Leveraging Well-Conditioned Bases: Streaming and Distributed Summaries in Minkowski p-Norms

Graham Cormode [*1] **Charlie Dickens** [*1] **David P. Woodruff** [*2]

Abstract

Work on approximate linear algebra has led to efficient distributed and streaming algorithms for problems such as approximate matrix multiplication, low rank approximation, and regression, primarily for the Euclidean norm ℓ_2. We study other ℓ_p norms, which are more robust for $p < 2$, and can be used to find outliers for $p > 2$. Unlike previous algorithms for such norms, we give algorithms that are (1) deterministic, (2) work simultaneously for every $p \geq 1$, including $p = \infty$, and (3) can be implemented in both distributed and streaming environments. We apply our results to ℓ_p-regression, entrywise ℓ_1-low rank approximation, and approximate matrix multiplication.

1. Introduction

Analyzing high dimensional, high volume data can be time-consuming and resource intensive. Core data analysis, such as robust instances of regression, involve convex optimization tasks over large matrices, and do not naturally distribute or parallelize. In response to this, approximation algorithms have been proposed which follow a "sketch and solve" paradigm: produce a reduced size representation of the data, and solve a version of the problem on this summary (Woodruff, 2014). It is then argued that the solution on the reduced data provides an approximation to the original problem on the original data. This paradigm is particularly attractive when the summarization can be computed efficiently on partial views of the full data—for example, when it can be computed incrementally as the data arrives (streaming model) or assembled from summarizations of disjoint partitions of the data (distributed model) (Woodruff, 2014; Agarwal et al., 2012; Feldman et al., 2006). This template has been instantiated for a number of fundamental tasks in high dimensional linear algebra such as matrix multiplication, low rank approximation, and regression.

Our understanding is well-established in the common case of the Euclidean norm, i.e., when distances are measured under the Minkowski p-norm for $p = 2$. Here, it suffices to choose a sketching matrix independent of the data—where each entry is i.i.d. Gaussian, Rademacher, or more efficient variants of these. For other p values, less is known, but these are often needed to handle limitations of the 2-norm. For instance, $p = 1$ is widely used as it is extremely robust with respect to the presence of outliers while $p > 2$ can be used to detect outlying observations.

We continue the study of algorithms for ℓ_p norms on streaming and distributed data. A particular novelty of our results is that unlike previous distributed and streaming algorithms, they can all be implemented *deterministically*, i.e., our algorithms make no random choices. While in a number of settings randomized algorithms are highly beneficial, leading to massive computational savings, there are other applications which require extremely high reliability, for which one needs to obtain guaranteed performance across a large number of inputs. If one were to use a randomized algorithm, then it would need vanishingly small error probability; however, many celebrated algorithms in numerical linear algebra succeed with only constant probability. Another limitation of randomized algorithms was shown in (Hardt & Woodruff, 2013): if the input to a randomized sketch depends on the output of a preceding algorithm using the same sketch, then the randomized sketch can give an arbitrarily bad answer. Hence, such methods cannot handle adaptively chosen inputs. Thus, while randomized algorithms certainly have their place, the issues of high reliability and adaptivity motivate the development of deterministic methods for a number of other settings, for which algorithms are scarce.

Our techniques can be viewed as a conceptual generalization of Liberty's Frequent Directions (in the 2-norm) (Liberty, 2013), which progressively computes an SVD on subsequent blocks of the input. This line of work (Liberty, 2013; Ghashami & Phillips, 2014; Ghashami et al., 2016; Ghashami et al., 2016) is the notable exception in numerical linear algebra, as it provides deterministic methods,

[*]Equal contribution [1]Department of Computer Science, University of Warwick, Coventry, UK [2]School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA. Correspondence to: Charlie Dickens <c.dickens@warwick.ac.uk>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018.

although all such methods are specific to the 2-norm. Our core algorithm is similar in nature, but we require a very different technical analysis to argue that the basis transformation computed preserves the shape in the target p-norm.

Our main application is to show how high dimensional regression and low rank approximation problems can be solved approximately and deterministically in the sketch and solve paradigm. The core of the summary is to find rows of the original matrix which have high *leverage scores*. That is, they contain a lot of information about the shape of the data. In the Euclidean norm, leverage scores correspond directly to row norms of an orthonormal basis. This is less straightforward for other ℓ_p norms, where the scores correspond to the row norms of so-called ℓ_p-well-conditioned bases. Moreover, while leverage scores are often used for sampling in randomized algorithms, we use them here in the context of fully deterministic algorithms.

We show how a superset of rows with high leverage scores can be found for arbitrary ℓ_p norms, based on only local information. This leads to efficient algorithms which identify rows with high (local) leverage scores within subsets of the data, and proceed hierarchically to collect a sufficient set of rows. These rows then allow us to solve regression problems: essentially, we solve the regression problem corresponding to just the retained input rows. We apply this technique to ℓ_p-regression and entrywise ℓ_p-low rank approximation. In particular, we use it to solve the ℓ_∞-regression problem with additive error in a stream. Note that the ℓ_∞ problem reduces to finding a ball of minimum radius which covers the data, and global solutions are slow due to the need to solve a linear program. Instead, we show that only a subset of the data needs to be retained in the streaming model to compute accurate approximations. Given the relationship between the streaming model and the distributed model that we later define, this could be seen in the context of having data stored over multiple machines who could send 'important' rows of their data to a central coordinator in order to compute the approximation.

Summary of Results. All our algorithms are deterministic polynomial time, and use significantly sublinear memory or communication in streaming and distributed models, respectively. We consider tall and thin $n \times d$ matrices A for overconstrained regression so one should think of $n \gg d$. We implement both deterministic and randomized variants of our algorithms.

Section 3 presents an algorithm which returns rows of high 'importance' in a data matrix with additive error. This follows by storing a polynomial number (in d) of rows and using these to compute a well-conditioned basis. The key insight here is that rows of high norm in the full well-conditioned basis cannot have their norm decrease too much in a well-conditioned basis associated with a subblock; in fact they remain large up to a multiplicative poly(d) factor.
Section 4 gives a method for computing a so-called ℓ_p-subspace embedding of a data matrix in polynomial time. The space is n^γ to obtain $d^{O(1/\gamma)}$ distortion, for $\gamma \in (0, 1)$ a small constant. This result is then applied to ℓ_p-regression which is shown to have a poly(d) approximation factor with the same amount of space.
Section 5 describes a deterministic algorithm which gives a poly(k)-approximation to the optimal low rank approximation problem in entrywise ℓ_1-norm. It runs in polynomial time for constant k. This method builds on prior work by derandomizing a subroutine from (Song et al., 2017).
Section 6 describes an algorithm for computing an additive-error solution to the ℓ_∞-regression problem, and shows a corresponding lower bound, showing that relative error solutions in this norm are not possible in sublinear space, even for randomized algorithms.
Section 7 concludes with an empirical evaluation. More experiments, intermediate results, and formal proofs can be found in the Supplementary Material, as can results on approximate matrix multiplication.

Comparison to Related Work. There is a rich literature on algorithms for numerical linear algebra in general p-norms; most of which are randomized with the notable exception of Frequent Directions. The key contributions of our work for each of the problems considered and its relation to prior work is as follows:

Finding high leverage rows: our algorithm is a single pass streaming algorithm and uses small space. We show that the global property of ℓ_p-leverage scores can be understood by considering only local statistics. Frequent Directions is the only comparable result to ours and outputs a summary of the rows only in the ℓ_2-norm. However, our method covers all $p \geq 1$. Theorem 3.3 is the key result and is later used to prove Theorem 6.1 and approximate the ℓ_∞-regression problem.

Subspace embedding, regression and ℓ_1 low-rank approximation: various approaches using row-sampling (Cohen & Peng, 2015; Dasgupta et al., 2008), and data oblivious methods such as low-distortion embeddings can solve regression in time proportional to the sparsity of the input matrix (Clarkson et al., 2013; Meng & Mahoney, 2013; Song et al., 2017; Woodruff & Zhang, 2013). However, despite the attractive running times and error guarantees of these works, they are all randomized and do not necessarily translate well to the streaming model of computation. Our contribution here is a fully deterministic algorithm that works for all $p \geq 1$ in both streaming and distributed models. Randomized methods for ℓ_1 low-rank approximation have also been developed in (Song et al., 2017) and our result exploits a derandomized subroutine from this work to obtain a deterministic result which applies in both models.

2. Preliminaries and Notation

We consider computing ℓ_p-leverage scores of a matrix, low-rank approximation, regression, and matrix multiplication. We assume the input is a matrix $A \in \mathbb{R}^{n\times d}$ and $n \gg d$ so $\mathrm{rank}(A) \leq d$ and the regression problems are over-constrained. Without loss of generality we may assume that the columns of the input matrix are linearly independent so that $\mathrm{rank}(A) = d$. Throughout this paper we rely heavily on the notion of a *well-conditioned basis* for the column space of an input matrix, in the context of the *entrywise p-norm* which is $\|A\|_p = (\sum_{i,j} |A_{ij}|^p)^{1/p}$.

