Subspace Embeddings for the L1-norm with Applications

Christian Sohler *
Department of Computer Science
Technische Universität Dortmund
christian.sohler@tu-dortmund.de

David P. Woodruff
IBM-Research Almaden
dpwoodru@us.ibm.com

ABSTRACT

We show there is a distribution over linear mappings $R : \ell_1^n \to \ell_1^{O(d \log d)}$, such that with arbitrarily large constant probability, for any fixed d-dimensional subspace L, for all $x \in L$ we have $\|x\|_1 \leq \|Rx\|_1 = O(d \log d)\|x\|_1$. This provides the first analogue of the ubiquitous subspace Johnson-Lindenstrauss embedding for the ℓ_1-norm. Importantly, the target dimension and distortion are independent of the ambient dimension n. We give several applications of this result. First, we give a faster algorithm for computing well-conditioned bases. Our algorithm is simple, avoiding the linear programming machinery required of previous algorithms. We also give faster algorithms for least absolute deviation regression and ℓ_1-norm best fit hyperplane problems, as well as the first single pass streaming algorithms with low space for these problems. These results are motivated by practical problems in image analysis, spam detection, and statistics, where the ℓ_1-norm is used in studies where outliers may be safely and effectively ignored. This is because the ℓ_1-norm is more robust to outliers than the ℓ_2-norm.

Categories and Subject Descriptors

F.2.1 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems; G.3 [**Probability and Statistics**]: Robust Regression

General Terms

Algorithms, Theory

Keywords

data stream algorithms, hyperplane fitting, regression, well-conditioned basis

*This research was supported by the DFG, Collaborative Research Center SFB876, project C4, and the EU-project CG Learning funded by the Future and Emerging Technologies unit of the European Commission within the 7th Framework Programme under contract No. 255827.

STOC'11, June 6–8, 2011, San Jose, California, USA.

1. INTRODUCTION

The Johnson-Lindenstrauss transform is a widely-used dimensionality reduction technique with applications to many areas such as compressed sensing [13], computational geometry [17], data stream algorithms [4], graph sparsification [46], machine learning [35, 45, 50], nearest-neighbor search [2, 28], and numerical linear algebra [18, 21, 41, 42]. The mapping is given by an appropriately scaled projection matrix $R : \ell_2^n \to \ell_2^k$, where $k = \Theta(\varepsilon^{-2} \log 1/\delta)$, with the guarantee that for any fixed vector $x \in \mathbb{R}^n$, with probability at least $1 - \delta$, $(1 - \varepsilon)\|x\|_2 \leq \|Rx\|_2 \leq (1 + \varepsilon)\|x\|_2$. Since R is a linear map, it also preserves pairwise ℓ_2-distances of a set of vectors. While there are other ℓ_2-sketches that have faster update time, such as the sketch by Thorup and Zhang [49], these are not embeddings into a normed space since the sketches involve inherently non-linear median operations. For many applications, such as ℓ_2-regression and low rank approximation of a matrix, this is inadequate, as the algorithms work by solving an optimization problem in the sketch-space. If the sketch-space is not normed or convex, solving these problems becomes intractable.

A natural question is whether the Johnson-Lindenstrauss transform extends to other ℓ_p-norms, in particular, the ℓ_1-norm. This question has inherent theoretical appeal, but also has many algorithmic applications. Indyk [26], in his FOCS 2001 tutorial, asks "Is there an analogue of the JL lemma for other norms, especially for ℓ_1? This would give a powerful technique for designing approximation algorithms for ℓ_1 norms..." Charikar and Sahai [14] showed that any linear embedding of r points in $\mathbb{R}^n$ into $\mathbb{R}^k$ must incur distortion $\Omega(\sqrt{r/k})$, which is tight up to a $\log r$ factor. Brinkman and Charikar [9] resolved the case of non-linear embeddings, showing that any embedding of r points with constant distortion must be embedded into $r^{\Omega(1)}$ dimensions; see also [36]. As with the JL transform, linear embeddings are most useful for the applications above, due to the fact that they can be compactly represented and updated efficiently in a distributed or streaming setting. Moreover, a key property of the JL transform is that the matrices R are independent of the point set, allowing for one to combine point sets with the same mapping or embed individual points without reading the entire pointset. We call a mapping R satisfying this property *oblivious*.

Given these negative results, our task of obtaining an analogue of the JL transform for the ℓ_1-norm might seem hopeless. However, the pointsets one works with often have additional structure, e.g., they sit in a low-dimensional manifold or subspace. This is true for regression, a problem which

is equivalent to finding the nearest point in a given low-dimensional subspace to an input point, or in low-rank approximation where one wants to preserve low-dimensional subspaces of the input which well-approximate it. This raises an intruiging question: how many dimensions are needed to embed a d-dimensional subspace of $\mathbb{R}^n$ into $\mathbb{R}^k$? That is, for a distortion bound D, if A is an $n \times d$ matrix of rank d and R is a mapping from ℓ_1^n into ℓ_1^k for which $\|y\|_1 \leq \|Ry\|_1 \leq D\|y\|_1$ for all $y = Ax$, $x \in \mathbb{R}^d$, how large does k need to be? The lower-bound instances of [9, 14] do not apply since their sets of r points have rank $r^{\Omega(1)}$. Moreover, as far as we are aware, there are no known linear oblivious embeddings of subspaces of ℓ_1 into ℓ_1, though there are non-oblivious maps [8, 30, 43, 47] that are based on the change of density technique [37], as well as embeddings from subspaces of ℓ_p into ℓ_1 for $p > 1$ [7, 23]. These embeddings have $D = O(1)$ and $k = O(d)$. We note that there are linear oblivious lopsided embeddings that guarantee for any *fixed* $y = Ax$, $\|y\|_1 \leq \|Ry\|_1$, and for any $C > 1$, with probability $1 - O(1/C)$, $\|Ry\|_1 \leq C\|y\|_1$ [27]. This upper tail is not strong enough to obtain low-distortion embeddings for subspaces.

1.1 Our Contributions

Oblivious embeddings for subspaces of ℓ_1. We give the first linear oblivious embeddings of subspaces of ℓ_1^n into $\ell_1^{O(d \log d)}$. Namely, we show there is a distribution over linear mappings R with the property that with arbitrarily large constant probability, for any fixed d-dimensional subspace L of ℓ_1^n, for all $x \in L$, we have $\|x\|_1 \leq \|Rx\|_1 = O(d \log d)\|x\|_1$. This provides the first analogue of the subspace Johnson-Lindenstrauss embedding for the ℓ_1-norm. The target dimension $O(d \log d)$ and distortion $O(d \log d)$ are both independent of n, mapping constant-dimensional subspaces of $\mathbb{R}^n$ to a constant number of dimensions with constant distortion. This result has the following applications to computation of well-conditioned bases, least absolute deviation regression (ℓ_1-regression), and the ℓ_1-norm best fit hyperplane problems.

Well-conditioned bases. One problem when dealing with norms other than l_2 is that the length of a vector is not invariant under rotations. Consider an $n \times m$ matrix A with rank d. While in l_2 we have that for a fixed vector x the norm of $\|Ax\|_2$ is identical for every matrix A whose columns are orthonormal, there is no analogue for other ℓ_p-norms. For the case of ℓ_p-norms there is a generalization/approximation of orthonormal bases called well-conditioned bases [16, 19]. The goal is to devise a basis U for the column space of A so that:

$$\forall x \in \mathbb{R}^d, \ \frac{1}{\text{poly}(d)} \cdot \|x\|_p \leq \|Ux\|_p \leq \text{poly}(d) \cdot \|x\|_p$$

(although the precise requirements look a bit different from this; see the formal definition in Section 2). Notice that the norm of x does not increase by more than a poly(d) factor, which is independent of the ambient dimension n. The algorithms of [16, 19] have time $O(nm^5 \log n + m^{O(1)})$ for computing a well-conditioned basis for ℓ_1. Moreover, these algorithm require the computation of Löwner-John ellipsoids, and seem unlikely to be very practical. In contrast, for ℓ_1 we obtain a faster algorithm with total time $(nd^{\omega-1} + nm + d^3) \cdot \text{polylog}(m)$, where $\omega < 2.376$ is the exponent of matrix multiplication. Importantly, our algorithm is conceptually simple and easy to implement: (1) compute RA, where R is a linear oblivious embedding for subspaces of ℓ_1, (2) compute an $m \times d$ matrix X so that RAX is orthonormal (in the standard ℓ_2-sense), and (3) output AX.

Least-absolute deviation regression. Regression is a basic method to study the dependencies between variables in the presence of noise, arising, for example, from experimental measurements. In the ℓ_1-regression problem, we are given an $n \times m$ matrix A of rank d and an n-dimensional vector b, and the problem is to compute $\text{argmin}_x \|Ax - b\|_1$. It is well-known that ℓ_1-regression is more robust than least squares regression, that is, the problem of computing $\text{argmin}_x \|Ax - b\|_2^2$ [39]. Regression has a number of applications to data mining and machine learning [15, 25], occurring, for instance, in calculating trend lines for business analytics, clinical trials, environmental science, and spam detection, see, e.g., the books [44, 51].

