5 on node \( N_i \) satisfies the following guarantee:

\[
\tilde{s}_i \approx_{\tilde{\rho}} \mathbb{E} \sigma_i^{S'}(\tilde{M}_i) \text{ with a probability at least } 1 - \tilde{\delta}.
\]

Next, we recall that each \( M_i \) is composed of only those rows of \( M \) whose leverage scores (computed with respect to \( C \)) lie in the \( i \)th bucket \( [2^{-i}, 2^{-i+1}] \). Combining this with Fact 2.2, this implies that the \( \ell_1 \) sensitivity of the \( j \)th row \( M_i[j] \) of the matrix \( M_i \), computed with respect to \( C \), satisfies

\[
\sqrt{\frac{2^{-i}}{|C|}} \leq \frac{\tau_C^C(M_i[j])}{|C|} \leq \sigma_1^C(M_i[j]) \leq \sqrt{\frac{2^{-i + 1}}{|C|}}.
\]

Additionally, we claim that

\[
\frac{1}{3(1 + \rho)} \sigma_1^C(M_i[j]) \leq \sigma_1^{S'}(M_i[j]) \leq 3(1 + \rho) \sigma_1^C(M_i[j]).
\]

To see this, we observe that

\[
\sigma_1^C(M_i[j]) = \max_{x \in \mathbb{R}^d} \frac{|x^T M_i[j]|}{\|Cx\|_1} \geq \max_{x \in \mathbb{R}^d} \frac{|x^T M_i[j]|}{3|Ax|_1} \geq \max_{x \in \mathbb{R}^d} \frac{|x^T M_i[j]|}{3|Ax|_1} = \frac{\sigma_1^{S'}(M_i[j])}{3(1 + \rho)},
\]

where the first step is by the definition of \( \ell_1 \) sensitivity of \( M_i[j] \) (i.e., the \( j \)th row of matrix \( M_i \)) with respect to \( C \); the second step is by the definition of \( C \); the fact that \( S \) is a 1/2-factor \( \ell_1 \) subspace embedding of \( A \), and because \( \tilde{M} \) is a subset of rows of \( A \); the third step is because \( S' \) is a \( \rho \)-approximate \( \ell_1 \) subspace embedding for \( A \), and the final step is by the definition of \( \sigma_1^C(M_i[j]) \).

In the other direction, we have

\[
\sigma_1^C(M_i[j]) = \max_{x \in \mathbb{R}^d} \frac{|x^T M_i[j]|}{\|Cx\|_1} \leq \max_{x \in \mathbb{R}^d} \frac{|x^T M_i[j]|}{0.5\|Ax\|_1} \leq \max_{x \in \mathbb{R}^d} \frac{(1 + \rho)|x^T M_i[j]|}{0.5\|S'Ax\|_1} \leq 3(1 + \rho)\sigma_1^{S'}(M_i[j]),
\]

(18)
where the second step is by the definition of $C$ as a vertical concatenation of $SA$ and $M$ and further using that $SA$ is a constant factor $\ell_1$ subspace embedding of $A$ and dropping the non-negative term $||Mx||_1$. Combining Inequality (B.15) and Inequality (B.16), we see that for all rows $j \in ||[M]_i||$, we have

$$\frac{1}{3(1+\rho)}\sqrt{\frac{2^{-i}}{|C|}} \leq \sigma_j^{SA}(M_i[j]) \leq 3(1+\rho)\sqrt{2^{-i+1}}.$$  

(B.19)

Since Inequality (B.19) shows upper and lower bounds on each sensitivity $\sigma_j^{SA}(M_i)$ of the matrix $M_i$, we may use Lemma A.3 to approximate the sum of these sensitivities. In particular, by Lemma A.3, we may construct a matrix $\tilde{M}_i$ composed of $\frac{20\sqrt{|C|}(1+2\rho)\log(\delta^{-1})}{\rho^2}$ rows of $M_i$ sampled uniformly at random with replacement, and we are guaranteed

$$(1+\rho)\frac{|M_i|}{|\tilde{M}_i|} \mathbb{E}_{i_1}^{SA}(\tilde{M}_i) \approx \mathbb{E}_{i_1}^{SA}(M_i)$$

with a probability at least $1 - \delta$.  

(B.20)

Note that the output $\hat{s}$ of Algorithm 5 when applied to node $N$ is defined as

$$\hat{s} := (1 + \rho) \sum_{i \in [B]} \frac{|M_i|}{|\tilde{M}_i|} \tilde{s}_i.$$  

(B.21)

Therefore, by union bound over the failure probability (cf. Fact B.3), we can combine Equation (B.14) and sum Equation (B.20) over all $i \in B$ child nodes to conclude that $\hat{s}$ from Equation (B.21) satisfies a $(\hat{\rho} + \rho, (\hat{\delta} + \delta)B)$-approximation guarantee of Definition B.4.

\[\square\]

B.2.3 Proof of Theorem B.1

Proof of Theorem B.1. We first sketch our strategy to prove Theorem B.1’s correctness guarantee. Essentially, our goal is to establish Definition B.4 for the root node $T_{(1,1)}$. That this goal suffices to prove the theorem is justified in Lemma B.7. We now show that the node $T_{(1,1)}$ satisfies a $(\hat{\rho}, \hat{\delta})$-approximation guarantee for some $\hat{\rho}$ and $\hat{\delta}$. We use the method of induction.