Definition 2.1 (Well-conditioned basis)**.** Let $A \in \mathbb{R}^{n\times d}$ have rank d. For $p \in [1,\infty)$ let $q = \frac{p}{p-1}$ be its dual norm. An $n \times d$ matrix U is an (α, β, p)-well-conditioned basis for A if the column span of U is equal to that of A, $\|U\|_p \leq \alpha$, for all $z \in \mathbb{R}^d, \|z\|_q \leq \beta\|Uz\|_p$, and $\alpha, \beta, d^{O(1)}$ are independent of n (Dasgupta et al., 2008).

We focus on the cases $p < 2$ and $p > 2$ because the deterministic $p = 2$ case is relatively straightforward. Indeed, for $p = 2$, A^TA can be maintained incrementally as rows are added, allowing x^TA^TAx to be computed for any vector x. So it is possible to find an exact ℓ_2 subspace embedding using $O(d^2)$ space in a stream and $O(nd^{\omega-1})$ time (ω is the matrix multiplication constant). We adopt the convention that when $p = 1$ we take $q = \infty$.

Theorem 2.2 ((Dasgupta et al., 2008))**.** Let A be an $n \times d$ matrix of rank d, let $p \in [1,\infty)$ and let q be its dual norm. There exists an (α, β, p)-well-conditioned basis U for the column space of A such that:

1. if $p < 2$ then $\alpha = d^{\frac{1}{p}+\frac{1}{2}}$ and $\beta = 1$,
2. if $p = 2$ then $\alpha = \sqrt{d}$ and $\beta = 1$, and
3. if $p > 2$ then $\alpha = d^{\frac{1}{p}+\frac{1}{2}}$ and $\beta = d^{\frac{1}{p}-\frac{1}{2}}$.

Moreover, U can be computed in deterministic time $O(nd^2 + nd^5 \log n)$ for $p \neq 2$ and $O(nd^2)$ if $p = 2$.

We freely use the fact that a well-conditioned basis $U = AR$ can be efficiently computed for the given data matrix A. Details for the computation can be found in (Dasgupta et al., 2008) but this is done by computing a change of basis R such that $U = AR$ is well-conditioned. Similarly, as R can be inverted we have the relation that $UR^{-1} = A$. Both methods are used so we adopt the convention that $U = AR$ when writing a well-conditioned basis in terms of the input and $US = A$ for the input in terms of the basis.

2.1. Computation Models

Our algorithms operate under the *streaming* and *distributed* models of computation. In both settings an algorithm receives as input a matrix $A \in \mathbb{R}^{n\times d}$. For a problem **P**, the algorithm must keep a subset of the rows of A and, upon reading the full input, may use a black-box solver to compute an approximate solution to **P** with only the subset of rows stored. In both models we measure the *summary size* (storage), the *update time* which is the time taken to find the local summary, and the *query time* which is the time taken to compute an approximation to **P** using the summary.

The Streaming Model: The rows of A are given to the (centralized) algorithm one-by-one. Let b be the maximum number of rows that can be stored under the constraint that b is sublinear in n. The stored subset is used to compute *local* statistics which determine those rows to be kept or discarded from the stored set. Further rows are then appended and the process is repeated until the full matrix has been read. An approximation to the problem is then computed by solving **P** on the reduced subset of rows.

The Distributed Summary Model: Given a small constant $\gamma \in (0, 1)$, the input in the form of matrix $A \in \mathbb{R}^{n\times d}$ is partitioned into blocks among distributed compute nodes so that no block exceeds n^γ rows. The computation then follows a tree structure: the initial blocks of the matrix form $n^{1-\gamma}$ leaves of the compute tree. Each internal node merges and reduces its input from its child nodes. The first phase is for the leaf nodes $l_1, \ldots, l_m$ of the tree to reduce their input by computing a local summary on the block they receive as input. This is then sent to parent nodes $p_1, \ldots, p_m$ which merge and reduce the received rows until the space bound is reached. The resulting summaries are passed up the tree until we reach the root where a single summary of bounded size is obtained which can be used to compute an approximation to **P**. In total, there are $O(1/\gamma)$ levels in the tree. As the methods require only light synchronization (compute summary and return to coordinator), we do not model implementation issues relating to synchronization.

Remark 2.3. The two models are quite close: the *streaming model* can be seen as a special case of the distributed model with only one participant who individually computes a summary, appends rows to the stored set, and reduces the new summary. This is represented as a deep binary tree, where each internal node has one leaf child. Likewise, the *Distributed Summary Model* can be implemented in a full streaming fashion over the entire binary tree. The experiments in Section 7 perform one round of merge-and-reduce in the distributed model to simulate the streaming approach.

3. Finding Rows of High Leverage

This section is concerned with finding rows of high leverage from a matrix with respect to various p-norms. We conclude the section with an algorithm that returns rows of high leverage up to polynomial additive error.

Definition 3.1. Let R be a change of basis matrix such that AR is a well-conditioned basis for the column space of A.

The (full) ℓ_p-leverage scores are defined as $w_i = \|\mathbf{e}_i^T AR\|_p^p$.

Note that w_i depends both on A and the choice of R, but we suppress this dependence in our notation. Next we present some basic facts about the ℓ_p leverage scores.

Fact 1. By Definition 2.1 we have $\sum_i w_i = \sum_i \|(AR)_i\|_p^p \leq \alpha^p$. Theorem 2.2 shows $\alpha = \text{poly}(d)$. Define $I = \{i \in [n] : w_i > \tau \|AR\|_p^p\}$ to be the index set of all rows whose ℓ_p leverage exceeds a τ fraction of $\|AR\|_p^p$, then: $\alpha^p \geq \sum_i w_i \geq \sum_{i \in I} w_i \geq |I| \cdot \tau \|AR\|_p^p$. Hence, $|I| \leq \alpha^p / \tau \|AR\|_p^p = \text{poly}(d)/\tau$. So there are at most $\text{poly}(d)/\tau$ rows i for which $w_i \geq \tau \|AR\|_p^p$.

Fact 2. Definition 2.1 and Hölder's inequality show that for any vector $\mathbf{x}$ we have $|(AR\mathbf{x})_i|^p \leq \beta \|\mathbf{e}_i^T AR\|_p^p \cdot \|ARx\|_p^p$. Then $\tau \leq |\mathbf{e}_i^T AR\mathbf{x}|^p / \|AR\mathbf{x}\|_p^p \leq \beta w_i$. From this we deduce that if a row contributes at least a τ fraction of $\|AR\mathbf{x}\|_p^p$ then $\tau \leq w_i \beta$. That is, $\tau \leq w_i$ for $p \in [1, 2]$ and $\tau \leq d^{1/2} w_i$ for $p \in (2, \infty)$ by using Theorem 2.2.

Definition 3.2. Let X be a matrix and Y be a subset of the rows of X. Define the *local ℓ_p-leverage scores of Y with respect to X* to be the leverage scores of rows Y found by computing a well-conditioned basis for Y rather than the whole matrix X.

A key technical insight to proving Theorem 3.3 below is that rows of high leverage globally can be found by repeatedly finding rows of local high leverage. While relative ℓ_p row norms of a submatrix are at least as large as the full relative ℓ_p norms, it is not guaranteed that this property holds for leverage scores. This is because leverage scores are calculated from a well-conditioned basis for a matrix which need not be a well-conditioned basis for a block. However, we show that local ℓ_p leverage scores restricted to a coordinate subspace of a matrix basis do not decrease too much when compared to leverage scores in the original space. Let i be a row in A with local leverage score $\hat{w}_i$ and global leverage score w_i. Then $\hat{w}_i \geq w_i / \text{poly}(d)$. The proof relies heavily on properties of the well-conditioned basis and details are given in the Supplementary Material, Lemma A.1. This lemma shows that local leverage scores can potentially drop in arbitrary ℓ_p norm, contrasting the behavior in ℓ_2. However, it is possible to find all rows exceeding a threshold globally by altering the local threshold. That is, to find all $w_i > \tau$ globally we can find all local leverage scores exceeding an adjusted threshold $\hat{w}_i > \tau / \text{poly}(d)$ to obtain a superset of all rows which exceed the global threshold. The price to pay for this is a $\text{poly}(d)$ increase in space cost which, importantly, remains *sublinear in* n. Hence, we can gradually prune out rows of small leverage and keep only the most important rows of a matrix. Combining Lemmas A.1 and A.2 we can present the main theorem of the section.

We prove Theorem 3.3 by arguing the correctness of Algorithm 1 which reads A once only, row by row, and so operates in the streaming model of computation as follows. Let A' be the submatrix of A induced by the b block of $\text{poly}(d)/\tau$ rows. Upon storing A', we compute U, a local well-conditioned basis for A' and the local leverage scores with respect to U, $\hat{w}_i(U)$ are calculated. Now, the local and global leverage scores can be related by Lemma A.1 as $w_i / \text{poly}(d) \leq \hat{w}_i$ so we can decide which rows to keep using an adjusted threshold. Any i for which the local leverage exceeds the adjusted threshold is kept in the sample and all other rows are deleted. The sample cannot be too large by properties of the well-conditioned basis and leverage scores so these kept rows can be appended to the next block which is read in before computing another well-conditioned basis and repeating in the same fashion. The proof of Theorem 3.3 is deferred to Appendix A.