We give an algorithm for the ℓ_1-regression problem for any values of n, m, d, and ε, which runs in time:

$$O((nd^{\omega-1+\beta} + nm + L1R(d^{3.5}/\varepsilon^2, d)) \cdot \text{polylog}(m)),$$

were ω is the exponent of matrix multiplication, and $\beta > 0$ is an arbitrarily small constant. Here L1R(k, d) denotes the time to solve an instance of ℓ_1-regression on a $k \times d$ matrix with a k-dimensional column vector, and is bounded by poly(kd). Our algorithm improves the previous $O(nm^5 \log n +$ poly$(m\varepsilon^{-1}))$-time algorithms of Clarkson [16] and Dasgupta et al. [19]. Even for constant m, d and ε, our algorithm improves the time of previous algorithms from $O(n \log n)$ to $O(n)$, resolving an open question in [16]. The idea behind our algorithm is to compute RA and Rb, where R is a linear oblivious embedding for subspaces of ℓ_1, and to perform most of the expensive algorithms of previous work in the low-dimensional sketch space. Importantly, our method can also be implemented as a single-pass streaming algorithm with poly$(m\varepsilon^{-1} \log n)$ bits of space, where the rows of A are presented in an arbitrary order. Previous algorithms [16, 19] did not have this property, even if seeing the entries of A in row order, because it was necessary to compute a well-conditioned basis of the matrix A, which required all the entries to be stored. Other algorithms with this property either require time that is exponential in m [22], or exponential in ε^{-1} [27]. More generally, our algorithm can be implemented in the turnstile model of streaming [40] where the entries appear in an arbitrary order and undergo any number of additive updates. The algorithm is 1-pass, has poly$(m\varepsilon^{-1} \log n)$ space and poly$(m\varepsilon^{-1} \log n)$ processing time. Streaming algorithms for regression are studied in [18, 22] as a means of coping with massive data sets; our work extends these works to give the first efficient such implementation for the important case of ℓ_1. Our ideas extend straightforwardly to the generalized regression problem in which X and B are matrices [19].

ℓ_1-norm best fit hyperplane. This is the problem of fitting a hyperplane through n points in $\mathbb{R}^m$, while minimizing the sum of ℓ_1-distances of the n points to the hyperplane. Ke and Kanade [32, 33] study this problem in the context of image analysis using the affine camera model [33]. Agarwal et al. [1] extend this to the perspective camera model. Kwak studies this problem for face recognition data [34], and a formal analysis is given, together with an implementation, by Brooks and Dulá [10]. The running time of the algorithm is polynomial in n and m and requires solving m LPs [10]. In

general, the ℓ_1-norm provides a more robust measure than the ℓ_2-norm for these problems [11]. This is true, for example, in the context of covariance matrix estimation [12, 24].

As algorithms for this problem involve m invocations of an ℓ_1-regression algorithm, we can use our more efficient regression algorithm to improve the space and time complexities for this problem. Moreover, our algorithm is the first for this problem that is a single-pass algorithm in the turnstile model. It uses only $\text{poly}(m\varepsilon^{-1}\log n)$ bits of space. This problem is a special case of approximating a matrix with one of lower rank, a topic that is important in a streaming context [18, 42] (see also Section 7.10 of [40]).

1.2 Roadmap

In Section 2 we discuss preliminaries. Section 3 contains our linear oblivious subspace embeddings for subspaces of ℓ_1. In Section 4 we give our first application to faster computation of well-conditioned bases. In Section 5, we solve the ℓ_1-regression problem. In Section 6 we show how to solve the ℓ_1-norm best fit hyperplane problem.

2. PRELIMINARIES

For a vector $x = (x_1, \ldots, x_m)^T \in \mathbb{R}^m$ we use $\|x\|_p = \sum_{i=1}^m (|x_i|^p)^{\frac{1}{p}}$ to denote its p-norm. The dual norm of $\|.\|_p$ is the norm $\|.\|_q$ with $1/p + 1/q = 1$. We use $\|A\|_p = (\sum_{i=1}^n \sum_{j=1}^m |A_{ij}|^p)^{\frac{1}{p}})$ to denote the generalized p-norm of an $n \times m$ matrix A. For a matrix A we use A_{*j} to denote its jth column and A_{i*} to denote its ith row.

DEFINITION 1 (WELL-CONDITIONED BASIS). *[19] Let A be an $n \times m$ matrix of rank d, let $p \in [1, \infty)$, and let $\|.\|_q$ be the dual norm of $\|.\|_p$, i.e. $1/p+1/q = 1$. Then an $n \times d$ matrix U is an (α, β, p)-well-conditioned basis for the column space of A, if the columns of U span the column space of A, and (1) $\|U\|_p \leq \alpha$ and (2) for all $z \in \mathbb{R}^d$, $\|z\|_q \leq \beta\|Uz\|_p$. We say that U is a p-well-conditioned basis for the column space of A, if α and β are $d^{O(1)}$, independent of m and n.*

Given an $n \times m$ matrix A, one can compute a well-conditioned basis efficiently:

THEOREM 2. *[19] Let A be an $n \times m$ matrix of rank d, let $p \in [1, \infty)$, and let $\|.\|_q$ be the dual norm of $\|.\|_p$, i.e., $1/p+1/q = 1$. Then there exists an (α, β, p)-well-conditioned basis U for the column space of A such that if $p < 2$, then $\alpha = d^{\frac{1}{p}+\frac{1}{2}}$ and $\beta = 1$; if $p = 2$, then $\alpha = d^{\frac{1}{2}}$ and $\beta = 1$; and if $p > 2$, then $\alpha = d^{\frac{1}{p}+\frac{1}{2}}$ and $\beta = d^{\frac{1}{q}-\frac{1}{2}}$. Moreover, U can be computed in $O(nmd+nd^5 \log n)$ time (or in just $O(nmd)$ time if $p = 2$).*

In fact, for our linear oblivious subspace embedding, we will only need the existence of a well-conditioned basis, rather than an efficient algorithm for finding one, and in this case for $p = 1$ the Auerbach bases give better parameters. If $U_1, \ldots, U_d$ is an Auerbach basis for a d-dimensional space, then $\|U_j\|_1 = 1$ for all $j \in [d]$, and whenever $\|\sum_{j=1}^d U_j \cdot \nu_j\|_1 \leq 1$, then $\|\nu\|_\infty \leq 1$. The existence of such a basis for ℓ_1 (in fact, a similar statement holds for every finite dimensional normed space) was proved by Auerbach [6] (also see [20, 48]). As noted in [19], it follows from the definition that such a basis is a $(d, 1, 1)$-well-conditioned basis.

THEOREM 3. *(see "Connection to Auerbach bases" in Section 3.1 of [19]) Let A be an $n \times m$ matrix of rank d. Then there exists a $(d, 1, 1)$-well-conditioned bassis U for the column space of A.*

An embedding of a finite metric space (P, D) into another finite metric space (P', D') is a mapping $f : P \to P'$ that approximately preserves distances. An embedding has contraction c_f if for all $x, y \in P$ we have $D(x, y) \leq c_f \cdot D'(f(x), f(y))$ and expansion e_f if $e_f \cdot D(x, y) \geq D'(f(x), f(y))$. The distortion of an embedding is the product of its expansion and its contraction.

We need the following tail inequality for heavy-tailed random variables.

THEOREM 4. *(Lemma 1 of [7]) Let $\Psi_1, \ldots, \Psi_s$ be independent, non-negative random variables with probability density functions $f_{\Psi_1}, \ldots, f_{\Psi_s}$ such that $\max_{1 \leq i \leq s} \|f_{\Psi_i}\|_\infty = B < \infty$. Then for every $t > 0$ and $u > 0$,*

$$\mathbf{Pr}[\sum_{i=1}^s \Psi_i^u < t] \leq \frac{(B'B)^s t^{s/u}}{u^{\frac{s-1}{2}} s^{s/u+1/2}},$$

where $B' > 0$ is an absolute constant.

In our streaming applications we assume access to a truly random string of length $n \cdot \text{poly}(m/\varepsilon)$. This assumption can be removed via a technique introduced by Indyk [27]. We show in the full version of the paper how to do this by incurring an additional $O(\log n)$ factor in the space, and a multiplicative $O(S \log n)$ factor in the update time, where S is the space of the algorithm with true randomness. To achieve our fastest time complexities in an offline setting, one should instead use true randomness.

3. LINEAR OBLIVIOUS EMBEDDINGS OF SUBSPACES OF L1

Let A be an $n \times d$ matrix with full rank. We show how to embed the column space of A into low-dimensional ℓ_1.

THEOREM 5 (LINEAR OBLIVIOUS EMBEDDINGS). *Let $L = \{y \in \mathbb{R}^n : x \in \mathbb{R}^d, Ax = y\}$ be an arbitrary d-dimensional linear subspace of $\mathbb{R}^n$, i.e., L is the column space of an $n \times d$ matrix A of rank d. Then there is an $r_0 = r_0(d) = O(d \log d)$ and a sufficiently large constant $C_0 > 0$, such that for any r with $r_0 \leq r \leq d^{O(1)}$, and any constant $C \geq C_0$, if R is an $r \times n$ matrix whose entries are choosen i.i.d. from the Cauchy distribution and are scaled by C/r, then with probability at least $99/100$ for every $y \in L$:*

$$\|y\|_1 \leq \|Ry\|_1 \leq O(d \log d)\|y\|_1 \ ,$$

i.e., the function $f : \mathbb{R}^n \to \mathbb{R}^r$, $f(y) := Ry$, is a linear oblivious embedding for d-dimensional subspaces of ℓ_1^n into $\ell_1^{O(d \log d)}$ with distortion $O(d \log d)$. The constant in the O-notation depends on the choice of C and the exponent in the $d^{O(1)}$ term.