Correctness guarantee of Theorem B.1. We know that the leaf nodes all satisfy the $(\rho, 0)$-approximation guarantee in Definition B.4 since at this level, the base case is triggered, where we compute a $\rho$-approximation estimate of the total $\ell_1$ sensitivity with respect to the subspace embedding $SA$. Having established this approximation guarantee at the leaf nodes, we can use Lemma B.8 to inductively propagate the property of $(\rho, \delta)$-approximation guarantee up the recursion tree $T$. From Lemma B.6, we have that the recursion tree $T$ of Algorithm 5 is of depth at most $D = O(\log \log(n + d))$, and the number of child nodes at any given node is at most $B = \Theta(\log(n))$; we factor these into the approximation guarantee and error probability.

Base case. At the leaf nodes of the recursion tree, we directly compute the total sensitivities with respect to $SA$. Recall that we denote the depth of the tree by $D$. Then, because $SA$ is, by definition, a $\rho$-approximate subspace embedding for $A$, we have, for each $i$ that indexes a leaf node,

$$\hat{s}_{(D,i)} \approx \rho \mathbb{E}_{i_1}^{SA}(M^{(D,i)})$$

with a probability of 1.  

(B.22)

Induction proof. Consider the subtree $T_j$ truncated at (but including) the nodes at level $j$ in the recursion tree $T$ corresponding to Algorithm 5. Assume the induction hypothesis that all the leaf nodes in the subtree $T_j$ satisfy Definition B.4 with parameters $\left(\rho \cdot (D + 1 - j), \delta \cdot \left(\frac{B^{D-j+1} - B}{B-1}\right)\right)$. That is, at the matrix of each such node $i$, we have an estimate $\hat{s}_{(j,i)}$ of its total sensitivity $\mathbb{E}_{i_1}^{SA}(M^{(j,i)})$ such that

$$\hat{s}_{(j,i)} \approx \rho \cdot (D + 1 - j) \mathbb{E}_{i_1}^{SA}(M^{(j,i)})$$

with a probability at least $1 - \delta \cdot \left(\frac{B^{D-j+1} - B}{B-1}\right)$.  

(B.23)

Then, each of the leaf nodes of the subtree $T_{j-1}$ (i.e., the subtree that stops one level above $T_j$) satisfies the assumption in Lemma B.8 since its child nodes are either leaf nodes of the entire recursion tree $T$ (for which this statement has been shown in Inequality (B.22)) or, if they are not
leaf nodes of the recursion tree, they satisfy Definition B.4 with parameters \( \tilde{\rho} = \rho \cdot (D + 1 - j) \) and \( \tilde{\delta} = \delta \cdot \left( \frac{B^{D-j+1} - B}{B - 1} \right) \). Therefore, Lemma B.8 implies that each of the leaf nodes of \( T_{j-1} \) satisfies Definition B.4 with approximation parameter \( \tilde{\rho} \) and error probability \( \tilde{\delta} \) defined as follows:

\[
\tilde{\rho} = \rho + \rho \cdot (D+1-j) = \rho \cdot (D+2-j) \quad \text{and} \quad \tilde{\delta} = B \cdot \left( \delta + \delta \cdot \left( \frac{B^{D-j+1} - B}{B - 1} \right) \right) = \delta \cdot \left( \frac{B^{D-j+2} - B}{B - 1} \right).
\]

In other words, the outputs \( \tilde{s}_{(j-1,k)} \) of Algorithm 5 on the matrices \( M^{(j-1,k)} \) in the leaf nodes of \( T_{j-1} \) each satisfy:

\[
\tilde{s}_{(j-1,k)} \approx_{O((\rho \cdot (D+2-j)))} S^{\rho \cdot A}(M^{(j-1,k)}) \quad \text{with a probability at least} \quad 1 - \delta \cdot \left( \frac{B^{D-j+2} - B}{B - 1} \right). \tag{B.24}
\]

The base case in Inequality (B.22) and the satisfaction of the induction hypothesis in Inequality (B.24) together finish the proof of the induction hypothesis.

**Finishing the proof of correctness.** In light of the above conclusion, for the root node \( T_{(1,1)} \), which is at the level \( j = 1 \), we may plug in \( j = 1 \) in Inequality (B.23) and obtain that the output \( \tilde{s}_{(1,1)} \) of Algorithm 5 at the root node satisfies

\[
\tilde{s}_{(1,1)} \approx_{O(D\rho)} S^{\rho \cdot A}(M^{(1,1)}) \quad \text{with a probability at least} \quad 1 - \delta \cdot \left( \frac{B^{D} - B}{B - 1} \right). \tag{B.25}
\]

In Algorithm 5, we choose the parameters \( \delta = 0.01 \frac{\rho}{\bar{\rho}} \) and \( \rho = O \left( \frac{1}{D} \right) \). Plugging these parameters into Inequality (B.25), we conclude that the root node satisfies Definition B.4 with a \((\gamma,0.01)\)-approximation guarantee. This finishes the proof of correctness of the claim.