Theorem 3.3. Let $\tau > 0$ be a fixed constant and let b denote a bound on the available space. There exists a deterministic algorithm, namely, Algorithm 1, which computes the ℓ_p-leverage scores of a matrix $A \in \mathbb{R}^{n \times d}$ with $O(bd^2 + bd^5 \log b)$ update time, $\text{poly}(d)$ space, and returns all rows of A with ℓ_p leverage score satisfying $w_i \geq \tau / \text{poly}(d)$.

4. ℓ_p-Subspace Embeddings

Under the assumptions of the Distributed Summary Model we present an algorithm which computes an ℓ_p-subspace embedding. By extension, this applies to both the distributed and streaming models of computation as described in Section 2.1. Two operations are needed for this model of computation: the merge and reduce steps. To reduce the input at each level a summary is computed by taking a block of input B (corresponding to a leaf node or a node higher up the tree) and computing a well-conditioned basis $B = US$. In particular, the summary is now the matrix S with U and B deleted. For the merge step, successive matrices S are concatenated until the space requirement is met. A further reduce step takes as input this concatenated matrix and the process is repeated. Further details, pseudocode, and proofs for this section are given in Appendix B.

Definition 4.1. A matrix T is a relative error (c_1, c_2)-ℓ_p subspace embedding for the column space of a matrix $A \in \mathbb{R}^{n \times d}$ if there are constants $c_1, c_2 > 0$ so that for all $x \in \mathbb{R}^d$, $c_1 \|Ax\|_p \leq \|Tx\|_p \leq c_2 \|Ax\|_p$.

Theorem 4.2. Let $A \in \mathbb{R}^{n \times d}$, $p \neq 2, \infty$ be fixed and fix a constant $\gamma \in (0, 1)$. Then there exists a one-pass deterministic algorithm which constructs a $(1/d^{O(1/\gamma)}, 1)$ *relative error* ℓ_p-subspace embedding in with $O(n^\gamma d^2 + n^\gamma d^5 \log n^\gamma)$ update time and $O(n^\gamma d)$ space in the streaming and distributed models of computation.

The algorithm is used in a tree structure as follows: split input $A \in \mathbb{R}^{n \times d}$ into $n^{1-\gamma}$ blocks of size n^γ, these form the leaves of the tree. For each block, a well-conditioned basis is

Algorithm 1 Deterministic High Leverage Scores

Require: $A \in \mathbb{R}^{n\times d}, \tau \in (0,1)$
procedure HIGH LEVERAGE SCORES(A, τ)
  $b \leftarrow \text{poly}(d)/\tau$
  $A' \leftarrow$ first b rows of A
  $B \leftarrow$ LEVSCORECHECK(wcb(A'), $A', \tau/\text{poly}(d)$)
  while Rows of A unseen **do**
    $A' \leftarrow$ next b rows of A
    $B \leftarrow$ LEVSCORECHECK(wcb($[A';B]$), $[A';B], \tau/\text{poly}(d)$)
Ensure: B

Algorithm 2 Finding high leverage rows

Require: Well-conditioned basis X for matrix W, threshold parameter $\tau > 0$
procedure LEVSCORECHECK(X, W, τ)
  $N \leftarrow$ Number of rows in X
  $Y \leftarrow \mathbf{0}$
  for $i = 1 : N$ **do**
    if $w_i(X) > \tau$ **then**
      $Y_i \leftarrow W_i$
Ensure: Nonzero rows of Y

Figure 1: $[X;Y]$ denotes row-wise appending of matrices, $U = \text{wcb}(M)$ denotes that U is a well-conditioned basis for M.

computed and the change of basis matrix $S \in \mathbb{R}^{d\times d}$ is stored and passed to the next level of the tree. This is repeated until the concatenation of all the S matrices would exceed n^γ. At this point, the concatenated S matrices form the parent node of the leaves in the tree and the process is repeated upon this node: this is the merge and reduce step of the algorithm. At every iteration of the merge-and-reduce steps it can be shown that a distortion of $1/d$ is introduced by using the summaries S. However, this can be controlled across all of the $O(1/\gamma)$ levels in the tree to give a deterministic relative error ℓ_p subspace embedding which requires only sublinear space and little communication. In addition, the subspace embedding can be used to achieve a deterministic relative-error approximate regression result. The proof relies upon analyzing the merge-and-reduce behaviour across all nodes of the tree.

ℓ_p-Regression Problem: Given matrix $A \in \mathbb{R}^{n\times d}$ and target vector $b \in \mathbb{R}^n$, find $\hat{x} = \text{argmin}_x \|Ax - b\|_p$.

Theorem 4.3. Let $A \in \mathbb{R}^{n\times d}, b \in \mathbb{R}^n$, fix $p \neq 2, \infty$ and a constant $\gamma > 0$. The *ℓ_p-regression problem* can be solved deterministically in the streaming and distributed models with a $(d+1)^{O(1/\gamma)} = \text{poly}(d)$ *relative error* approximation factor. The update time is $\text{poly}(n^\gamma(d+1))$ and $O((1/\gamma)n^\gamma(d+1))$ storage. The query time is $\text{poly}(n^\gamma)$ for the cost of convex optimization.

5. Low-Rank Approximation

ℓ_1-Low-Rank Approximation Problem: Given matrix $A \in \mathbb{R}^{n\times d}$ output a matrix B of rank k s.t., for constant k:

$$\|A - B\|_1 \leq \text{poly}(k) \min_{A':\text{rank}k} \|A - A'\|_1. \qquad (1)$$

Theorem 5.1. Let $A \in \mathbb{R}^{n\times d}$ be the given data matrix and k be the (constant) target rank. Let $\gamma > 0$ be an arbitrary (small) constant. Then there exists a deterministic distributed and streaming algorithm (namely Algorithm 5 in Appendix C) which can output a solution to the ℓ_1-Low Rank Approximation Problem with *relative error* $\text{poly}(k)$ approximation factor, update time $\text{poly}(n,d)$, space bounded by $n^\gamma\text{poly}(d)$, and query time $\text{poly}(n,d)$.

The key technique is similar to that of the previous section by using a tree structure with merge-and-reduce operations. For input $A \in \mathbb{R}^{n\times d}$ and constant $\gamma > 0$ partition A into $n^{1-\gamma}$ groups of rows which form the leaves of the tree. The tree is defined as previously with the same 'merge' operation, but the 'reduce' step to summarize the data exploits a derandomization (subroutine Algorithm 4) of (Song et al., 2017) to compute an approximation to the optimal ℓ_1-low-rank approximation. Once this is computed, k of the rows in the summary are kept for later merge steps.

This process is continued with the successive k rows from n^γ rows being 'merged' or added to the matrix until it has n^γ rows. The process is repeated across all of the groups in the level and again on the successive levels on the tree from which it can be shown that the error does not propagate too much over the tree, thus giving the desired result.

6. Application: ℓ_∞-Regression

Here we present a method for solving ℓ_∞-regression in a streaming fashion. Given input A and a target vector b, it is possible to achieve additive approximation error of the form $\varepsilon\|b\|_p$ for arbitrarily large p. This contrasts with both Theorems 4.2 and 4.3 which achieve a relative error $\text{poly}(d)$ approximation. Both of these theorems require that p is constant and not equal to the ∞-norm. This restriction is due to a lower bound for ℓ_∞- regression showing that it cannot be approximated with relative error in sublinear space. The key to proving Theorem 6.1 below is using Theorem 3.3 to find high leverage rows and arguing that these are sufficient to give the claimed error guarantee.

The ℓ_∞-regression problem has been previously studied in the overdetermined case and can naturally be applied to curve-fitting under this norm. ℓ_∞-regression can be solved

by linear programming (Sposito, 1976) and such a transformation allows the identification of outliers in the data. Also, if the errors are known to be distributed uniformly across an interval then ℓ_∞-regression estimator is the maximum-likelihood parameter choice (Hand, 1978). The same work argues that such uniform distributions on the errors often arise as round-off errors in industrial applications whereby the error is controlled or is small relative to the signal. There are further applications such as using ℓ_∞-regression to remove outliers prior to ℓ_2 regression in order to make the problem more robust (Shen et al., 2014). By applying ℓ_∞ regression on subsets of the data an approximation to the Least Median of Squares (another robust form of regression) can be found. We now define the problem and proceed to show that it is possible to compute an approximate solution with additive error in ℓ_p-norm for arbitrarily large p.

Approximate ℓ_∞-Regression problem: Given data $A \in \mathbb{R}^{n\times d}$, target vector $b \in \mathbb{R}^n$, and error parameter $\varepsilon > 0$, compute an additive $\varepsilon\|b\|_p$ error solution to:

$$\min_{x\in\mathbb{R}^d} \|Ax - b\|_\infty = \min_{x\in\mathbb{R}^d} \left[\max_i |(Ax)_i - b_i|\right].$$

Theorem 6.1. Let $A \in \mathbb{R}^{n\times d}, b \in \mathbb{R}^n$ and fix constants $p \geq 1, \varepsilon > 0$ with $p \neq \infty$. There exists a one-pass deterministic streaming algorithm which solves the *ℓ_∞-regression problem* up to an additive $\varepsilon\|b\|_p$ error in $d^{O(p)}/\varepsilon^{O(1)}$ space, $O(md^5 + md^2 \log m)$ update time and $T_{\text{solve}}(m, d)$ query time.