PROOF. The probability density function (p.d.f.) of a standard Cauchy distribution is $\phi(x) = \frac{1}{\pi} \cdot \frac{1}{1+x^2}$. It is 1-stable, meaning that if $X_1, \ldots X_n$ are i.i.d. Cauchy, then $\sum a_i X_i$ is distributed as $(\sum |a_i|) \cdot X$ for a Cauchy X. Let R be an $r \times n$-matrix whose entries are chosen from the Cauchy distribution and scaled by a factor C/r. Thus, the result of the embedding $f : \mathbb{R}^n \to \mathbb{R}^r$, $f(y) := Ry$, applied to a

point $y \in L$, is distributed in each coordinate as a standard Cauchy scaled by $C\|y\|_1/r$ and is therefore not concentrated because a Cauchy random variable has unbounded mean. We analyze the probability that $\|Ry\|_1 \approx \|y\|_1$, which will happen if the sum of i.i.d. half-Cauchy[1] random variables falls in a certain range. To determine this range, we use Theorem 4 to analyze the lower tail, while we use a Markov bound to analyze the upper tail. Intuitively, the lower tail of such a sum behaves like a sum of binomially distributed random variables, while the upper tail tends to be distributed as a Cauchy random variable.

LEMMA 6 (NO CONTRACTION). *There is a constant $C_1 > 0$ such that for any $r \geq 1$ and any constant $C \geq C_1$, if R is an $r \times n$ matrix whose entries are i.i.d. standard Cauchy random variables scaled by C/r then, for any fixed $y \in \mathbb{R}^n$,*

$$\mathbf{Pr}[\|Ry\|_1 < \|y\|_1] \leq \frac{1}{2^r}$$

.

PROOF. We can assume, by linearity, that $\|y\|_1 = 1$. By 1-stability of the Cauchy distribution, we have that $R_{j*}y$ is distributed as a Cauchy variable scaled by $\frac{C}{r} \cdot \|y\|_1$ and so $\frac{r}{C} \cdot \frac{1}{\|y\|_1} \cdot \|Ry\|_1 = \frac{r}{C} \cdot \|Ry\|_1$ is the sum of r i.i.d. half-Cauchy random variables. As the density function of a half-Cauchy is $\frac{2}{\pi}\frac{1}{1+x^2}$, it satisfies the condition of Theorem 4 with $B = O(1)$. Applying Theorem 4 with $s = r$, $u = 1$ and $t = \frac{r}{C} \cdot \frac{1}{\|y\|_1} = \frac{r}{C}$,

$$\begin{aligned}\mathbf{Pr}[\|Ry\|_1 < \|y\|_1] &= \mathbf{Pr}[\frac{r}{C} \cdot \|Ry\|_1 < \frac{r}{C} \cdot \|y\|_1] \\ &\leq \frac{(B'')^r \left(\frac{r}{C}\right)^r}{r^{r+1/2}} \\ &\leq \frac{1}{2^r},\end{aligned}$$

for a constant B''. Here the last inequality follows for $C \geq C_1 := 2B''$ a sufficiently large constant. The lemma follows. □

We would like to apply this lemma to a net on a sphere inside the subspace L. If the net is sufficiently fine, we not only approximate the vectors of the net, but any vector in the subspace. This approach works for the lower bound, but has no guarantees for the upper bound. Since the upper tail of a sum of half-Cauchy's is heavy-tailed, there is indeed no analogue for the upper bound. We instead condition on the average dilation of a basis vector in a fixed well-conditioned basis of the space being small. This single event turns out to be sufficient to claim small dilation for all vectors in L. We do not know how to upper bound the distortion of our embedding without using the existence of a well-conditioned basis, which we have not seen used to obtain embedding results before.

The statement of Theorem 5 does not depend on the representation of $L = \{y \in \mathbb{R}^n : Ax = y\}$. Therefore, in the remainder of the proof we will assume that A is a well-conditioned basis that satisfies the guarantees of Theorem 3, that is, the column space of A is a $(d, 1, 1)$-well-conditioned basis for L.

[1]We use half-Cauchy to refer to the distribution of $|X|$, where X is a standard Cauchy.

LEMMA 7 (FIXED SUM OF DILATIONS). *Let R be an $r \times n$ matrix whose entries are i.i.d. standard Cauchy random variables scaled by C/r for a constant C, $r \geq 1$. Then there is a constant $C_2 = C_2(C) > 0$ such that for any fixed set $\{y_1, \ldots, y_d\}$ of d vectors in L,*

$$\mathbf{Pr}[\sum_{i=1}^{d} \|Ry_i\|_1 \geq C_2 \log(rd) \cdot \sum_{i=1}^{d} \|y_i\|_1] \leq \frac{1}{1000}.$$

PROOF. By 1-stability of the Cauchy distribution we have that $R_{j*}y_i$ is distributed as a Cauchy variable scaled by $\frac{C}{r} \cdot \|y_i\|_1$ and so $\|Ry_i\|_1$ is the sum of r i.i.d. half-Cauchy's $Y_{i,1}, \ldots, Y_{i,r}$, each scaled by $\frac{C}{r} \cdot \|y_i\|_1$. Like Cauchy random variables, also half-Cauchy random variables do not have a finite expectation. Therefore, we will use truncated half-Cauchy's for our analysis. For a $B \geq 2$, define the truncated half-Cauchy $Z_{i,j}^B$ to equal $Y_{i,j}$ if $Y_{i,j} \leq B$, otherwise $Z_{i,j}^B = B$. We will choose the value of B in such a way that the probability that any $Y_{i,j}$ exceeds B is small. Since we have $O(rd)$ half-Cauchy's and by the p.d.f. of the Cauchy distribution, this is true for sufficiently large $B = O(rd)$. This way we can reduce our analysis to the analysis of truncated half-Cauchy's, which have a finite expectation. Namely, it is known that $\mathbf{E}[Z_{i,j}^B] = O(\log B)$ [27]. Put $Y_i = \sum_{j=1}^{r} Y_{i,j}$ and $Z_i^B = \sum_{j=1}^{r} Z_{i,j}^B$. For any constant $C_2 > 0$ we get

$$\begin{aligned}&\mathbf{Pr}[\sum_{i=1}^{d} \|Ry_i\|_1 \geq C_2 \cdot \log B \sum_{i=1}^{d} \|y_i\|_1] \\ \leq\ &\mathbf{Pr}[\exists i, j \ \mid \ Y_{i,j} > B] \\ +\ &\mathbf{Pr}[\frac{C}{r}\sum_{i=1}^{d} Y_i \cdot \|y_i\|_1 \geq C_2 \log B \sum_{i=1}^{d} \|y_i\|_1 \mid \forall i, j\ Y_{i,j} \leq B].\end{aligned}$$

For $B = O(rd)$ sufficiently large, we have

$$\mathbf{Pr}[\exists i, j \ \mid \ Y_{i,j} > B] \leq \frac{1}{2000}.$$

By Markov's inequality we then have,

$$\begin{aligned}&\mathbf{Pr}[\frac{C}{r}\sum_{i=1}^{d} Y_i \cdot \|y_i\|_1 \geq C_2 \log B \sum_{i=1}^{d} \|y_i\|_1 \mid \forall i, j\ Y_{i,j} \leq B] \\ \leq\ &\frac{\mathbf{E}[\sum_{i=1}^{d} Y_i \cdot \frac{C}{r} \cdot \|y_i\|_1 \mid \forall i, j\ Y_{i,j} \leq B]}{C_2 \log B \sum_{i=1}^{d} \|y_i\|_1} \\ =\ &\frac{\mathbf{E}[\sum_{i=1}^{d} Z_i^B \cdot \frac{C}{r} \cdot \|y_i\|_1 \mid \forall i, j\ Y_{i,j} \leq B]}{C_2 \log B \sum_{i=1}^{d} \|y_i\|_1} \\ \leq\ &\frac{\mathbf{E}[\sum_{i=1}^{d} Z_i^B \cdot \frac{C}{r} \cdot \|y_i\|_1]}{C_2 \log B \sum_{i=1}^{d} \|y_i\|_1}\end{aligned}$$

where the equality follows from the definition of Z_i^B and the constraint $Y_{i,j} \leq B$ for all i, j and the last inequality follows from $\mathbf{E}[Z_i^B \mid \forall i, j\ Y_{i,j} \leq B] \leq \mathbf{E}[Z_i^B]$. Now we use $\mathbf{E}[Z_{i,j}^B] = O(\log B)$ and linearity of expectation to get

$$\begin{aligned}&\frac{\mathbf{E}[\sum_{i=1}^{d} Z_i^B \cdot \frac{C}{r} \cdot \|y_i\|_1]}{C_2 \log B \sum_{i=1}^{d} \|y_i\|_1} \\ \leq\ &\frac{C}{r \cdot C_2} \cdot \frac{O(r \log B) \cdot \sum_{i=1}^{d} \|y_i\|_1}{\log B \cdot \sum_{i=1}^{d} \|y_i\|_1} \\ \leq\ &\frac{1}{2000},\end{aligned}$$

where the last inequality follows for $C_2 = C_2(C)$ being large enough. This finishes the proof. $\square$