**Proof of runtime.** Let \( b \) denote the maximum number of rows of a matrix at a leaf node of \( \mathcal{T} \). Further, recall from Fact 2.4 that the cost of computing one \( \ell_1 \) sensitivity of an \( n \times d \) matrix is,

\[
L = \tilde{O}(nnz(A) + d^{\omega}). \tag{B.26}
\]

Then as stated in Line 1, the algorithm computes the \( \rho \)-approximate total sensitivity at the leaf nodes; Fact 2.4, this incurs a cost of

\[
C(b) = \tilde{O}(nnz(A) + b \cdot d^{\omega}). \tag{B.27}
\]

At any other node, the computational cost is the work done at that level plus the total cost of the recursive calls at that level. Let \( C(r) \) denote the runtime of Algorithm 5 when the input is an \( r \times d \) matrix. Then,

\[
C(r) = \begin{cases} 
B \cdot C \left( \frac{\sqrt{r(1+\rho)}}{\rho^{\frac{\omega}{2}}} \log(\delta^{-1}) \right) + L & r > b \\
C(b) & r \leq b
\end{cases},
\]

where \( b \) is the number of rows in the base case, and \( L \) is the cost of computing leverage scores and equals \( L = \tilde{O}(nnz(A) + d^{\omega}) \) from Fact 2.1. For the depth of the recursion \( D \) (where the top level is \( D = 1 \)), plugging in our choices of \( \delta = \frac{0.01}{\bar{\rho}^{\omega}} \) and \( \rho = O \left( \frac{1}{D} \right) \), this may be expressed as via expanding the recursion as:

\[
C(n) = B^{D} \cdot C(b) + L \cdot \left( \frac{B^{D} - 1}{B - 1} \right) \tag{B.28}
\]

for \( n \geq b \). We plug into this expression the values of \( b \) and \( D \) from Lemma B.6, \( L \) from Equation (B.26), \( C(b) \) from Equation (B.27), and \( B = \Theta(\log(n)) \) to get the claimed runtime. \( \square \)

**B.3 Estimating the maximum \( \ell_1 \) sensitivity**

**Theorem 3.7 (Approximating the Maximum of \ell_1 Sensitivities).** Given a matrix \( A \in \mathbb{R}^{n \times d} \), there exists an algorithm, which in time \( \tilde{O}(nnz(A) + d^{\omega+1}) \), outputs a positive scalar \( \tilde{s} \) that satisfies

\[
\Omega(\|\sigma_1(A)\|_{\infty}) \leq \tilde{s} \leq O(\sqrt{d}\|\sigma_1(A)\|_{\infty}).
\]
Proof. First, we have
\[ \| S_1 A x \|_1 = \Theta(\| A x \|_1). \] (B.29)
We use Equation (B.29) to establish the claimed bounds below. First we set some notation. Define \( x^* \) and \( a_{i^*} \) as follows:
\[ x^*, i^* = \arg \max_{x \in \mathbb{R}^d, \ell \in [\| A \|]} \{ \| a_\ell^T x \| \} \] (B.30)
Thus, \( x^* \) is the vector that realizes the maximum sensitivity of the matrix \( A \), and the row \( a_{i^*} \) is the row of \( A \) with maximum sensitivity with respect to \( A \). Suppose the matrix \( S_\infty A \) contains the row \( a_{i^*} \). Then we have
\[ \max_{x \in \mathbb{R}^d} \sigma_1(x) = \max_{x \in \mathbb{R}^d, c_i \in S_\infty A} \frac{\| c_i^T x \|_1}{\| S_1 A x \|_1} = \Theta(1) \max_{x \in \mathbb{R}^d, c_i \in S_\infty A} \frac{\| c_i^T x \|_1}{\| A x \|_1} = \Theta(1) \max_{x \in \mathbb{R}^d} \frac{\| a_i^T x \|_1}{\| A x \|_1}, \] (B.31)
where the first step is by definition of \( \ell_1 \) sensitivity of \( S_\infty A \) with respect to \( S_1 A \), the second step is by Equation (B.29), and the third step by noting that the matrix \( S_\infty A \) is a subset of the rows of \( A \) (which includes \( a_{i^*} \)). By definition of \( \sigma_1(a_{i^*}) \) in Equation (B.30), we have
\[ \max_{x \in \mathbb{R}^d} \frac{\| a_i^T x \|_1}{\| A x \|_1} = \| \sigma_1(A) \|_\infty. \] (B.32)
Then, combining Equation (B.31) and Inequality (B.32) gives the guarantee in this case. In the other case, suppose \( a_{i^*} \) is not included in \( S_\infty A \). Then we observe that the upper bound from the preceding inequalities still holds. For the lower bound, we observe that
\[ \| \sigma_1^{S_1 A}(S_\infty A) \|_\infty = \max_{x \in \mathbb{R}^d, c_j \in S_\infty A} \frac{\| c_j^T x \|_1}{\| S_1 A x \|_1} = \Theta(1) \max_{x \in \mathbb{R}^d, c_j \in S_\infty A} \frac{\| S_\infty A x \|_\infty}{\| S_1 A x \|_1} = \Theta(1) \max_{x \in \mathbb{R}^d, c_j \in S_\infty A} \frac{\| S_\infty A x \|_\infty}{\| A x \|_1}, \] (B.33)
where the second step is by choosing a specific vector in the numerator and the third step uses \( \| S_1 A x \|_1 = \Theta(\| A x \|_1) \). We further have,
\[ \max_{x \in \mathbb{R}^d} \frac{\| S_\infty A x \|_\infty}{\| A x \|_1} \geq \frac{\| S_\infty A x^* \|_\infty}{\| A x^\star \|_1} = \frac{\| A x^\star \|_1}{\sqrt{d}} \leq \frac{1}{\sqrt{d}} \| \sigma_1(A) \|_\infty, \] (B.34)
where the first step is by choosing \( x = x^* \), the second step is by the distortion guarantee of \( \ell_\infty \) subspace embedding [35], and the final step is by definition of \( \sigma_1(a_{i^*}) \). Combining Inequality (C.10) and Inequality (C.11) gives the claimed lower bound on \( \| \sigma_1^{S_1 A}(S_\infty A) \|_\infty \). The runtime follows from the computational cost and row dimension of \( S_\infty A \) from [35] and the cost of computing \( \ell_1 \) sensitivities with respect to \( S_1 A \) as per Fact 2.4. \( \square \)