Note that $T_{\text{solve}}(m, d)$ query time is the time taken to solve the linear program associated with the above problem on a reduced instance size. Also, observe that Theorem 6.1 requires $p < \infty$. This restriction is necessary to forbid relative error with respect to the infinity norm. Indeed, p can be an arbitrarily large constant, but for $p = \infty$ we can look for rows above an $\varepsilon/\text{poly}(d)$ threshold in the case when A is an all-ones column n-vector (so an $n \times 1$ matrix). Then $\|Ax\|_\infty = \|x\|_\infty$ since x is a scalar. Also, A is a well-conditioned basis for its own column span but the number of rows of leverage exceeding $\varepsilon/\text{poly}(d) = \varepsilon$ is n for a small constant ε. This intuition allows us to prove the following theorem.

Theorem 6.2. Any algorithm which outputs an $\varepsilon\|b\|_\infty$ relative error solution to the ℓ_∞-regression problem requires $\min\left\{n, 2^{\Omega(d)}\right\}$ space.

7. Experimental Evaluation

To validate our approach, we evaluate the use of high ℓ_p-leverage rows in order to approximate ℓ_∞-regression[1], focusing particularly on the cases using ℓ_1 and ℓ_2 well-conditioned bases. It is straightforward to model ℓ_∞-regression as a linear program in the offline setting. We use this to measure the accuracy of our algorithm. The implementation is carried out in the single pass streaming model with a fixed space constraint, m, and threshold, α^p/m for both conditioning methods to ensure the number of rows kept in the summary did not exceed m. Recall from Remark 2.3 that the single-pass streaming implementation is equivalent to the distributed model with only one participant applying merge-and-reduce, so this experiment can also be seen as a distributed computation with the merge step being the appending of new rows and the reduce step being the thresholding in the new well-conditioned basis.

Methods. We analyze two instantiations of our methods based on how we find a well-conditioned basis and repeat over 5 independent trials with random permutations of the data. The methods are as follows:

`SPC3`: We use an algorithm of Yang et al. (2013) to compute an ℓ_1-wcb. This method is randomized as it employs the Sparse Cauchy Transform and is only an ℓ_1-well-conditioned basis with constant probability We also implemented a check condition which showed that almost always, roughly 99% of the time, the randomized construction `SPC3` would return a $(d^{2.5}, 1, 1)$-well-conditioned basis. Thus, we bypassed this check in our experiment to ensure quick update times.

`Orth`: In addition, we also used an orthonormal basis using the QR decomposition which is an ℓ_2-wcb. This method is fully deterministic and outputs a $(\sqrt{d}, 1, 2)$-well- conditioned basis.

`Sample`: A sample of the data is chosen uniformly at random and the retained summary has size exactly m.

`Identity`: No conditioning is performed. For a block B of the input, the *surrogate* scores $w_i(B) = \|\mathbf{e}_i^T B\|_2^2/\|B\|_F^2$ are used to determine which rows to keep. As the sum of these $w_i(B)$ is 1, we keep all rows which have $w_i(B) > 2/m$. Since no more than $m/2$ of the rows can satisfy $w_i(B) > 2/m$, the size of the stored subset of rows can be controlled and cannot grow too large.

Remark 7.1. The `Identity` method keeps only the rows with high norm which contrasts our conditioning approach: if most of the mass of the block is concentrated on a few rows then these will appear heavy locally despite the possibility that they may correspond to previously seen or unimportant directions. In particular, if these heavy rows significantly outweigh the weight of some sparse directions in the data it is likely that the sparse directions will not be found at all. For instance, consider data $X \in \mathbb{R}^{n\times d}$ which is then augmented by appending the identity (and zeros) so that these are the only vectors in the new directions. That is, set $X' = [X, \mathbf{0}_{n\times k}; \mathbf{0}_{k\times d}, I_{k\times k}]$ and then

[1] Code available at `https://github.com/c-dickens/stream-summaries-high-lev-rows`

permute the rows of X'. The appended sparse vectors from $I_{k\times k}$ will have leverage of 1 so will be detected by the well-conditioned basis methods. However there is no guarantee that the `Identity` method will identify these directions if the entries in X significantly outweigh those in $I_{k\times k}$. In addition, there is also no guarantee that using uniform sampling will identify these points, particularly when k is small compared to n and d. So while choosing to do no conditioning seems attractive, this example shows that doing so may not give any meaningful guarantees and hence we prefer the approach in Section 3. We compare only to these baselines as we are not aware of any other competing methods in the small memory regime for the ℓ_∞-regression problem.

Datasets. We tested the methods on a subset of the *US Census Data* containing 5 million rows and 11 columns[2] and *YearPredictionMSD*[3] which has roughly 500,000 rows and 90 columns (although we focus on a fixed 50,000 row sample so that the LP for regression is tractable: see Figure 4c in the Supplementary Material, Appendix F). For the census dataset, space constraints between 50,000 and 500,000 rows were tested and for the YearPredictionsMSD data space budgets were tested between 2,500 and 25,000. The general behavior is roughly the same for both datasets so for brevity we primarily show the results for *US Census Data*, and defer corresponding plots for *YearPredictionsMSD* to Appendix F.

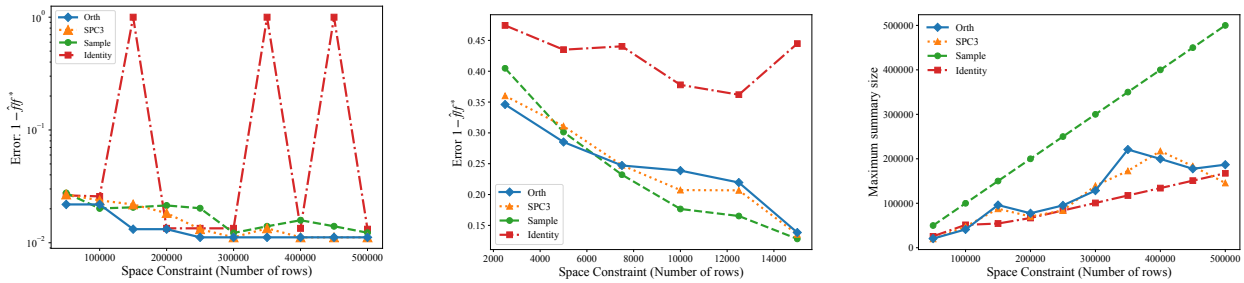
Results on approximation error compared to storage Let f^* denote the minimal value of the full regression obtained by x^* and let x' be the output of the reduced problem. The approximate solution to the full problem is then $\hat{f} = \|Ax' - b\|_\infty$ and approximation error is measured as $\hat{f}/f^* - 1$ (note that $\hat{f} \geq f^*$). An error closer to 0 demonstrates that $\hat{f}$ is roughly the same as f^* so the optimal value is well-approximated. Figures 2a and 2b show that on both datasets the `Identity` method consistently performs poorly while `Sample` achieves comparable accuracy to the conditioning methods. Despite the simplicity of uniform sampling to keep a summary, the succeeding sections discuss the increased time and space costs of using such a sample and show that doing so is not favourable. Thus, neither of the baseline methods output a summary which can be used to approximate the regression problem both *accurately* and *quickly*, hence justifying our use of leverage scores. Our conditioning methods perform particularly well in the *US Census Data* data (Figure 2a) with `Orth` appearing to give the most accurate summary and `SPC3` performing comparably well but with slightly more fluctuation: similar behaviour is observed in the *YearPredictionMSD* (Figure 2b) data too. The conditioning methods are also seen to be robust to the storage constraint, give accurate performance across both datasets using significantly less storage than sampling, and give a better estimate in general than doing no conditioning.

Results on Space Complexity. Recall that the space constraint is m rows and throughout the stream, after a local computation, the merge step concatenates more rows to the existing summary until the bound m is met, prior to computing the next reduction. During the initialization of the block A' by Algorithm 1, the number of stored rows is exactly m. However, we measure the maximum number of rows kept in a summary after every reduction step to understand how large the returned summary can grow. As seen in Figure 2c, `Identity` keeps the smallest summary but there is no reason to expect it has kept the most important rows. In contrast, if m is the bound on the summary size, then uniform sampling always returns a summary of size exactly m. However, we see that this is not optimal as both conditioning methods can return a set of rows which are pruned at every iteration to roughly half the size and contains only the most important rows in that block. Both conditioning methods exhibit similar behavior and are bounded between both `Sample` and `Identity` methods. Therefore, both of the conditioning methods respect the theoretical bound and, crucially, return a summary which is sublinear in the space constraint and hence a significantly smaller fraction of the input size.