We now define $C_0 := 2 \cdot C_1$ and assume that $C \geq C_0$. We condition on the event

$$\sum_{i=1}^{d} \|RA_{*i}\|_1 < C_2 \log(rd) \cdot \sum_{i=1}^{d} \|A_{*i}\|_1$$

for the constant $C_2 = C_2(C)$ from Lemma 7. This event is guaranteed to hold with probability 999/1000 by Lemma 7 applied to the columns $A_{*1}, \ldots, A_{*d}$ of the well-conditioned basis A. We show below that this suffices for the upper bound on the expansion of the embedding $f(y) = Ry$. Let y be an arbitrary point from L. Using that $y = Ax$ for some $x \in \mathbb{R}^d$ we get

$$\begin{aligned} \|Ry\|_1 &= \|RAx\|_1 \\ &\leq \sum_{1 \leq j \leq d} \|RA_{*j}x_j\|_1 \\ &= \sum_{1 \leq j \leq d} |x_j| \cdot \|RA_{*j}\|_1 \\ &\leq \|x\|_\infty C_2 \log(rd) \sum_{1 \leq j \leq d} \|A_{*j}\|_1, \end{aligned}$$

where the first inequality follows from the triangle inequality and the last inequality follows with probability 999/1000 by Lemma 7. We continue by using the properties of well-conditioned bases, i.e. $\sum_{1 \leq j \leq d} \|A_{*j}\|_1 = \|A\|_1 \leq d$ and $\|x\|_\infty \leq \|Ax\|_1$, as well as $y = Ax$ to obtain

$$\|x\|_\infty C_2 \log(rd) \sum_{1 \leq j \leq d} \|A_{*j}\|_1 \leq C_2 \cdot d \log(rd) \|y\|_1 \qquad (1)$$

which is $O(d \log d) \cdot \|y\|_1$ provided that $r = d^{O(1)}$. To complete the upper bound, we now specify r_0, and to do so, fill in the details of the lower bound.

Definition 8. *Let A be an $n \times d$ matrix of full rank and $L = \{y \in \mathbb{R}^n : y = Ax,\ x \in \mathbb{R}^d\}$ be its column space. A γ-net of the ℓ_1-unit sphere $\mathcal{S}^{d-1}$ in the subspace L, $\mathcal{S}^{d-1} = \{y \in L \mid \|y\|_1 = 1\}$, is a set $G \subset \mathcal{S}^{d-1}$ of points so that for every $y \in \mathcal{S}^{d-1}$, there is a point $g \in G$ for which $\|y - g\|_1 \leq \gamma$.*

A standard result (see, e.g., the ball B on page 2068 of [19]) is that for any $\gamma \in (0,1)$, a γ-net G of size at most $(3/\gamma)^d$ exists. Let G be a γ-net for $\gamma = \frac{1}{C_2 d \log(rd)}$ and define $r_0 := \inf\{r | r \geq \log(1000) + \log((\frac{3}{\gamma})^d)\} + 1 = O(d \log d)$. For $r \geq r_0$ and $C \geq C_1$ we obtain by Lemma 6 for any fixed $y \in \mathbb{R}^n$

$$\mathbf{Pr}[\|Ry\|_1 < \|y\|_1] \leq \frac{1}{1000} \cdot \frac{1}{(3/\gamma)^d}.$$

By the union bound we obtain with probability at least 999/1000 for every $g \in G$, $\|Rg\|_1 \geq \|g\|_1$. Further, since C is the scaling factor of our embedding, we obtain for $C \geq C_0 = 2 \cdot C_1$ that with probability at least 999/1000 for every $g \in G$, $\|Rg\|_1 \geq 2 \cdot \|g\|_1$. Now consider an arbitrary vector $y \in L$ with $\|y\|_1 = 1$. Then we can write $y = g + z$, where $g \in G$ is the grid point closest to y. By our choice of γ, we have $\|z\|_1 \leq \frac{1}{C_2 d \log(rd)}$. Since $y, g \in L$, so is $z = y - g$, and so by Equation 1 we obtain $\|Rz\|_1 \leq 1$. By our discussion above, we also have $\|Rg\|_1 \geq 2 \cdot \|g\|_1 = 2$. Now, the triangle inequality implies $\|Ry\|_1 \geq \|g\|_1 - \|z\|_1 \geq 1$. Since an arbitrary vector $y' \in L$ can be written as $\ell \cdot y$ for a unit vector $y \in L$, by the linearity, we have $\|Ry'\|_1 = \|R(\ell \cdot y)\|_1 = \ell \cdot \|Ry\|_1 \geq \ell$. We conditioned on two events that each occur with probability 999/1000. Thus the theorem follows for our choices of C_0 and r_0 by the union bound.

4. FAST AND SIMPLE COMPUTATION OF WELL-CONDITIONED BASES

In this section we give a simple algorithm for computing well-conditioned bases that avoids the computation of a Löwner-John ellipsoid and linear programming techniques of previous approaches [16, 19]. Our algorithm is also much faster than the algorithms of previous approaches. Let A be an $n \times m$ matrix of rank d.

As mentioned in Section 1, the algorithm consists of three steps: (1) compute RA, where R is the linear oblivious subspace embedding, (2) compute an $m \times d$ matrix X so that RAX is orthonormal, and (3) output AX.

Theorem 9 (Well-Conditioned Basis Algorithm). *Let A be an $n \times m$ matrix of rank d, and let R be an $r \times n$ matrix, $r \geq r_0$, satisfying the conditions guaranteed with probability at least 99/100 by Theorem 5. Let $U' = RAX$ be an $(\alpha, \beta, 1)$-well-conditioned basis for the column space of RA. Then $U = AX$ is an $(\alpha, O(d \log d) \cdot \beta, 1)$-well-conditioned basis for the column space of A.*

Proof. Let A be an arbitrary $n \times m$ matrix of rank d and let R be a matrix satisfying the precondition of the theorem. Let $L = \{y \in \mathbb{R}^n : y = Ax, x \in \mathbb{R}^m\}$ be the column space of A. Note that L has dimension d and that Theorem 5 applies since it is independent of the representation of L. Let $U' = RAX$ be an $(\alpha, \beta, 1)$-well conditioned basis of RA and let $U = AX$. We have $\|x\|_\infty \leq \beta \|U'x\|_1$ by the definition of a well-conditioned basis. Hence, $\|x\|_\infty \leq \beta \cdot \|U'x\|_1 = \beta \cdot \|RAXx\|_1 = O(\beta \cdot d \log d) \cdot \|AXx\|_1$, where the last equality follows from Theorem 5 since AXx is a vector in L. It remains to show that $\|AX\|_1$ is small. Observe that $\|U'\|_1 = \|RAX\|_1 \leq \alpha$ by the definition of a well-conditioned basis. We have

$$\begin{aligned} \alpha \geq \|U'\|_1 &= \|RAX\|_1 = \sum_{1 \leq j \leq d} \|RAX_{*j}\|_1 \\ &\geq \sum_{1 \leq j \leq d} \|AX_{*j}\|_1 = \|AX\|_1 \ , \end{aligned}$$

where the last inequality follows from Theorem 5 since AX_{*j} is in L. Notice that U is indeed a basis for the column space of A, since it is contained in the column space of A and by the property above, preserves all lengths of vectors in the column space of A up to a relative error. Hence, U is an $(\alpha, O(d \log d)\beta, 1)$-well-conditioned basis. $\square$

Theorem 10 (Algorithm Efficiency). *Let A be an $n \times m$ matrix of rank d. Then there is an algorithm that computes in time $(nd^{\omega-1} + nm + d^3) \cdot \text{polylog}(m)$ with probability 49/50 an $(\alpha, \beta, 1)$-well-conditoned matrix AX of the column space of A with $\alpha = O(d^{3/2} \log^{1/2} d)$ and $\beta = O(d \log d)$, where $\omega < 2.376$ is the matrix multiplication exponent. The algorithm also outputs X in the same amount of time.*

Proof. Theorem 5 asserts that with probability 99/100 the preconditions of Theorem 9 are satisfied, provided we

choose $d^{O(1)} \geq r \geq r_0 = O(d \log d)$. Now, our algorithm computes $U' = RAX$ which is an orthonormal basis. It follows that each column U'_{*j} of U' satisfies $\|U'_{*,j}\|_2 = 1$, and since U'_{*j} has $O(d \log d)$ coordinates, $\|U'_{*j}\|_1 = O(\sqrt{d \log d})$. It follows that $\|U'\|_1 = O(d^{3/2} \log^{1/2} d)$. Moreover,

$$\|x\|_\infty \leq \|x\|_2 = \|U'x\|_2 \leq \|U'x\|_1,$$

and so U' is an $(O(d^{3/2} \log^{1/2} d), 1, 1)$-well-conditioned basis for the column space of RA. It follows by Theorem 9 that AX is an $(O(d^{3/2} \log^{1/2} d, O(d \log d), 1)$-well-conditioned basis for the column space of A.

For the time taken, we show how with probability 99/100, in $(nd^{\omega-1} + nm + d^3) \cdot \text{polylog}(m)$ time we can replace A with an $n \times d \cdot \text{polylog}(m)$ matrix A' with the same column space. Given this, the overall time complexity of $(nd^{\omega-1} + nm + d^3) \cdot \text{polylog}(m)$ follows, since, we can in $d^3 \cdot \text{polylog}(m)$ time compute an orthornormal basis $RA'X$ of the $d \log d \times d \cdot \text{polylog}(m)$ matrix RA', as well as the matrix X, via a QR-decomposition, and we can perform the matrix multiplications RA' and $A'X$ in time $nd^{\omega-1}\text{polylog}(m)$ via fast matrix (block) multiplication.

To quickly find the rank of A, we use a result of Ailon and Liberty [3] with a binary search, though other methods are also possible.