C Omitted proofs: \( \ell_p \) sensitivities

We recall the following notation that we use for stating our results in this setting.

Definition C.1 (Notation for \( \ell_p \) Results). We introduce the notation \( L_P(m, d, p) \) to denote the cost of approximating one \( \ell_p \) sensitivity of an \( m \times d \) matrix up to an accuracy of a given constant factor.

C.1 Estimating all \( \ell_p \) sensitivities

We generalize Algorithm 1 to the case \( p \geq 1 \) below.

Theorem C.2 (Approximating all \( \ell_p \) sensitivities). Given a full-rank matrix \( A \in \mathbb{R}^{n \times d} \), an approximation factor \( 1 < \alpha < n \), and a scalar \( p \geq 1 \), let \( \sigma_p(a_i) \) be the \( i \)th \( \ell_p \) sensitivity. Then there exists an algorithm that returns a vector \( \bar{\sigma} \in \mathbb{R}^n \) such that with high probability, for each \( i \in [n] \), we have
\[ \Omega(\sigma_p(a_i)) \leq \bar{\sigma}_i \leq O(\alpha^{p-1} \sigma_p(a_i)) + \frac{\alpha^{p}}{n} \bar{\sigma}_p(A). \] (C.1)
Our algorithm runs in time \( \tilde{O}(mnz(A) + \frac{n}{\alpha} \cdot LP(d^{\max(1,p/2)}, d, p)) \) (cf. Definition 1.2).
Algorithm 6 Approximating $\ell_p$-Sensitivities: Row-wise Approximation

| Inputs: Matrix $A \in \mathbb{R}^{n \times d}$, approximation factor $\alpha \in (1, n]$, scalar $p \geq 1$ |
|----------------------|-----------------------------------------------------------|
| Output: Vector $\tilde{\sigma} \in \mathbb{R}^n$ that satisfies, for each $i \in [n]$, with probability $0.9$, that $\sigma_p(a_i) \leq \tilde{\sigma}_i \leq O(\alpha^{p-1} \sigma_p(a_i) + \frac{n}{\alpha} \mathbb{E}_p(A))$ |

1: Compute, for $A$, an $\ell_p$ subspace embedding $S_pA \in \mathbb{R}^{\tilde{n}} \tilde{O}(d^{max(1, p/2)}) \times d$
2: Partition $A$ into $\tilde{n}$ blocks $B_1, \ldots, B_{n/\tilde{n}}$ each comprising $\alpha$ randomly selected rows.
3: for the $\ell_p$th block $B_\ell$, with $\ell \in [\tilde{n}]$ do
4: Sample $\alpha$-dimensional independent Rademacher vectors $r_1^{(\ell)}, \ldots, r_{100}^{(\ell)}$
5: For each $j \in [100]$, compute the row vectors $r_j^{(\ell)}B_\ell \in \mathbb{R}^d$
6: end for
7: Let $P \in \mathbb{R}^{100\tilde{n} \times d}$ be the matrix of all vectors computed in Line 5. Compute $\sigma_p^{S_pA}(P)$ using Definition 1.2
8: for each $i \in [n]$ do
9: Denote by $J$ the set of row indices in $P$ that $a_i$ is mapped to in Line 5
10: Set $\tilde{\sigma}_i = \max_{j \in J} (\sigma_p^{S_pA}(P_j))$
11: end for
12: Return $\tilde{\sigma}$

Proof. Suppose the $i$th row of $A$ falls into the bucket $B_\ell$. Suppose, further, that the rows from $B_\ell$ are mapped to those in $P$ with row indices in the set $J$. Then, Algorithm 6 returns a vector of sensitivity estimates $\sigma \in \mathbb{R}^n$ with the $i$th coordinate defined as $\sigma_i = \max_{j \in J} \sigma_p(a_j)$. For any $x \in \mathbb{R}^d$, we have

$$\|S_pAx\|_p \approx \|Ax\|_p.$$  \hfill (C.2)