Results on Time Complexity. There are three time costs measured. The first is the update time taken to compute the local well-conditioned basis which is theoretically $O(md^2 + md^5 \log m)$ by Theorem 2.2. However, the two bases that we test are an orthonormal basis, computable in time $O(md^2)$ and the `SPC3` transform which takes time $O(\text{nnz}(B)\log m)$ for a block B with m rows and $\text{nnz}(B)$ non-zero entries. Figure 3a demonstrates that `SPC3` is faster than `Orth` on this data in practice but this small absolute difference becomes negligible over the entirety of the stream as seen in Figure 3c. The query time in Figure 3b is roughly proportional to the summary size in all instances but here the conditioning methods perform noticeably better due to the smaller summary size that is returned as discussed in the previous section. However, as seen in Figure 4c, (Supplementary Material, Appendix F) this disparity becomes hugely significant on higher dimensionality data due to the increased size summary retained by sampling, further justifying our approach of pruning rows at every stage. While `Identity` appears to have fast query time, this is due to the summary being smaller. Although it may seem that for smaller summaries more local bases need to be computed and this time could prohibitively increase over the stream, Figure 3c demonstrates that even using small blocks does not cause the *overall time* (to process the stream and pro-

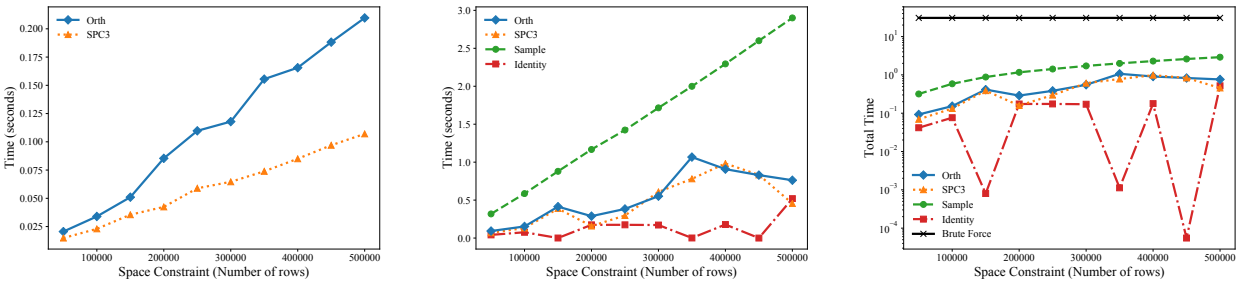
[2] `http://www.census.gov/census2000/PUMS5.html`

[3] `https://archive.ics.uci.edu/ml/datasets/yearpredictionmsd`



(a) U.S. Census (b) YearPredictionMSD (c) Max summary size vs space constraint

Figure 2: Error vs Space Constraint in (a) and (b) and Maximum Summary Size vs Space Constraint (c). Total input size is 5000000×11.



(a) Time to compute local basis (b) Solution time for ℓ_∞-regression (c) Total time

Figure 3: Computation Times compared to summary size

duce an approximate query) to increase too much. Hence, an approximation can be obtained which is highly accurate, and in total time faster than the brute force solver.

Experimental Summary. While it might seem attractive not to perform any conditioning on the matrix and just pick heavy rows, our experiments show that this strategy is not effective in practice, and delivers poor accuracy. Although a simple sample of randomly chosen rows can be easily maintained, this appears less useful due to the increased time costs associated with larger summaries when conditioning methods output a similar estimate in less time over the entire stream. As the ℓ_∞-regression problems depend only on a few rows of the data there are cases when uniform sampling can perform well: if many of the critical rows look similar then there is a chance that uniform sampling will select some examples. In this case, the leverage of the important direction is divided across the repetitions, and so it is harder to ensure that desired direction is identified. Despite this potential drawback we have shown that both `Orth` and `SPC3` can be used to find accurate summaries which perform robustly across each of the measures we have tested. It appears that `SPC3` performs comparably to `Orth`; both are relatively quick to compute and admit accurate summaries in similar space. In particular, both conditioning methods return summaries which are a fraction of the space budget and hence highly sublinear in the input size, which give accurate approximations and are robust to the concatenation of new rows. All of these factors make the conditioning method fast in practice to both find the important rows in the data and then compute the reduced regression problem with high accuracy.

Due to the problems in constructing summaries which can be used to solve regression quickly and accurately when using random sampling or no transformation, our methods are shown to be efficient and accurate alternatives. Our approach is vindicated both theoretically and practically: this is most clear in the U.S. Census dataset where small error can be achieved using a summary roughly 2% the size of the data. This also results in an overall speedup as solving the optimization on the reduced set is much faster than solving on the full problem. Such significant savings show that this general approach can be useful in large-scale applications.

Acknowledgements

The work of G. Cormode and C. Dickens is supported by European Research Council grant ERC-2014-CoG 647557 and The Alan Turing Institute under the EPSRC grant EP/N510129/1. D. Woodruff would like to acknowledge the support by the National Science Foundation under Grant No. CCF-1815840.

References

Agarwal, Pankaj, Cormode, Graham, Huang, Zengfeng, Phillips, Jeff, Wei, Zheiwei, and Yi, Ke. Mergeable summaries. In *ACM Principles of Database Systems*, 2012.

Clarkson, K. L., Drineas, P., Magdon-Ismail, M., Mahoney, M. W., Meng, X., and Woodruff, D. P. The fast cauchy transform and faster robust linear regression. In *Proc. of the 24-th Annual SODA*, pp. 466–477, 2013.

Cohen, Michael B and Peng, Richard. ℓ_p row sampling by lewis weights. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pp. 183–192. ACM, 2015.

Dasgupta, A., Drineas, P., Harb, B., Kumar, R., and Mahoney, M. W. Sampling algorithms and coresets for l_p regression. In *Proc. of the 19th Annual SODA*, pp. 932–941, 2008.

Feldman, Jon, Muthukrishnan, S., Sidiropoulos, Anastasios, Stein, Cliff, and Svitkina, Zoya. On the complexity of processing massive, unordered, distributed data. Technical Report CoRR abs/cs/0611108, ArXiV, 2006.

Ghashami, M., Liberty, E., and Phillips, J. M. Efficient Frequent Directions Algorithm for Sparse Matrices. *ArXiv e-prints*, 2016.

Ghashami, Mina and Phillips, Jeff M. Relative errors for deterministic low-rank matrix approximations. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pp. 707–717, 2014.

Ghashami, Mina, Liberty, Edo, Phillips, Jeff M., and Woodruff, David P. Frequent directions: Simple and deterministic matrix sketching. *SIAM J. Comput.*, 45(5): 1762–1792, 2016.

Hand, Michael Lawrence. Aspects of linear regression estimation under the criterion of minimizing the maximum absolute residual. 1978.

Hardt, Moritz and Woodruff, David P. How robust are linear sketches to adaptive inputs? In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pp. 121–130, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2029-0. doi: 10.1145/2488608.2488624. URL http://doi.acm.org/10.1145/2488608.2488624.

Kremer, Ilan, Nisan, Noam, and Ron, Dana. On randomized one-round communication complexity. *Computational Complexity*, 8(1):21–49, 1999.

Kushilevitz, E. and Nisan, N. *Communication Complexity*. Cambridge University Press, 1997.

Liberty, Edo. Simple and deterministic matrix sketching. In *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*, pp. 581–588, 2013.

Meng, Xiangrui and Mahoney, Michael W. Low-distortion subspace embeddings in input-sparsity time and applications to robust linear regression. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pp. 91–100, 2013.

Shen, Fumin, Shen, Chunhua, Hill, Rhys, van den Hengel, Anton, and Tang, Zhenmin. Fast approximate l_∞ minimization: speeding up robust regression. *Computational Statistics & Data Analysis*, 77:25–37, 2014.

Song, Z., Woodruff, D., and Zhong, P. Low rank approximation with entrywise ℓ_1-norm error. In *STOC*, 2017.

Sposito, VA. Minimizing the maximum absolute deviation. *ACM SIGMAP Bulletin*, (20):51–53, 1976.

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

Woodruff, David and Zhang, Qin. Subspace embeddings and l_p-regression using exponential random variables. *JMLR: Workshop and Conference Proceedings*, 30:1–22, 2013.

Yang, J., Meng, X., and Mahoney, M. W. Quantile regression for large-scale applications. *Proc. of the 30th ICML Conference, JMLR W&CP*, 28(3):881–887, 2013.

Supplementary Material for Leveraging Well-Conditioned Bases: Streaming and Distributed Summaries in Minkowski p-Norms

A. Proofs for Section 3

Lemma A.1. Denote the ith global leverage score of A by w_i and its associated local leverage score in a block of input A be denoted $\hat{w}_k$. Then $w_i/\text{poly}(d) \leq \hat{w}_k$. In particular, $w_i/(d\alpha^p\beta) \leq \hat{w}_k$.