THEOREM 11. *([3]) Suppose we choose a $k \times m$ matrix Φ, drawing each row uniformly at random from the unnormalized $m \times m$ Hadamard matrix, and then we choose an $m \times m$ diagonal matrix D with each diagonal element uniformly chosen from $\{-1, 1\}$. Then for any set G of N vectors, with probability at least 3/4, if $k = \Theta(\log N \log^4 m)$ then for every $x \in G$, $\|\Phi Dx\|_2 = \sqrt{k}(1 \pm 1/3)\|x\|_2$. For any $x \in \mathbb{R}^m$, ΦDx can be computed in $O(m \log m)$ time.*

Let $\gamma = \frac{1}{2\sqrt{m}}$. We apply Theorem 11 to a γ-net G of the space of $\{x^T A \mid \|x^T A\|_2 = 1\}$. Since the column space of A has rank d, the row space of A has rank d. So the number of points in the γ-net is $\leq \left(\frac{3}{\gamma}\right)^d$. Suppose we apply Theorem 11 with $k = \Theta(d \log^5 m)$. For $y \in \{x^T A \mid \|x^T A\|_2 = 1\}$, we write $y = g + z$, where $g \in G$ and $\|z\|_2 \leq \gamma$. Since Φ is a sign matrix, for any row Φ_{i*}, $\Phi_{i*} Dz \leq \|z\|_1$. Hence, using a basic relationship between the ℓ_1 and ℓ_2 norms,

$$\|\frac{1}{\sqrt{k}} \Phi Dz\|_2 \leq \|z\|_1 \leq \sqrt{m} \cdot \gamma \leq \frac{1}{2}.$$

By the triangle inequality and now assuming that the guarantees of the mapping of Theorem 11 hold for all $g \in G$,

$$\|\frac{1}{\sqrt{k}} \Phi Dy\|_2 \geq \|\frac{1}{\sqrt{k}} \Phi Dg\|_2 - \|\frac{1}{\sqrt{k}} \Phi Dz\|_2 \geq 1 - \frac{1}{3} - \frac{1}{2} > 0.$$

Hence, with probability at least 3/4, $\text{rank}(AD^T \Phi^T) = d$.

Now that $AD^T \Phi^T$ is an $n \times d \cdot \text{polylog}(m)$ matrix, we can use the following lemma.

LEMMA 12. *(special case of Lemma 3,4 of [18]) If S is an $O(d) \times n$ sign matrix and B is an $n \times d$ matrix with orthonormal columns, then with probability at least 3/4, for all x with $\|x\|_2 = 1$, it holds that $\|SBx\|_2 > 0$.*

Letting B be a set of orthonormal columns in the column space of $AD^T \Phi^T$ of maximal rank, and applying Lemma 12, it follows that for a random $O(d) \times n$ sign matrix S, with probability at least 3/4, $\text{rank}(SB) = \text{rank}(B)$. By a union bound, with probability at least 1/2, $\text{rank}(SAD^T \Phi^T) = d$. By repeating the procedure $O(\log \log d)$ times and taking the maximum rank found, with probability $1 - 1/(100 \log d)$ there will be a repetition for which $\text{rank}(SAD^T \Phi^T) = d$.

Our algorithm linearly searches for an index i for which $2^{i+1} > \text{rank}(A) \geq 2^i$, setting d in the above procedure to be 2^{i+1}. In the procedure we first compute $AD^T \Phi^T$ in $O(nm \log m)$ time. We then compute $SAD^T \Phi^T$ in $2^{i(\omega-1)} \cdot n \cdot \text{polylog}(m)$ time. We then compute $\text{rank}(SAD^T \Phi^T)$ in $2^{3i}\text{polylog}(m)$ time by Gaussian elimination.

Since the column space of $AD^T \Phi^T$ is contained in that of A, as soon as $\text{rank}(AD^T \Phi^T) = d$, we can set $A' = AD^T \Phi^T$. As $2^i = O(\text{rank}(A))$, the overall time complexity follows. The probability of error is 99/100, by a union bound over the at most $\log d$ guesses of i. □

5. L1-REGRESSION

Regression analysis is a basic technique from statistics used to analyze linear dependencies between data points. In the algebraic version of ℓ_p-regression we are given an $n \times m$ matrix A and an n-dimensional vector b and we would like to solve

$$\min_{x \in \mathbb{R}^m} \|Ax - b\|_p.$$

The rows of A can be viewed as measurement points and the corresponding entries in b as the measured value. We will consider the regression problem for $p = 1$. This problem is called ℓ_1-regression or least absolute deviation regression. Geometrically, the ℓ_1-regression problem asks to find a hyperplane in $\mathbb{R}^{m+1}$ such that the sum of absolute differences between the b_i and the values in the last coordinate of the hyperplane at point A_{i*} are minimized.

The turnstile streaming model we consider here assumes that the number n of input points in the stream is known (it will suffice that n is an upper bound on the number of points in the stream). The input stream consists of updates to an $n \times m$ matrix A and an n-dimensional vector b. Initially, all entries in A and b are 0. An update specifies the entry in A or b that is updated and the amount c by which the entry is changed. For example, an update to position (i, j) in matrix A by a value c will add c to $A_{i,j}$. Note that c may also be negative. At the end of the stream (or any point in time) we would like to approximately solve the current regression problem $\min_{x \in \mathbb{R}^m} \|Ax - b\|_1$.

One can also interpret the regression problem as a problem in the n-dimensional space $\mathbb{R}^n$. Namely, Ax is a linear combination of the columns of A, so every point that can be written as such a linear combination lies in a d-dimensional linear subspace of $\mathbb{R}^n$, where d is the rank of A. Since b is also a vector in $\mathbb{R}^n$, the regression problem can be viewed as finding the closest point $v = Ax$, in a linear subspace spanned by the columns of A, to a target point b, where the distance is ℓ_1-distance. The vector x is the solution to the regression problem. The cost of a solution x is $\|Ax - b\|_1$, which is the ℓ_1-norm of a vector in the space spanned by the columns of A and the vector b.

We use results from the previous sections to improve an algorithm for ℓ_1-regression by Clarkson [16]. In order to solve the ℓ_1-regression problem, the algorithm of Clarkson computes a well-conditioned basis A' of the column space of A as well as a constant factor approximation x_C. Then it samples the points (rows) of A' with probability proportional to the ℓ_1-norm of the rows plus the corresponding entry in

$\eta b'$, where $b' = Ax_C - b$ is the residual from the constant factor approximation scaled by η such that $\|\eta b'\|_1 = d$. It then solves the regression problem on the sample using linear programming. The result will be, with high probability, a $(1+\epsilon)$-approximation. The time complexity of his algorithm is $O(nm^5 \log n + \text{poly}(m\varepsilon^{-1}))$, which is an offline algorithm with linear space.

Our main theorem for ℓ_1-regression is the following. We use $\text{L1R}(k, d)$ to denote the time to solve an instance of ℓ_1-regression instance on a $k \times d$ matrix with a k-dimensional column vector. To achieve the dependence on d instead of m in the offline algorithm, we use Theorem 10.

THEOREM 13. *There is a $(1+\epsilon)$-approximation for the ℓ_1-regression problem with running time*

$$O((nd^{\omega-1+\beta} + nm + L1R(d^{7/2}/\varepsilon^2 \log(d/\epsilon), d)) \cdot \text{polylog}(m)),$$

were ω is the exponent of matrix multiplication, and $\beta > 0$ is an arbitrarily small constant.

The algorithm can be implemented as a streaming algorithm in the turnstile model. The streaming implementation uses $\text{poly}(m\varepsilon^{-1} \log n)$ space and has $\text{poly}(m\varepsilon^{-1} \log n)$ update time. The algorithm maintains a summary such that with probability at least 3/4, a $(1+\varepsilon)$-approximation can be output. The time to extract a solution from the summary maintained by the algoritm is $\text{poly}(m\varepsilon^{-1} \log n)$.

5.1 Sketch of the proof

We will first describe and analyze the offline algorithm. Then we will sketch how to implement the algorithm in the streaming setting. A detailed proof will appear in the full version of the paper. As mentioned, our algorithm is a more efficient implementation of an algorithm of Clarkson [16]. Our main novelties over Clarkson's algorithm are the ability to compute the well-conditioned basis in the sketch space using our subspace-preserving embedding for ℓ_1, and to obtain an approximate solution from the sketch space.

The main bottleneck in Clarkson's algorithm is the computation of a well-conditioned basis. We bypass this bottleneck using the algorithm from Theorem 10. We first replace A with a well-conditioned basis A' using Theorem 10. We further require a constant-factor approximation to define the distribution of Clarkson's algorithm. For this, it is not enough to apply Theorem 5 on the column space of A' adjunct b. This results in an $O(d \log d)$-approximation, whereas we need an $O(1)$-approximation to achieve our fastest running time. We instead use a lopsided embedding of Indyk.

THEOREM 14. *(Theorem 5 of [27]) For any $1 > \varepsilon > \gamma > 0$ and $\delta > 0$, there is a probability space over linear mappings $f : \ell_1^m \to \ell_1^k$, where $k = (\ln(1/\delta))^{1/(\varepsilon-\gamma)}/c(\gamma, 1-\varepsilon)$, for a function $c(\gamma, 1-\varepsilon) > 0$ depending only on γ and ε, such that for any pair of points $p, q \in \ell_1^m$,*

- $\mathbf{Pr}[\|f(p) - f(q)\|_1 < (1-\varepsilon)\|p-q\|_1] \leq \delta$, *and*
- $\mathbf{Pr}[\|f(p) - f(q)\|_1 > (1+\varepsilon)\|p-q\|_1] \leq \frac{1+\gamma}{1+\varepsilon}$.