We use Equation (C.2) to establish the claimed bounds below. Let $x^* = \arg \max_{x \in \mathbb{R}^d} |a^\top x|_p$ be the vector that realizes the $i$th $\ell_p$ sensitivity of $A$. Further, let the $j$th row of $P$ be $r^{(\ell)}_{k,j}B_\ell$. Then, with probability of at least $1/2$, we have

$$\sigma_p^{S_pA}(p_j) = \max_{x \in \mathbb{R}^d} \frac{|r_{k,j}^{(\ell)}B_\ell x|_p}{\|S_pAx\|_p} \geq \frac{|r_{k,j}^{(\ell)}B_\ell x^*|_p}{\|S_pAx^*\|_p} = \Theta(1) \frac{|r_{k,j}^{(\ell)}B_\ell x^*|_p}{\|Ax^*\|_p} = \Theta(1) \sigma_p(a_j),$$  \hfill (C.3)

where the first step is by definition of $\sigma_p(a_j)$, the second is by definition of $x^*$, the third is by Equation (C.2), and the final step is by definition of $x^*$ and $\sigma_p(a_j)$. For the fourth step, we use the fact that $|r_{k,j}^{(\ell)}B_\ell x^*|_p = |r_{k,j}^{(\ell)}a_j^\top x^* + \sum_{j \neq k} r_{k,j}^{(\ell)}(Bx^*)_j|_p \geq |a_j^\top x^*|_p$ with a probability of at least $1/2$ since the vector $r_{k,j}^{(\ell)}$ has coordinates that are $-1$ or $+1$ with equal probability. By a union bound over $j \in J$ independent rows that block $B_\ell$ is mapped to, we establish the claimed lower bound in Inequality (C.3) with probability at least $0.9$. To show an upper bound on $\sigma_p^{S_pA}(p_j)$, we observe that

$$\max_{x \in \mathbb{R}^d} \frac{|r_{k,j}^{(\ell)}B_\ell x^*|_p}{\|S_pAx\|_p} \leq \alpha^{p-1} \max_{x \in \mathbb{R}^d} \frac{\|B_\ell x^*\|_p}{\|S_pAx\|_p} \leq \Theta(\alpha^{p-1}) \sum_{j : a_j \in B_\ell} \max_{x \in \mathbb{R}^d} \frac{|a_j^\top x^*|_p}{\|Ax^*\|_p} = \Theta(\alpha^{p-1}) \sum_{j : a_j \in B_\ell} \sigma_p(a_j),$$  \hfill (C.4)

where the second step is by Equation (C.2) and opening up $\|B_\ell x\|_p$ in terms of the rows in $B_\ell$, and the final step is by definition of $B_\ell$. To see the first step, we observe that

$$|r_{k,j}^{(\ell)}B_\ell x^*|_p \leq \|B_\ell x\|_p \cdot \|r_{k,j}^{(\ell)}\|_q \leq \|B_\ell x\|_p \cdot \alpha^{p/q} = \|B_\ell x\|_p \cdot \alpha^{p-1},$$

where $\frac{1}{p} + \frac{1}{q} = 1$, the first step is by Hölder’s inequality, the second is because each entry of $r_{k,j}^{(\ell)}$ is either $+1$ or $-1$ and $r_{k,j}^{(\ell)} \in \mathbb{R}^\alpha$, and the third is because $\frac{p}{q} = p - 1$. Because $B_\ell$ is a group of $\alpha$ rows
The runtime of the algorithm is

\[ \mathbb{E} \left\{ \sum_{j:a_j \in \mathcal{B}, j \neq i} \sigma_p(a_j) \right\} = \frac{\alpha - 1}{n - 1} \sum_{j \neq i} \sigma_p(a_i). \]

Therefore, Markov inequality gives us that with a probability of at least 0.9, we have

\[
\sigma_p^{S_pA}(p_j) \leq O(\alpha^{p-1} \sigma_p(a_i) + \frac{\alpha^p}{n} \mathbb{E}(\sigma_p(A))). \tag{C.5}
\]

The median trick then establishes Inequality (C.4) and Inequality (C.5) with high probability. To establish the runtime, note that we first construct the subspace embedding \( S_pA \) and then compute the \( \ell_p \) sensitivities of \( P \) (with \( O(n/\alpha) \) rows) with respect to \( S_pA \). The size of the subspace embedding \( S_pA \) is \( O(d^{1/2} \times d) \) (see [75, Table 1]). This completes the runtime claim. \( \square \)

### C.2 Estimating the maximum of \( \ell_p \) sensitivities

In section 3.2 and Appendix B.3, we showed our result for estimating the maximum \( \ell_1 \) sensitivity. In this section, we show how to extend this result to all \( \ell_p \) sensitivities. Algorithm 7 generalizes Algorithm 3.