Proof of Lemma A.1. Let $U = AR$. Recall that $w_i = \|\mathbf{e}_i^T AR\|_p^p$. Then for some coordinate j we must have $|\mathbf{e}_i^T AR\mathbf{e}_j|^p \geq w_i/d$. Taking $\mathbf{x} = \mathbf{e}_j$ we see that

$$|(AR\mathbf{x})_i|^p \geq \frac{w_i}{d}. \tag{2}$$

However, Fact 1 implies :

$$\|AR\mathbf{x}\|_p^p \leq \|AR\|_p^p \leq \alpha^p \leq \text{poly}(d). \tag{3}$$

Hence, there exists a $\mathbf{y} \in \text{col}(A)$ with $\mathbf{y} = AR\mathbf{x}$ such that $|\mathbf{y}_i|^p \geq w_i/d$ from Equation (2). Also, $\|\mathbf{y}\|_p^p \leq \alpha^p$ from Equation (3). Thus,

$$\frac{|\mathbf{y}_i|^p}{\|\mathbf{y}\|_p^p} \geq \frac{|\mathbf{y}_i|^p}{\alpha^p} \tag{4}$$

$$\geq \frac{w_i}{d\alpha^p} \tag{5}$$

$$\geq \frac{w_i}{\text{poly}(d)}. \tag{6}$$

From this we see;

$$w_i \leq \frac{|\mathbf{y}_i|^p d\alpha^p}{\|\mathbf{y}\|_p^p}. \tag{7}$$

Now, let B be a block of rows from A. We manipulate B by considering it either as an individual matrix or as a coordinate subspace of A; i.e all rows are zero except for those contained in B which will be denoted by $\hat{A}$. Define $\hat{\mathbf{y}} = \hat{A}R\mathbf{x}$. Then $\hat{\mathbf{y}}_{j'} = \mathbf{y}_{j'}$ when j' is a row from B and $\hat{\mathbf{y}}_{j'} = 0$ otherwise. Thus, $\|\hat{\mathbf{y}}\|_p^p \leq \|\mathbf{y}\|_p^p$ and:

$$w_i \leq \frac{|\mathbf{y}_i|^p d\alpha^p}{\|\hat{\mathbf{y}}\|_p^p}. \tag{8}$$

For rows i which are also found in B (indexed as k) we see that $|\hat{\mathbf{y}}_k|^p = |\mathbf{y}_i|^p$. So, *for such indices*, using Equations (7) and (8):

$$w_i \leq \frac{|\hat{\mathbf{y}}_k|^p d\alpha^p}{\|\hat{\mathbf{y}}\|_p^p}. \tag{9}$$

Since $\hat{\mathbf{y}}$ is the restriction of $\mathbf{y}$ to coordinates of B we can write $\hat{\mathbf{y}} = B\hat{R}\hat{x}$ where $B\hat{R}$ is well-conditioned. Let $\hat{w}_k = \|\mathbf{e}_k^T B\hat{R}\|_p^p$ be the kth local leverage score in B. By applying the same argument as in Fact 2 it can be shown that $|\hat{\mathbf{y}}_k|^p/\|\hat{\mathbf{y}}\|_p^p \leq \text{poly}(d)\hat{w}_k$. Indeed,

$$|\hat{\mathbf{y}}_k|^p = |(\hat{A}\hat{R}\hat{x})_k|^p \tag{10}$$

$$\leq \|e_k^T \hat{A}\hat{R}\|_p^p \|\hat{x}\|_q^p \quad \text{by Hölder's inequality} \tag{11}$$

$$\leq \hat{w}_k\beta\|B\hat{R}\hat{x}\|_p^p \tag{12}$$

$$\leq \beta\hat{w}_k\|\hat{\mathbf{y}}\|_p^p. \tag{13}$$

The second inequality uses condition 2 from Theorem 2.1 and the fact that $B\hat{R}$ is a well-conditioned basis. Then using Equation 9, the following then proves the latter claim of the lemma:

$$\frac{w_i}{d\alpha^p} \leq \frac{|\hat{\mathbf{y}}_k|^p}{\|\hat{\mathbf{y}}\|_p^p} \leq \beta\hat{w}_k.$$

Finally, Theorem 2.2 states that β is at most $\text{poly}(d)$ which proves the result. □

Lemma A.2. All global leverage scores above a threshold can be found by computing local leverage scores and increasing the space complexity by a $\text{poly}(d)$ factor.

Proof of Lemma A.2. First we determine the space necessary to find all leverage scores exceeding δ. Let $I = \{i : w_i > \delta\}$. Then $\alpha^p \geq \sum_{i=1}^n w_i \geq \sum_{i\in I} w_i \geq \delta|I|$ by arguing as in Fact 1 Hence, the space necessary is $|I| \leq \alpha^p/\delta$. Now focus on finding these rows in the streaming fashion. By Lemma A.1 we see that for rows k in the block which is stored from the stream we have the property that $w_i/d\alpha^p\beta \leq \hat{w}_k$. Hence, any $w_i > \delta$ results in $\hat{w}_k > \delta/d\alpha^p\beta$ for the local thresholding. So to keep all such $w_i > \delta$, we must store all $\hat{w}_k > \delta/d\alpha^p\beta = \hat{\delta}$. Arguing similarly as in Fact 1 again define $\hat{I} = \{k : \hat{w}_k > \hat{\delta}\}$ so that: $\alpha^p \geq \sum_k \hat{w}_k \geq \sum_{k\in\hat{I}} \hat{w}_k \geq \hat{\delta}|\hat{I}|$. Hence $|\hat{I}| \leq \alpha^p/\hat{\delta} = d\alpha^{2p}\beta/\delta$. That is, $|\hat{I}| \leq d\beta\alpha^p \cdot |I|$ which proves the claim as Theorem 2.2 states that all of the parameters are $\text{poly}(d)$. □

Proof of Theorem 3.3. We claim that the output of Algorithm 1 is a matrix B which contains rows of high leverage in A. The algorithm initially reads in b rows and inserts these to matrix A'. A well-conditioned basis U for A' is then computed using Theorem 2.2 and incurs the associated $O(bd^2 + bd^5 \log b)$ time. The matrix U and A' are passed to Algorithm 2 whereby if a row i in U has local leverage exceeding τ then row i of A' is kept. There are at most $\text{poly}(d)/\tau$ of these rows as seen in Lemma A.2 and the space required is $\text{poly}(d)$ by the same lemma. So on the first call to Algorithm 2 a matrix is returned with rows

whose ℓ_p local leverage satisfies $w_i/\text{poly}(d) \leq \hat{w}_i$ (where w_i is the global leverage score and $\hat{w}_i$ is the associated local leverage score) and only those exceeding $\tau/\text{poly}(d)$ are kept.

The algorithm proceeds by repeating this process on a new set of rows from A and an improved matrix B which contains high leverage rows from A already found. Proceeding inductively, we see that when Algorithm 2 is called with matrix $[A'; B]$ then a well-conditioned basis U is computed. Again $[A'; B]_i$ is kept if and only if the local leverage score from U, $w_i(U) > \tau$. By Lemma A.2 this requires $\text{poly}(d)$ space and the local leverage score is at least a $1/\text{poly}(d)$ factor as large as the global leverage score by Lemma A.1. Repeating over all blocks B in A, only the rows of high leverage are kept. Any row of leverage smaller than $\tau/\text{poly}(d)$ is ignored so this is the additive error incurred. □

B. Proofs for Section 4

Algorithm and Discussion

The pseudocode for the first level of the tree structure of the deterministic ℓ_p subspace embedding described in Section 4 is given in Algorithm 3. We use the following notation: m is a counter to index the block of input currently held, denoted $A_{[m]}$, and ranges from 1 to $n^{1-\gamma}$ for the first level of the tree. Similarly, t indexes the current summary, $P^{(t)}$ which are all initialized to be an empty matrix. Again we use the notation $[X; Y]$ to denote the row-wise concatenation of two matrices X and Y with equal column dimension.

Note that Algorithm 3 can be easily distributed as any block of sublinear size can be given to a compute node and then a small-space summary of that block is returned to continue the computation. In addition, the algorithm can be performed using sublinear space in the streaming model because at any one time a summary T of the input can be computed which is of size $d \times d$. Upon reading $A_{[1]}$, a small space summary $P^{(1)}$ is computed and stored with the algorithm proceeding to read in $A_{[2]}$. Similarly, the summary $P^{(2)}$ is computed and if $[P^{(1)}; P^{(1)}]$ does not exceed the storage bound, then the two summaries are merged and this process is repeated until at some point the storage bound is met. Once the summary is large enough that it meets the storage bound, it is then *reduced* by performing the well-conditioned basis reduction (line (5)) and the reduced summary is stored with the algorithm continuing to read and summarize input until a corresponding block in the tree is obtained (or the blocks can be combined to terminate the algorithm).

Proof of Theorem 4.2. Let $A \in \mathbb{R}^{n \times d}$ and $B \in \mathbb{R}^{n^\gamma \times d}$. We compute an ℓ_p well-conditioned basis for B in time $\text{poly}(n^\gamma d)$ by Theorem 2.2; so let $B = US$ for $U \in \mathbb{R}^{n^\gamma \times d}$ and $S \in \mathbb{R}^{d \times d}$ a change of basis matrix.