REMARK 15. The only difference between this and Indyk's theorem is that we allow $\varepsilon < 1$, whereas [27] requires $\varepsilon \leq 1/2$. This difference allows us to set k to be much closer to $\ln(1/\delta)$ by setting ε very close to 1 and γ very close to 0. Inspecting the changes required of Indyk's proof, we note that the only difference is on p.11 of [27], where now the value α satisfies $0 < \alpha < 1$ instead of $1/2 \leq \alpha < 1$, but this does not affect the remainder of the proof.

The mapping in Theorem 14 is a matrix of i.i.d. Cauchy random variables scaled by C/k for a constant C. Let $\zeta > 0$ be an arbitrarily small positive constant. We apply Theorem 14 with $\varepsilon = 1-\zeta$ and $\gamma = \zeta$, while $\delta = \Theta(1/(d \log d))^{O(d+1)}$. Let the column space of A' adjunct b be denoted L. By placing a net on L we can use the first condition of Theorem 14 to argue as in the last paragraph of the proof of Theorem 5 that with probability at least 99/100, $\|f(y)\|_1 \geq \zeta\|y\|_1$ for all $y \in L$. It follows that if also $\|f(Ax_{opt} - b)\|_1 \leq (2-\zeta)\|Ax_{opt} - b\|_1$, where x_{opt} is the optimal solution, which occurs with probability at least $\frac{1}{3}$ by the second condition of Theorem 14, then by solving the regression problem in the sketch space using linear programming with matrix $f(A')$ and vector $f(b')$, we obtain an $O(1)$-approximation with probability at least $\frac{1}{4}$. By repeating this procedure $O(1)$ times and taking the vector found across the repetitions resulting in the minimal cost, we obtain an $O(1)$-approximation with probability at least, say, 99/100.

The time complexity is, up to a constant factor, at most $(nd^{\omega-1} + nm + d^3) \cdot \text{polylog}(m)$ to compute A', then $nd^{\omega-1+\beta}$ to sketch A', and then $\text{L1R}(O(d^{1+\beta}), d)$ to solve the smaller regression problems, for $\beta > 0$ an arbitrarily small constant depending on ζ. Let us call the approximate solution x_C. If the approximate solution has cost 0, we return it. Otherwise, we continue by performing Clarkson's weighted sampling and solving the regression problem on the sample. The pseudocode of the algorithm is given below.

FASTREGRESSION(A, b, s)
1. Compute a well-conditioned basis A' of the column space of A and X, $A' = AX$, using the algorithm of Theorem 10
2. Compute the $O(1)$-approxiation x_C as described above, using the matrix A'
3. Let R be an $r_0 \times n$ matrix of i.i.d. Cauchy random variables
4. Compute $A^* = RA'$ and $b^* = Rb$
5. If $\|A^* x_c - b^*\|_1 = 0$ then return Xx_c
6. Compute the residual vector $b' = A'x_C - b$ and scale it by a factor η such that $\eta \cdot \|b'\|_1 = d^{3/2} \log^{1/2} d$ Define $b'' = \eta \cdot \|b'\|_1$
7. Sample a set of s rows from A' adjunct b'' such that the probability to sample row i is $p_i = \min\{1, s \cdot (\|A'_{i*}\|_1 + |b''_i|)/(d^{3/2} \log^{1/2} d)\}$
8. Assign a weight $w_i = 1/p_i$ to each sample row
9. Solve the weighted problem to obtain solution x^*
10. Return Xx^*/η

We now briefly analyze the algorithm and show a value of s for which it is correct. The idea is to prove that for a sufficient discretization of the solution space every solution is approximated by our random sample. From this, it follows that the solution returned is a $(1+\epsilon)$-approximation. The analysis is similar to that of Clarkson [16].

Let A' be an $(\alpha, \beta, 1)$-well conditioned basis with $\alpha = O(d^{3/2} \log^{1/2} d)$ and $\beta = O(d \log d)$ being the values guaranteed by Theorem 10. We will first argue that any solution x to $\|A'x - b''\|_1$ that achieves an approximation ratio better than 2 satisfies $\|x\|_\infty \leq 3 \cdot \beta \cdot \|b''\|_1$. This can be seen as follows. By the definition of a well-conditioned basis, $\|x\|_\infty \leq \beta\|Ax\|_1$ and so $\|x\|_\infty > 3 \cdot \beta \cdot \|b''\|_1$ implies

that $\|A'x - b''\|_1 \geq \|A'x\|_1 - \|b''\|_1 > 2 \cdot \|b''\|_1$. Further the solution $x = 0$ has cost $\|b''\|_1$. Hence, x is not a 2-approximation.

The cost of an optimal solution is $\Theta(d^{3/2}\log^{1/2} d)$ since the solution x_C is an $O(1)$-approximation (and due to the scaling of b').

Our next step is to prove the following lemma, which, together with an appropriate discretization of the solution space, proves the approximation guarantee of the algorithm.

LEMMA 16. *Let Z be a diagonal matrix with $Z_{ii} = 1/p_i$ with probability p_i and 0, otherwise. Let $x_f \in \mathbb{R}^d$ be fixed with $\|x_f\|_\infty = O(d^{5/2}\log^{3/2} d)$. There is an*

$$s_0 = O(d^{5/2}\log^{3/2} d \ln(1/\delta)/\epsilon^2),$$

such that for any $s \geq s_0$ we have

$$\mathbf{Pr}[\big|\|Z(A'x_f - b'')\|_1 - \|A'x_f - b''\|_1\big| > \epsilon\|A'x_f - b''\|_1] \leq \delta.$$

PROOF. We first observe that $\mathbf{E}[\|Z(A'x_f - b'')\|_1] = \|A'x_f - b''\|_1 = \Omega(d^{3/2}\log^{1/2} d)$. We can assume that $p_i < 1$ for every p_i since any $p_i = 1$ will only increase the expected value but does not contribute to the variance. Further, we have $\|A'_{i*}x_f - b''_i\|_1 \leq \|A'_{i*}x_f\|_1 + \|b''_i\|_1 \leq \|A'_{i*}\|_1 \cdot \|x_f\|_\infty + \|b''_i\|_1$ by Hölder's inequality. Hence, $\frac{\|A'_{i*}x_f - b''_i\|_1}{\|A'_{i*}\|_1 + \|b''_i\|} \leq \|x_f\|_\infty + 1$. This implies that $\frac{1}{p_i} \cdot \|A'_{i*}x_f\|_1 = O(d^4\log^2 d/s)$. We choose $s_0 = O(d^{5/2}\log^{3/2} d\ln(1/\delta)/\epsilon^2)$ sufficiently large. After appropriate scaling we can apply Chernoff bounds on independent random variables in the real unit interval $[0, 1]$ to obtain:

$$\mathbf{Pr}[\big|\|Z(A'x_f - b'')\|_1 - \|A'x_f - b''\|_1\big| > \epsilon\|A'x_f - b''\|_1] \leq \delta.$$

□

It remains to find a sufficiently dense discretization of the possible solutions. We observe that $\|A'x - b''\|_1$ changes by at most $\kappa \cdot \|A'\|_1 \leq \kappa\alpha$, if we change a coordinate of x by κ. If we discretize all coordinates in steps of κ, then changing to the nearest coordinate changes the cost of a solution to $\|A'x - b''\|_1$ by at most $\kappa \cdot d \cdot \alpha$. Further, $\|ZA'\|_1 = d^{O(1)}$ with probability at least 99/100 by Markov's bound. Hence, a solution to $\|Z(A'x - b'')\|$ with $\|x\|_\infty \leq d^{O(1)}$ also changes by at most $\kappa \cdot d^{O(1)}$. It follows that by setting $\kappa = \epsilon/d^{O(1)}$ we have a net of the solution space of all x, $\|x\|_\infty = O(d^{5/2}\log^{1.5} d)$ of size $(d/\epsilon)^{O(d)}$ and such that any solution in our solution space is approximated by the closest solution from the net. If we apply Lemma 16 with $\delta = (\epsilon/d)^{O(d)}$ to all solutions from the net, we know by the union bound and the previous discussion that each solution with $\|x\|_\infty = O(d^{5/2}\log^{1.5} d)$ is approximated by our sample upto a factor of $1 + O(\epsilon)$. Thus, the solution returned by our algorithm is a $(1 + \epsilon)$-approximation.

The running time of the algorithm is, as discussed above, $(nd^{\omega-1} + nm + d^3) \cdot \text{polylog}(m) + nd^{\omega-1+\beta} + \text{L1R}(O(d^{1+\beta}), d)$ for steps 1 and 2. Steps 3-8 can be implemented in $nd^{\omega-1}$ time. The remaining steps can be done in $\text{L1R}(O(d^{7/2}/\varepsilon^2 \cdot \log(d/\epsilon)), d)$ time.

The streaming algorithm.

We will now discuss, how the previous algorithm can be turned into a streaming algorithm. We observe that lines 1-5 of the algorithm can be implemented relatively easy in the streaming setting. The main difference is that in order to compute a well-conditioned basis, in Theorem 10 we do not perform the optimization using Theorem 11 to replace the number m of columns of A with d. Instead, we just use m (this is because we cannot do a binary search to find d in a single-pass). As a result, we get slightly different parameters for the well-conditioned basis and our space complexity depends more significantly on m (rather than d). Maintaining the product of a matrix R and the input matrix A and vector b is easy in a data stream since R is a linear map. The only issue here is that we cannot store the whole matrix R. Instead we use a standard approach introduced by Indyk [27] to generate the entries on the fly using Nisan's pseudorandom generator. Details can be found in the full version of the paper.