**Algorithm 7 Approximating the Maximum of \( \ell_p \)-Sensitivities**

- **Input**: Matrix \( A \in \mathbb{R}^{n \times d} \) and \( p \geq 1 \) (with \( p \neq 2 \))
- **Output**: Scalar \( \tilde{s} \in \mathbb{R}_{\geq 0} \) that satisfies that
  \[
  \|\sigma_p(A)\|_\infty \leq \tilde{s} \leq C \cdot d^{p/2} \cdot \|\sigma_p(A)\|_\infty
  \]

1. Compute, for \( A \), an \( \ell_\infty \) subspace embedding \( S_\infty A \in \mathbb{R}^{O(d \log^2(d)) \times d} \) such that \( S_\infty A \) is a subset of the rows of \( A \) \[35\]
2. Compute, for \( A \), an \( \ell_p \) subspace embedding \( S_pA \in \left\{ \begin{array}{ll} \mathbb{R}^{O(d \times d)} & p \in [1, 2) \\
\mathbb{R}^{O(d^{p/2} \times d)} & p > 2 \end{array} \right. \)
3. Return \( d^{p/2} \cdot \|\sigma_p^{S_pA}(S_\infty A)\|_\infty \)

**Theorem C.3** (Approximating the maximum of \( \ell_p \) sensitivities). Given a matrix \( A \in \mathbb{R}^{n \times d} \) and \( p \geq 1 \) (with \( p \neq 2 \)), there exists an algorithm, which outputs a positive scalar \( \tilde{s} \) that satisfies

\[
\Omega(\|\sigma_p(A)\|_\infty) \leq \tilde{s} \leq O(d^{p/2} \|\sigma_p(A)\|_\infty).
\]

The runtime of the algorithm is \( \tilde{O}(\text{nnz}(A) + d^{\omega+1}) \) for \( p \in [1, 2) \) and \( \tilde{O}(\text{nnz}(A) + d^{p/2} \cdot \text{LP}(O(d^{p/2}, d, p)) \) for \( p > 2 \).

**Proof.** First, we have

\[
\|S_pAx\|^p_p = \Theta(\|Ax\|^p_p). \tag{C.6}
\]

We use Equation (C.6) to establish the claimed bounds below. First we set some notation. Define \( x^* \) and \( a_i^* \) as follows:

\[
x^*, i^* = \arg \max_{x \in \mathbb{R}^d, i \in [|A|]} \frac{|a_i^\top x|^p}{\|Ax\|^p_p} \tag{C.7}
\]

Thus, \( x^* \) is the vector that realizes the maximum sensitivity of the matrix \( A \), and the row \( a_i^* \) is the row of \( A \) with maximum sensitivity with respect to \( A \). Suppose the matrix \( S_\infty A \) contains the row \( a_i^* \). Then we have

\[
\max_{c_i \in S_\infty A} \sigma_p^{S_pA}(c_i) = \max_{x \in \mathbb{R}^d, c_i \in S_\infty A} \frac{|c_i^\top x|^p}{\|S_pAx\|^p_p} = \Theta(1) \max_{x \in \mathbb{R}^d, c_i \in S_\infty A} \frac{|c_i^\top x|^p}{\|Ax\|^p_p} = \Theta(1) \max_{x \in \mathbb{R}^d} \frac{|a_i^\top x|^p}{\|Ax\|^p_p}. \tag{C.8}
\]

\footnote{For \( p = 2 \), the result of [23] gives a constant factor approximation to all leverage scores in \( \text{nnz}(A) + d^\omega \) time.}
where the first step is by definition of $\ell_p$ sensitivity of $S_\infty A$ with respect to $S_p A$, the second step is by Equation (C.6), and the third step by noting that the matrix $S_\infty A$ is a subset of the rows of $A$ (which includes $a_i$). By definition of $\sigma_p(a_i \cdot)$ in Equation (C.7), we have

$$\max_{x \in \mathbb{R}^d} \frac{|a_i^T x|^p}{\|Ax\|_p^p} = \|\sigma_p(A)\|_\infty.$$  \hspace{1cm} (C.9)

Then, combining Equation (C.8) and Equation (C.9) gives the guarantee in this case. In the other case, suppose $a_i$ is not included in $S_\infty A$. Then we observe that the upper bound from the preceding inequalities still holds. For the lower bound, we observe that

$$\|\sigma_{S_p}^p(A_\infty A)\|_\infty = \max_{x \in \mathbb{R}^d, c \in S_\infty A} \frac{|c^T x|^p}{\|SpAx\|_p^p} \geq \max_{x \in \mathbb{R}^d} \frac{S_\infty Ax\|_\infty^p}{\|SpAx\|_p^p} = \Theta(1) \max_{x \in \mathbb{R}^d} \frac{\|S_\infty Ax\|_\infty^p}{\|Ax\|_p^p},$$

where the the second step is by choosing a specific vector in the numerator and the third step uses $\|SpAx\|_p^p = \Theta(\|Ax\|_p^p)$. We further have,

$$\max_{x \in \mathbb{R}^d} \frac{\|S_\infty Ax\|_\infty^p}{\|Ax\|_p^p} \geq \frac{\|\sigma_p(A^*)\|_\infty^p}{\|Ax^*\|_p^p} \geq \frac{|a_i^T x^*|^p}{dp/2 \|Ax^*\|_p^p} = \frac{1}{dp/2} \|\sigma_p(A)\|_\infty^p, \hspace{1cm} (C.11)$$

where the first step is by choosing $x = x^*$, the second step is by the guarantee of $\ell_\infty$ subspace embedding, and the final step is by definition of $\sigma_p(a_i \cdot)$). Combining Inequality (C.10) and Inequality (C.11) gives the claimed lower bound on $\|\sigma_{S_p}^p(A_\infty A)\|_\infty$. The runtime follows from the cost of computing $S_\infty A$ from [35] and the cost of computing and size of $S_p A$ from [23, Figure 1] \qed