Algorithm 3 Deterministic ℓ_p subspace embedding

1: **procedure** ℓ_p-SUBSPACEEMBEDDING($A, p, \gamma < 1$))
2: Counters $m, t \leftarrow 1$
3: Summaries $P^{(t)} \leftarrow$ EMPTY for all t.
4: **for** $m = 1 : n^{1-\gamma}$ **do**
5: $A_{[m]} = US$ # U an ℓ_p wcb for A
6: **if** num. rows$(P^{(t)}) + d \leq n^\gamma$ **then**
7: $P^{(t)} \leftarrow [P^{(t)}; S]$
8: **else**
9: $P^{(t+1)} \leftarrow S$
10: $t \leftarrow t + 1$
11: *Merge* all $P^{(t)}$: $T = [P^{(1)}; \ldots; P^{(\cdot)}]$
12: *Reduce* T by splitting into blocks of n^γ and repeating lines (2) - (10) with T in place of A.
13: **return** T

From (Dasgupta et al., 2008), U satisfies $\|x\|_p \leq \|Ux\|_p \leq d\|x\|_p$. This is because $\|x\|_2 \leq \|Ux\|_p \leq \sqrt{d}\|x\|_2$. There are then two cases: if $p < 2$ then

$$\frac{\|x\|_p}{\sqrt{d}} \leq \|x\|_2 \leq \|Ux\|_p \leq \sqrt{d}\|x\|_2 \leq \sqrt{d}\|x\|_p$$

so that $\|x\|_p \leq \|Ux\|_p \leq d\|x\|_p$ by rescaling by $\sqrt{d}$. The third inequality is from (Dasgupta et al., 2008). Similarly, if $p > 2$ then

$$\|x\|_p \leq \|x\|_2 \leq \|Ux\|_p \leq \sqrt{d}\|x\|_2 \leq d\|x\|_p$$

from which $\|x\|_p \leq \|Ux\|_p \leq d\|x\|_p$. Next, the algorithm ignores U and retains only S after computing the well-conditioned basis. Using the above two bounds we readily see that $\|Sx\|_p \leq \|USx\|_p = \|Bx\|_p$. Also, $\|Sx\|_p \geq \|USx\|_p/d = \|Bx\|_p/d$. Now we have obtained a matrix S which satisfies:

$$\frac{\|Bx\|_p}{d} \leq \|Sx\|_p \leq \|Bx\|_p. \tag{14}$$

So $\|Sx\|_p$ agrees with $\|Bx\|_p$ up to a distortion factor of d.

Algorithm 3 applies the *merge and reduce* framework. The matrix A is seen a row at a time and n^γ rows are stored which are used to construct a tree. So at every level a subspace embedding with distortion d is constructed. This error propagates through each of the $O(1/\gamma)$ levels in the tree so the overall distortion to construct the subspace embedding for A is $d^{O(1/\gamma)}$. The space bound is similar; we need $n^\gamma d$ storage per group so require $O(1/\gamma)n^\gamma d$ overall. □

Proof of Theorem 4.3. The task is to minimise $\|Ax - b\|_p$.

Let $Z = [A, b] \in \mathbb{R}^{n\times(d+1)}$ and compute a subspace embedding S for Z using Theorem 4.2. Note that R has $O(1/\gamma)n^\gamma(d+1)$ rows. Let $\Delta = (d+1)^{O(1/\gamma)}$, then for all $y \in \mathbb{R}^{d+1}$ we have:

$$\frac{\|Zy\|_p}{\Delta} \leq \|Sy\| \leq \|Zy\|. \tag{15}$$

Since this condition holds for all $y \in \mathbb{R}^{d+1}$ it must hold, in particular, for vectors $y' = (x, -1)^T$ where $x \in \mathbb{R}^d$ is arbitrary. However, observe that:

$$\|Zy'\|_p = \left\|[A, b]\begin{bmatrix} x \\ -1 \end{bmatrix}\right\|_p = \|Ax - b\|_p. \tag{16}$$

Denote the first d columns of S by $S_{1:d}$ and the last column by S_{d+1}. Then

$$\|Sy'\|_p = \left\|[S_{1:d}, S_{d+1}]\begin{bmatrix} x \\ -1 \end{bmatrix}\right\|_p = \|S_{1:d}x - S_{d+1}\|_p. \tag{17}$$

Now we have transformed the subspace embedding relationship into an instance of regression. In particular, $S_{1:d}$ has only $O(1/\gamma)n^\gamma d$ rows so is a smaller instance than the original problem. We now focus on the task of finding $\min_{x\in\mathbb{R}^d} \|S_{1:d}x - S_{d+1}\|_p$. By using Equation (15) with y' and utilising Equations $(16), (17)$ we have:

$$\frac{\|Ax - b\|_p}{\Delta} \leq \|S_{1:d}x - S_{d+1}\|_p \leq \|Ax - b\|_p. \tag{18}$$

Convex optimisation can now be used to find $\min_{x\in\mathbb{R}^d} \|S_{1:d}x - S_{d+1}\|_p$. Let $\hat{x} = \operatorname{argmin}_{x\in\mathbb{R}^d} \|S_{1:d}x - S_{d+1}\|_p$ which is output from the optimisation and let $x^* = \operatorname{argmin}_{x\in\mathbb{R}^d} \|Ax - b\|_p$ be the optimal solution we would like to estimate. By optimality of $\hat{x}$ we have:

$$\|S_{1:d}\hat{x} - S_{d+1}\|_p \leq \|S_{1:d}x^* - S_{d+1}\|_p. \tag{19}$$

However, combining Equation (19) with Equation (18) we see that:

$$\begin{aligned} \frac{\|A\hat{x} - b\|_p}{\Delta} &\leq \|S_{1:d}\hat{x} - S_{d+1}\|_p && (20)\\ &\leq \|S_{1:d}x^* - S_{d+1}\|_p && (21)\\ &\leq \|Ax^* - b\|_p && (22) \end{aligned}$$

Therefore, $\|A\hat{x} - b\|_p \leq \Delta\|Ax^* - b\|_p$ and $\Delta = \text{poly}(d+1)$ so the ℓ_p-regression problem has been solved up to a polynomial $d+1$ approximation factor. The overall time complexity is the time taken to compute the subspace embedding, which is poly(nd) by Theorem 4.2, and the time for the convex optimisation. However, the optimisation costs poly$(O(1/\gamma)n^\gamma)$ (Woodruff & Zhang, 2013) which is subsumed by the dominant time cost for computing the embedding. Finally, the space cost is immediate from computing the subspace embedding in Theorem 4.2. □

Algorithm 4 Deterministic ℓ_1 low rank approximation (derandomized version of algorithm from (Song et al., 2017))

procedure L1-KRANKAPPROX(X, n, d, k)
  $r = O(k \log k)$
  $m = O(r \log r)$
  $t_1 = O(r \log r)$
  $t_2 = O(m \log m)$
  Generate all diagonal $R \in \mathbb{R}^{d\times d}$ with only r 1s
  Compute all possible sampling and rescaling matrices $D, T_1 \in \mathbb{R}^{n\times n}$ corresponding to Lewis Weights of AR whose entries are powers of 2 between 1 and $1/nd$. There are m and t_1 nonzero entries on the diagonal, respectively.
  Compute all sampling and rescaling matrices $T_2^T \in \mathbb{R}^{d\times d}$ according to the Lewis weights of $(DA)^T$ with t_2 nonzero entries, powers of 2 between 1 and $1/nd$ on the diagonal.
  Evaluate $\|T_1ARXYDAT_2 - T_1AT_2\|_1$ for all choices of above matrices.
  Take the minimal solution
  return ARX, YDA

C. Proofs for Section 5

To prove correctness of Algorithm 5 for Theorem 5.1 we will need to invoke the following algorithm at each level of the tree. This is a derandomized version of an algorithm which returns a low rank approximation to an input matrix. The derandomization follows from generating and testing all possible combinations of the necessary matrices.

Lemma C.1. Algorithm 4 runs in time poly(nd).

Proof. Every matrix which is generated in Algorithm 4 has a number of nonzero entries bounded by $O(k\text{polylog}(k))$. We can test all of the matrices which will take time proportional to the dimension of the matrix (n or d) with exponent $O(k\text{polylog}(k))$ resulting in time poly(nd) overall, since k is constant. □

We need one further lemma which describes the approximation error induced by using well-conditioned bases to decompose a matrix.

Lemma C.2. Let $M \in \mathbb{R}^{N\times D}$ have rank ρ and suppose $U \in \mathbb{R}^{N\times\rho}$ is a well-conditioned basis for M. Let $M = US$ for a change of basis $S \in \mathbb{R}^{\rho\times D}$. Then for all $x \in \mathbb{R}^D$:

$$\frac{\|Sx\|_1}{\text{poly}(D)} \leq \|Mx\|_1 \leq \text{poly}(D)\|Sx\|_1.$$

Proof. For the left-hand side we can just calculate:

$$\begin{aligned}\|Sx\|_1 &\leq D\cdot\|Sx\|_\infty &(23)\\ &\leq D\cdot\text{poly}(D)\|USx\|_1 &(24)\\ &= \text{poly}(D)\cdot\|Mx\|_1. &(25)\end{aligned}$$

The second inequality follows from a property of the well-conditioned basis U. The result follows from observing:

$$\begin{aligned}\|Mx\|_1 = \|USx\|_1 &\leq \|U\|_1\|Sx\|_\infty &(26)\\ &= \text{poly}(D)\|Sx\|_1 &(27)\end{aligned}$$

□

C.1. Proof of Theorem 5.1

For the proof of Theorem 5.1 we introduce Algorithm 5. It is enough to show that for every level, the low rank approximations of each group is polynomially bounded by k in error. The result follows by reasoning how this error grows as we progress through the tree. Denote the jth block of A by $A_{[j]}$.