The main difficulty is to implement the sampling step in lines 6-8. In the streaming context it is not possible to sample using exactly the distribution of Clarkson. Fortunately, data structures with approximately the same guarantees were already designed by Andoni et al. [5] in an unrelated context of estimating the earthmover distance, and we can almost directly use them here. The only obstacle to apply their sampling algorithm is that Clarkson's distribution depends on the well-conditioned basis A' and the approximate solution x_C rather than the input matrix A. The main novelty of our streaming algorithm and its analysis is showing that noisy approximations to sampled rows as provided by [5] can be maintained and used instead of exact samples to solve a smaller regression instance in a single pass, providing a good approximation to the original regression problem.

Thus, we need to maintain a sketch that on input X and x_C returns a row of AX adjunct b'' distributed according to the distribution in Clarkson's algorithm. This sketch is independent of the sketch used for the computation of x_C and X. It receives X and x_C as input at the end of the stream. This sketch

1. only approximates the distribution in Clarkson's algorithm,
2. only guarantees that a noisy sampled row is returned,
3. works even if X and b'' are not known until the end of the stream.

For our space complexity, we also need later that we can round the entries of X in such a way that AX is still well-conditioned when X is of bounded precision. The proof of this is given in the full version of the paper.

LEMMA 17. *Suppose the entries of A are integer multiples of $(nm)^{-O(1)}$, and bounded in absolute value by $(nm)^{O(1)}$ and that $\alpha, \beta = (nm)^{O(1)}$. Suppose AX' is an $(\alpha, \beta, 1)$-well-conditioned basis with $\|AX'\|_1 \leq \alpha$ and $\|z\|_\infty \leq \beta\|AX'z\|_1$ for all $z \in \mathbb{R}^d$ and we round each of the entries of X' to the nearest integer multiple of $(nm)^{-O(1)}$, obtaining a matrix X. Then the matrix AX is a 1-well-conditioned basis of the columnspace of A with $\|AX\|_1 \leq \alpha + 1$ and $\|z\|_\infty \leq (\beta + 1)\|AXz\|_1$ for all $z \in \mathbb{R}^d$.*

In the streaming algorithm we will round the entries in matrix X to the nearest multiple of $(nm)^{-O(1)}$, satisfying the requirements of Lemma 17.

The sampling data structure.

During the processing of the stream we treat several quantities as formal random variables. Since our sketch is linear, it suffices to maintain the coefficients of these variables and plug in their values when they become available. For example, we do not know b', so we write $b' = A'x_C - b$ treating the vector x_C as a formal variable. Similarly, we do not know X, so we treat the entries of X as formal random variables.

Our algorithm requires a value W for which $\|A'x_c - b\|_1 \leq W \leq 2 \cdot \|A'x_c - b\|_1$. Since we do not know x_c until the end of the stream, it suffices to guess the value W, and $O(\log(nm))$ guesses suffice. In parallel we run the sketching algorithm of Theorem 14, which can naturally be implemented in as a streaming algorithm, to obtain an $O(1)$-approximation. These sketches are linear, so we can treat the entries of x_C as formal variables, and plug them in at the end of the stream, thereby obtaining W. Similarly, we obtain M' with $\|AX\|_1 \leq M' \leq 2 \cdot \|AX\|_1$. These sketches use $O(m^2 \log n)$ total bits of space (since we treat the entries of X as formal variables, since we do not know X until the end of the stream). Here we use the sketching algorithm of [31] which estimates the ℓ_1-norm of a vector (or matrix) up to an $O(1)$-factor in $O(\log n)$ bits of space.

Define $b'' = mb'/W$ and let B be the matrix AX adjunct b''. Furthermore, let $M = M' + m$ be an upper bound on $\|B\|_1$. Define thresholds $t_j := \eta \cdot M/2^j$, where η is chosen uniformly at random from $[1, 2]$. Define level j to be the set $I_j = \{i : \|B_{i*}\|_1 \in (t_j, 2 \cdot t_j]\}$.

We fix a level j and consider only the rows indexed in I_j. Since each row of B whose index is in I_j has roughly the same norm, it is sampled with roughly the same probability in the regression algorithm. We therefore try to sample from these rows roughly uniformly at random. The idea is to subsample elements from I_j and hash them to buckets such that, ideally, no elements from $\bigcup_{j' \leq j} I_{j'}$ collide. The hashing also creates noise based on rows from $\bigcup_{j' > j} I_{j'}$. However, the norm of the noise is typically small compared to the norm of elements of $\bigcup_{j' \leq j} I_{j'}$. So we can recognize all buckets containing rows from I_j, and return them.

The sampling data structures by Andoni et al. [5] are referred to as SAMPLER and EXTRACT. Algorithm SAMPLER hashes rows randomly into a small number of buckets. In each bucket the vector sum of rows hashed to the bucket is maintained. The vectors with large norm will be perfectly hashed. SAMPLER also sub-samples the rows in powers of 2, and maintains a hashing for each level of sub-sampling. The idea behind EXTRACT is to walk through the buckets in each sub-sampling phase, check if the norm in the bucket is in a given level (up to a small error for noise), and if so, return a weighted version of the row to be used in a smaller regression instance. Our data structures only differ in the following simple ways from the SAMPLER and EXTRACT algorithms used in [5] for other purposes: (1) we do not know the matrix AX until the end of the first pass, and (2) we need (at least) $(1 \pm \varepsilon)$-approximations to the norms of the sampled rows for regression (as opposed to just $O(1)$-approximations), so we need to use randomly shifted thresholds. This makes it unlikely that any of the rows we sample will be near the borders of the level it belongs to, and so it will be correctly classified. We note that the idea of using randomly shifted thresholds was also used in a precursor to the work of [5], namely in [29]. We defer these data structures to the full version of the paper.

The analysis of the streaming algorithm is similar to that of the offline algorithm. The main difference is that we also have to handle the noise incurred by our data structure, i.e., we do not recover rows of B (recall that B is the matrix AX adjunct b''), but rather there are other rows of B of small ℓ_1-norm which also collide with the sampled rows of B (the ℓ_1-norms of the colliding rows are smaller because in the level of sub-sampling for which the sampled row is obtained, it is one of the rows of largest ℓ_1-norm). The sampling data structure guarantees that with high probability, the noise of colliding rows for every sample row B_{i*} is of ℓ_1-norm at most $\zeta\|B_{i*}\|_1$ for a value $\zeta > 0$ specified below. Thus, the output of the sampler is a vector $B_{i*} + N_{i*}$, where N_{i*} is an arbitrary noise vector of ℓ_1-norm at most $\zeta\|B_{i*}\|_1$. From the analysis of the offline algorithm, we can restrict ourselves to solutions x with $\|x\|_1 = m^{O(1)}$ (recall that we use m in place of d for the streaming algorithm). Further, the cost of any such solution is well-approximated by the solution to the ℓ_1-regression problem on our set of sampled rows. Also, by the scaling of b' used to obtain b'', we have that an optimal solution has cost $\Omega(m)$. Hence, any solution has cost $\Omega(m)$, plus or minus the cost incurred by the noise. We further have that $w_i \cdot \|N_{i*}\|_1 = \zeta \cdot m^{O(1)}/s$ since $w_i = \max\{1, m^{O(1)}/(s \cdot \|B_{i*}\|_1)\}$ and $\|N_{i*}\|_1 \leq \zeta\|B_{i*}\|_1$. Hence, since $\|x\|_1 = m^{O(1)}$, we also have that $w_i \cdot \|N_{i*} \cdot x\|_1 = \zeta \cdot m^{O(1)}/s$. If we now sum up over all sampled rows and set $\zeta = \epsilon/m^{O(1)}$, then the overall effect of the noise is bounded by $O(\epsilon \cdot m)$ with high probability, and so any solution changes its cost by at most an additional factor of $1 + \epsilon$. Thus, it is sufficient to continue the analysis as in the case of the offline algorithm, for which there is no noise on the samples.

6. L1-NORM BEST FIT HYPERPLANE PROBLEM

Our ℓ_1-regression algorithm gives the first efficient solution for the ℓ_1-norm best fit hyperplane problem in the streaming model. Here one has n points in $\mathbb{R}^m$, and one wants a hyperplane that minimizes the sum of ℓ_1-distances of the input points to the hyperplane. Our algorithm for the ℓ_1-norm best fit hyperplane problem is the first 1-pass algorithm, works in the general turnstile model, and uses $\text{poly}(m\varepsilon^{-1} \log n)$ space for $(1+\varepsilon)$-approximation. The total time is $n \cdot \text{poly}(m\varepsilon^{-1} \log n)$.

Our algorithm relies on the following lemma, shown in [10, 38]. For each subset S of $m-1$ columns of A, let $A(S)$ be the $n \times (m-1)$ submatrix of A consisting of columns in S. Let A_{m-1} be the $n \times m$ matrix representing the ℓ_1-projections of the n points onto a hyperplane minimizing the sum of ℓ_1-distances, where we use ℓ_1-projection of a point p on a hyperplane to denote the closest point to p on the hyperplane (breaking ties arbitrarily).