D Lower Bounds

**Theorem D.1 ({$\ell_p$} Regression Reduces to {$\ell_p$} Sensitivities).** Suppose that we are given an algorithm $\mathcal{A}$, which for any matrix $A' \in \mathbb{R}^{n \times d}$ and accuracy parameter $\varepsilon' \in (0, 1)$, computes $(1 \pm \varepsilon')\sigma_p(A')$ in time $T(n', d', m nz(A'), \varepsilon')$. Then, there exists an algorithm that takes $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^n$ as inputs and computes $(1 \pm \varepsilon) \min_{y \in \mathbb{R}^d} \|Ay - b\|_p^p \pm \lambda p \varepsilon$ in time $T(n + 1, d + 1, m nz(A) + m nz(b) + 1, \varepsilon)$ for any $\lambda > 0$.

**Proof.** Given $A \in \mathbb{R}^{n \times d}$ and $b \in \mathbb{R}^d$, consider the matrix $A' := \begin{bmatrix} A & -b \\ 0^T & -\lambda \end{bmatrix} \in \mathbb{R}^{(n + 1) \times d}$. Then the $n + 1$-th $\ell_p$ sensitivity of $A'$ is

$$\sigma_p(a'_{n+1}) = \max_{x \in \mathbb{R}^d} \frac{|e_d, x|^p}{\|A'x\|_p^p} = \max_{x \in \mathbb{R}^d, x = \lambda^{-1}} \frac{1}{\|A'x\|_p^p} = \frac{1}{1 + \min_y \|Ay - \lambda b\|_p^p}.$$

where the second step is by the scale-invariance of the definition of sensitivity with respect to the variable of optimization. By scale-invariance we see that for any $c > 0$, and $\min_y \|A_{-i}/c y - A_{-i}/b\|_p^p = c^p \min_y \|A_{-i}/c y - A_{-i}/b\|_p^p$.

Therefore, note that rearranging gives $\min_y \|Ay - b\|_p^p = \frac{\lambda p}{\sigma_p(a'_{n+1})} - \lambda p$, which is how we can estimate the cost of the regression. Then computing $\sigma_p(a'_{n+1})$ to a multiplicative accuracy of $\varepsilon'$ gives the value of $\min_y \|Ay - b\|_p^p$ up to error $\varepsilon \frac{\lambda p}{\sigma_p(a'_{n+1})} = \varepsilon \lambda p + \min_y \|Ay - b\|_p^p$, giving our final bounds. \qed

Using the same technique as in Theorem D.1, we can similarly extend the reduction to a multiple regression task. In particular, we show that computing the values of a family of certain regularized leave-one-out regression problems for a matrix may be obtained by simply computing leverage scores of an associated matrix; therefore, this observation demonstrates that finding fast algorithms for sensitivities is as hard as multiple regression tasks. Specifically, for some matrix $A \in \mathbb{R}^{n \times d}$, we denote $A_{-i} \in \mathbb{R}^{n \times d-1}$ as the submatrix of $A$ with its $i$-th column, denoted as $A_{-i}$, removed. We show that approximate sensitivity calculations can solve $\min_y \|A_{-i}/b + A_{-i}/y\|_2$ approximately for all $i$. 29
Lemma D.2 (Regularized Leave-One-Out $\ell_p$ Multiregression Reduces to $\ell_p$ Sensitivities). Suppose that we are given an algorithm $A$, which for any matrix $A' \in \mathbb{R}^{n' \times d'}$ and accuracy parameter $\varepsilon' \in (0, 1)$, computes $(1 \pm \varepsilon'\sigma_p(A'))$ in time $T(n', d', \text{nnz}(A'), \varepsilon')$. Given a matrix $A \in \mathbb{R}^{n \times d}$ with $n \geq d$, let $OPT_i := \min_{y \in \mathbb{R}^{d-1}} \|A_{-i} y + A_{-i}\|_p$ and $y_i^* := \arg \min_{y \in \mathbb{R}^{d-1}} \|A_{-i} y + A_{-i}\|_p$ for all the $i \in [d]$. Then, there exists an algorithm that takes $A \in \mathbb{R}^{n \times d}$ and computes $(1 \pm \varepsilon)OPT_i \pm \lambda^p(1 + \|y_i^*\|_p^p)$ in time $T(n + d, d, \text{nnz}(A), \varepsilon)$ for any $\lambda > 0$.