LEMMA 18. *(Observed in [10])*

$$\|A - A_{m-1}\|_1 = \min_{j \in [m]} \min_{x \in \mathbb{R}^m} \|A([m] \setminus \{j\})x - A(\{j\})\|_1.$$

PROOF. In Corollary 2.3 of [38], it is shown that if w is the normal vector for the hyperplane and q is the ℓ_1-projection onto it, then one of its possible ℓ_1-projections is $p(q) = q - \frac{\langle w, q \rangle}{\|w\|_\infty} \cdot y(w)$, where $y(w)$ is any vector which is $\text{sign}(w_i)$ on a coordinate i for which $w_i = \|w\|_\infty$, and is 0 on all other coordinates. The ℓ_1-projection amounts to changing one coordinate of q, and the identity of this

coordinate depends only on w. Hence, $\|A - A_{m-1}\|_1 = \min_{j\in[m]} \min_{x\in\mathbb{R}^m} |A([m]\setminus\{j\})x - A(\{j\})|_1$. $\square$

This leads to the following algorithm. For each $j \in [m]$, we solve the ℓ_1-regression problem $\min_{x\in\mathbb{R}^m} \|A([m]\setminus\{j\})x - A(\{j\})\|_1$ up to a $(1+\varepsilon)$-factor using our ℓ_1-regression algorithm. We can solve ℓ_1-regression for all sets S by multiplying the space and time by a factor of m. We return the set S and corresponding x which result in the minimum cost. We therefore have the following theorem.

THEOREM 19. *There is a 1-pass $n\cdot\mathrm{poly}(m\varepsilon^{-1}\log n)$-time $\mathrm{poly}(m\varepsilon^{-1}\log n)$-space algorithm for approximating the ℓ_1-norm best fit hyperplane problem up to a factor of $1+\varepsilon$ in the turnstile model.*

Acknowledgements. We thank Ken Clarkson, Piotr Indyk, and the anonymous referees for helpful comments.

7. REFERENCES

[1] S. Agarwal, M. K. Chandraker, F. Kahl, D. J. Kriegman, and S. Belongie. Practical global optimization for multiview geometry. In *ECCV (1)*, pages 592–605, 2006.

[2] N. Ailon and B. Chazelle. The fast johnson–lindenstrauss transform and approximate nearest neighbors. *SIAM J. Comput.*, 39(1):302–322, 2009.

[3] N. Ailon and E. Liberty. An almost optimal unrestricted fast johnson-lindenstrauss transform. In *SODA*, 2011.

[4] N. Alon, Y. Matias, and M. Szegedy. The Space Complexity of Approximating the Frequency Moments. *J. Comput. Syst. Sci.*, 58(1):137–147, 1999.

[5] A. Andoni, K. D. Ba, P. Indyk, and D. P. Woodruff. Efficient sketches for earth-mover distance, with applications. In *FOCS*, pages 324–330, 2009.

[6] H. Auerbach. *On the Area of Convex Curves with Conjugate Diameters*. PhD thesis, University of Lwów, Lwów, Poland, 1930. (in Polish).

[7] J. Bernués and M. López-Valdes. Tail estimates and random embedding of ℓ_p^n into $\ell_r^{(1+\varepsilon)n}$, $0 < r < p < 2$. 1–2:9–18, 2007.

[8] J. Bourgain, J. Lindenstrauss, and V. Milman. Approximation of zonoids by zonotopes. *Acta Math*, 162:73–141, 1989.

[9] B. Brinkman and M. Charikar. On the impossibility of dimension reduction in ℓ_1. *J. ACM*, 52(5):766–788, 2005.

[10] J. Brooks and J. Dulá. The ℓ_1-norm best-fit hyperplane problem. Technical report, Optimization Online, 2009.

[11] J. Brooks, J. Dulá, and E. Boone. A pure ℓ_1-norm principal component analysis. Technical report, Optimization Online, 2010.

[12] N. Campbell. Robust procedures in multivariate analysis i: Robust covariance estimation. *Applied Statistics*, 29:231–237, 1980.

[13] E. J. Candès and T. Tao. Near-optimal signal recovery from random projections: Universal encoding strategies? *IEEE Transactions on Information Theory*, 52(12):5406–5425, 2006.

[14] M. Charikar and A. Sahai. Dimension reduction in the ℓ_1 norm. In *FOCS*, pages 551–560, 2002.

[15] S. Chetterjee, A. Hadi, and B. Price. *Regression Analysis by Example*. Wiley, New York, 2000.

[16] K. L. Clarkson. Subgradient and sampling algorithms for ℓ_1 regression. In *SODA*, 2005.

[17] K. L. Clarkson. Tighter bounds for random projections of manifolds. In *Symposium on Computational Geometry*, pages 39–48, 2008.

[18] K. L. Clarkson and D. P. Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC)*, pages 205–214, 2009.

[19] A. Dasgupta, P. Drineas, B. Harb, R. Kumar, and M. W. Mahoney. Sampling algorithms and coresets for ℓ_p regression. *SIAM J. Comput.*, 38(5):2060–2078, 2009.

[20] M. Day. Polygons circumscribed about closed convex curves. *Trans. Amer. Math. Soc.*, 62:315–319, 1947.

[21] P. Drineas, M. W. Mahoney, S. Muthukrishnan, and T. Sarlós. Faster least squares approximation. *CoRR*, abs/0710.1435, 2007.

[22] D. Feldman, M. Monemizadeh, C. Sohler, and D. P. Woodruff. Coresets and sketches for high dimensional subspace problems. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2010.

[23] O. Friedland and O. Guéden. Random embedding of ℓ_p^n into ℓ_r^n. 2010.

[24] J. Galpin and D. Hawkins. Methods of ℓ_1 estimation of a covariance matrix. *Computational Statistics and Data Analysis*, 5:305–319, 1987.

[25] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer-Verlag, New York, 2003.

[26] P. Indyk. Algorithmic applications of low-distortion embeddings. In *FOCS*, pages 10–33, 2001.

[27] P. Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM*, 53(3):307–323, 2006.

[28] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *STOC*, pages 604–613, 1998.

[29] P. Indyk and D. P. Woodruff. Optimal approximations of the frequency moments of data streams. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 202–208, 2005.

[30] W. Johnson and G. Schechtman. Very tight embeddings of subspaces of l_p, $1 = p < 2$, into ℓ_p^n. *Geometric and Functional Analysis*, (4):845–851, 2003.

[31] D. M. Kane, J. Nelson, and D. P. Woodruff. On the exact space complexity of sketching and streaming small norms. In *SODA*, 2010.

[32] Q. Ke and T. Kanade. Robust subspace computation using ℓ_1 norm, 2003. Technical Report CMU-CS-03-172, Carnegie Mellon University, Pittsburgh, PA.

[33] Q. Ke and T. Kanade. Robust l_1 norm factorization in the presence of outliers and missing data by alternative convex programming. In *CVPR (1)*, pages 739–746, 2005.

[34] N. Kwak. Principal component analysis based on l1-norm maximization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(9):1672–1680, 2008.

[35] J. Langford, L. Li, and A. Strehl. Vowpal wabbit online learning. Technical report, 2007.

[36] J. R. Lee, A. Naor, and A. Immediate. Embedding the diamond graph in ℓ_p and dimension reduction in ℓ_1, 2003.

[37] D. Lewis. Finite dimensional subspaces of l_p. *Studia Math*, 63:207–211, 1978.

[38] O. L. Mangasarian. Arbitrary-norm separating plane. *Operations Research Letters*, 24:15–23, 1997.

[39] R. A. Maronna, D. R. Martin, and V. J. Yohai. *Robust Statistics: Theory and Methods*. Wiley, 2006.

[40] S. Muthukrishnan. Data Streams: Algorithms and Applications. *Foundations and Trends in Theoretical Computer Science*, 1(2):117–236, 2005.

[41] V. Rokhlin, A. Szlam, and M. Tygert. A randomized algorithm for principal component analysis. *SIAM Journal on Matrix Analysis and Applications*, 31(3):1100, 1124, 2009.

[42] T. Sarlós. Improved approximation algorithms for large matrices via random projections. In *FOCS*, pages 143–152, 2006.

[43] Schechtman. More on embedding subspaces of l_p into ℓ_r^n. *Composition Math*, 61:159–170, 1987.

[44] G. A. Seber and A. J. Lee. *Linear Regression Analysis*. Wiley Series in Probability and Statistics, 2003.

[45] Q. Shi, J. Petterson, G. Dror, J. Langford, A. J. Smola, A. Strehl, and V. Vishwanathan. Hash kernels. In *AISTATS 12*, 2009.

[46] D. A. Spielman and N. Srivastava. Graph sparsification by effective resistances. In *STOC*, pages 563–568, 2008.

[47] M. Talagrand. Embedding subspaces of l_1 into ℓ_1^n. *Proceedings of the American Mathematical Society*, 108(2):363–369, 1990.

[48] A. Taylor. A geometric theorem and its applications to biorthogonal systems. *Bull. Amer. Math Soc.*, 53:614–616, 1947.

[49] M. Thorup and Y. Zhang. Tabulation based 4-universal hashing with applications to second moment estimation. In *SODA*, pages 615–624, 2004.

[50] K. Q. Weinberger, A. Dasgupta, J. Attenberg, J. Langford, and A. J. Smola. Feature hashing for large scale multitask learning. *CoRR*, abs/0902.2206, 2009.

[51] X. Yan and X. Su. *Linear Regression Analysis: Theory and Computing*. World Scientific, 2009.