Proof. Given $A \in \mathbb{R}^{n \times d}$, consider the matrix $A' := \begin{bmatrix} A & \lambda I \end{bmatrix} \in \mathbb{R}^{(n+d) \times d}$ formed by vertically appending a scaled identity matrix to $A$, with $\lambda > 0$. By the definition of $\ell_p$ sensitivities, we have the following.

$$
\sigma_p(a_{n+i}') = \max_{x \in \mathbb{R}^d} \frac{|x^T a_{n+i}'|_p}{\|A'x\|_p} = \max_{x \in \mathbb{R}^d : x_1 = \lambda^{-1}} \frac{1}{\|A'x\|_p} = \frac{1}{\min_{y \in \mathbb{R}^{d-1}} \{1 + \lambda^p\|y\|_p + \|A_{-i}y + \lambda^{-1}A_{-i}\|_p\}}
$$

Recall that $y_i^* := \arg \min_{y \in \mathbb{R}^{d-1}} \|A_{-i} y + A_{-i}\|_p$, and since the power is a monotone transform, $OPT_i := \min_{y \in \mathbb{R}^{d-1}} \|A_{-i} y + A_{-i}\|_p$. By scale-invariance we see that for any $c > 0$, $y_i^* = \arg \min_{y \in \mathbb{R}^{d-1}} \|cA_{-i} y + cA_{-i}\|_p$ and $\min_y \|A_{-i} y + cA_{-i}\|_p = c^p \text{OPT}_i$. Therefore, by plugging $y = \lambda^{-1}y_i^*$, note that

$$
\min_{y \in \mathbb{R}^{d-1}} \{1 + \lambda^p\|y\|_p + \|A_{-i} y + \lambda^{-1}A_{-i}\|_p\} \leq 1 + \|y_i^*\|_p^p + \lambda^{-p}\text{OPT}_i.
$$

However, we also note that by non-negativity,

$$
\min_{y \in \mathbb{R}^{d-1}} \{1 + \lambda^p\|y\|_p^p + \|A_{-i} y + \lambda^{-1}A_{-i}\|_p\} \geq \lambda^{-p}\text{OPT}_i.
$$

Combining this all together gives the claimed approximation by noting that

$$
\text{OPT}_i + \lambda^p\|y_i^*\|_p^p + \lambda^p \geq \frac{\lambda^p}{\sigma_p(a_{n+i}')} \geq \text{OPT}_i
$$

Therefore, our algorithm is to simply return $\lambda^p/s_p(a_{n+i}')$, where $s$ is the $\varepsilon$-approximate estimate of the sensitivity and our bound follows. The runtime guarantees follow immediately from assumption that the runtime of calculating the $\ell_p$ sensitivities of $A'$.

**Corollary D.3.** Assuming that leave-one-out $\ell_p$ multi-regression with $\Omega(d)$ instances takes $\text{poly}(d) \text{nnz}(A) + \text{poly}(1/\varepsilon)$ time [44, 45], computing $\ell_p$ sensitivities of an $n \times d$ matrix $A$ costs at least $\text{poly}(d) \text{nnz}(A) + \text{poly}(1/\varepsilon)$.

### E Additional Experiments

Here we include additional experiments on other similar datasets.

<table>
<thead>
<tr>
<th>$p$</th>
<th>Total Sensitivity Upper Bound</th>
<th>Brute-Force Approximation</th>
<th>Brute-Force Runtime (s)</th>
<th>Approximate Runtime (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11</td>
<td>4.7</td>
<td>3.9</td>
<td>1940</td>
</tr>
<tr>
<td>1.5</td>
<td>11</td>
<td>8.3</td>
<td>10.3</td>
<td>1980</td>
</tr>
<tr>
<td>2.5</td>
<td>20</td>
<td>10.3</td>
<td>11.8</td>
<td>1970</td>
</tr>
<tr>
<td>3</td>
<td>36.4</td>
<td>6.9</td>
<td>7.3</td>
<td>1970</td>
</tr>
</tbody>
</table>

Table 2: Runtime comparison for computing total sensitivities for the *fires* dataset, which has matrix shape $(517, 11)$. We include the theoretical upper bound for the total sensitivities using lewis weights calculations.
<table>
<thead>
<tr>
<th>$p$</th>
<th>Total Sensitivity</th>
<th>Brute-Force</th>
<th>Approximation</th>
<th>Brute-Force Runtime (s)</th>
<th>Approximate Runtime (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11</td>
<td>4.1</td>
<td>5.2</td>
<td>540</td>
<td>154</td>
</tr>
<tr>
<td>1.5</td>
<td>11</td>
<td>10.1</td>
<td>6.7</td>
<td>560</td>
<td>201</td>
</tr>
<tr>
<td>2.5</td>
<td>20</td>
<td>19.9</td>
<td>12.2</td>
<td>390</td>
<td>117</td>
</tr>
<tr>
<td>3</td>
<td>36.4</td>
<td>8.8</td>
<td>6.3</td>
<td>354</td>
<td>136</td>
</tr>
</tbody>
</table>

Table 3: Runtime comparison for computing total sensitivities for the concrete dataset, which has matrix shape (101, 11). We include the theoretical upper bound for the total sensitivities using lewis weights calculations.

Figure 2: Average absolute log ratios for all $\ell_p$ sensitivity approximations for fires.