Algorithm 5 Deterministic ℓ_1 low rank approx

procedure ℓ_1-k-RANKAPPROX(A, k, γ)
  $m, t \leftarrow 1, P_t \leftarrow 0$
  for $i = 1 : 1/\gamma$ **do**
    while $m < n^{1-i\gamma}$ **do**
      while number of rows of $P_m < n^\gamma$ **do**
        Run Algorithm 4 on $A_{[m]}$ and k which outputs matrix $B \in \mathbb{R}^{k\times d}$
        $B \leftarrow WV^T$ (k-rank decomposition)
        Set $W = US$ for well-conditioned basis U
        $P_t \leftarrow SV^T$
        $m \leftarrow m + 1$
  Merge-and-Reduce all P_m until we have an $n^\gamma \times d$ matrix.
  Set P to be matrix of final k rows.
  Solve $\min_Q \|QP - A\|_1$.
  return QP

For every level in the tree we can take a group of rows, C, and perform Algorithm 4. For every C used as input to Algorithm 4 a k-rank matrix B of dimensions $n^\gamma \times d$ is returned. In particular, B has the following property:

$$\|C - B\|_1 \leq \text{poly}(k)\min_{B' \text{rank} k}\|C - B'\|_1. \quad (28)$$

Now factor B using a k rank decomposition. That is, set $B = WV^T$ where W has k columns and V^T has k rows. Further decompose W as $W = US$ for a well-conditioned basis U. Note that W is $n^\gamma \times k$ (and of rank k) by the rank decomposition so U is also $n^\gamma \times k$ and S is $k \times k$. The dimensions of these matrices ensure that individually they do not exceed the space budget from the theorem.

Apply Lemma C.2 with W and k. Then we have for every $x \in \mathbb{R}^k$ that $\|Sx\|_1 = \text{poly}(k)\|Wx\|_1$. Since U is n^γ by k and $k < \text{poly}(d)$, U remains within the required space bound when we use it for the calculation. Now ignore U and store SV^T. Note that each SV^T is a matrix of k directions in $\mathbb{R}^d$. Pass SV^T to the next level of the tree.

Merge the SV^T for each group until we have a matrix of n^γ rows. Repeat the process over all $O(1/\gamma)$ levels in the tree. We require $n^\gamma d$ storage for every group so as we merge and pass SV^T down the levels this combines to total storage of $O(1/\gamma)n^\gamma\text{poly}(d)$. This part of the algorithm is a repeated use of Algorithm 4 which is poly(nd) by Lemma C.1 and some further lower time cost manipulations. Repeating these steps gives poly(nd) as the overall time complexity.

When this is done over all levels we will again have k directions in $\mathbb{R}^d$. Let P be the matrix with these directions as rows. Then we claim that P can be used to construct our approximate ℓ_1 low-rank approximation.

Proposition C.3. Let P be as described above. Then there exists QP which is an ℓ_1 low-rank approximation for A:

$$\min_Q \|QP - A\|_1 \leq \text{poly}(k)\|A - A'\|_1$$

Proof. Each use of Algorithm 4 admits a poly(k) approximation at every level of the tree. Every time the well-conditioned basis U is constructed and then ignored we admit a further poly(k) error due to property 1 of Definition 2.1. The distortion is blown up by a factor of poly(k) every time we use Lemma C.2 which is at every level in the tree. Hence, the total contribution of using Algorithm 4 is $\text{poly}(k)^{O(1/\gamma)} = \text{poly}(k)$ for constant γ. □

Proposition C.3 proves the approximation is poly(k) as claimed. By Lemma C.1 we know that Algorithm 4 is poly(nd) time. The most costly steps in Algorithm 5 are invocations of Algorithm 4 so combining this we see that the overall time cost is poly(nd) as claimed, proving the theorem.

D. Proofs for Section 6

Proof of Theorem 6.1. Given A, the first step is to store all rows of A whose ℓ_p leverage score is above the threshold $\varepsilon/\text{poly}(d)$. This step requires a polynomial increase to $\text{poly}(d)/\varepsilon$ storage from Lemma A.2. Next the change of basis matrix R is computed so that AR is well-conditioned. The stored matrix is B with rows corresponding to those of large ℓ_p leverage scores from A' and zero elsewhere. Also, store all entries in b whose magnitude is greater than $\varepsilon\|b\|_p$ and zero the rest out. Call this vector b'.

We now focus on the task of solving $\min_{x\in\mathbb{R}^d}\|A'Rx - b'\|_\infty$. Any solution must necessarily have $\|x\|_p \leq \text{poly}(d)\|b\|_p$ as otherwise $x = \mathbf{0}$ is a better solution. Recall that $\alpha = d^{1/p+1/2}$ for a well-conditioned basis AR with $p > 2$. Hence, the sum of all the ℓ_p leverage scores is $\alpha^p = d^{O(p)}$. Then the number of rows with leverage score greater than the $\varepsilon/\text{poly}(d)$ is at most $\text{poly}(d)/\varepsilon \cdot d^{O(p)} = \text{poly}(d)/\varepsilon$ for a constant p.

Now, take any row for which the ℓ_p leverage score is less than the $\varepsilon/\text{poly}(d)$ threshold. Then:

$$\begin{aligned}|\langle (AR)_i, x\rangle| &\leq \|(AR)_i\|_\infty\|x\|_1\\ &\leq \|(AR)_i\|_p\|x\|_1\\ &\leq \|(AR)_i\|_p \cdot d\|x\|_p\\ &\leq d\frac{\varepsilon}{\text{poly}(d)}\text{poly}(d)\|b\|_p.\end{aligned}$$

By an appropriate choice of the $\text{poly}(d)$ factors scaling ε we see that $|\langle (AR)_i, x\rangle| \leq \varepsilon\|b\|_p$. On such coordinates the ℓ_∞ cost is $|b_i| \pm \varepsilon\|b\|_p$ so by replacing the row with one which is all zero we still pay $|b_i|$ which is within the $\varepsilon\|b\|_p$ had we included the row. The remaining high-leverage score rows are stored in their entirety so the cost on these rows is the same as in the original regression problem. □

Proof of Theorem 6.2. Let S be a set of $2^{\Omega(d)}$ strings in $\{0,1\}^d$ with each coordinate in a string uniformly sampled randomly from $\{0,1\}$. Let $x, y \in S$ and fix a constant $0 < c < 1$. By a Chernoff bound it follows that there are at least cd coordinates in $[d]$ for which $x_i = 0$ and $y_i = 1$ with probability $1 - 2^{-\Omega(d)}$. This implies for appropriate constants in the $\Omega(\cdot)$, by a union bound, all pairs of strings $x, y \in S$ have this property. Hence, such an S exists and we will fix this for the proof.

The regression problem can be reduced to an instance of the `Indexing` problem (Kremer et al., 1999) in data streams as follows. In the stream, the vector b will be all 1s. We will see a random subset T of some elements from S. We claim that it is possible to decide which case we are in: given a random string y, whether y is in S independent of T, or y is in T. This corresponds to solving `Indexing` which requires space $\Omega(|S|) = \Omega(\min\{n, 2^{\Omega(d)}\})$ even with randomization, via communication complexity arguments (Kushilevitz & Nisan, 1997).

Given a test vector y, negate its coordinates so that $y \in \{0,-1\}^d$. Now, append y as a row to the final b coordinate of 1 at the end of the stream to obtain the last item in the stream $(y, 1)$. If y were in S then both y and its complement would be seen as rows of the matrix A. Hence, the optimal cost for ℓ_∞-regression is at least 1. Otherwise, y is not in S. Consider the set of coordinates R where $y_i = 0$. Set $x_i = 1/d$ for $i \in R$ and $-c/2d$ otherwise.

Now we consider the cost of using x. On the row corresponding to the negated vector y the value will be at least $(-1)(-c/2d)(cd) = c^2/2$. Since $b_i = 1$ the cost will be at most $|1 - c^2/2|$ for this coordinate. On all other rows, by using the fact there are at least cd occurrences of $x_i = 0, y_i = 1$ the value will be at least

$$cd(1/d) - (d - cd)(c/2d) \geq c - c/2 = c/2.$$

Hence the cost on these coordinates is at most $|1 - c/2|$. Since $c < 1$, the ℓ_∞ cost is at most $|1 - c^2/2|$. This is a constant factor less than the ℓ_∞ cost of 1 from the previous case so it is possible to decide which of the two cases we are in and hence the space is $\Omega(\min\{n, 2^{\Omega(d)}\})$ as claimed. □

E. Deterministic Approximate Matrix Multiplication

Despite the generality of the subspace embedding result in Theorem 4.2, there may be occasions where the overheads are sufficiently large that it does not make sense to employ this method. One such example is for the *matrix multiplication* problem. Let $A, B \in \mathbb{R}^{n\times d}$ and consider the task of finding a matrix C for which $\|A^TB - C\|_1 < \varepsilon\|A\|_1\|B\|_1$ where $0 < \varepsilon < 1$ and the norm is *entrywise* 1*-norm*.

Lemma E.1. Let $x, y \in \mathbb{R}^n$ have unit entrywise 1-norm. Let $\varepsilon > 0$. Define:

$$\bar{x}_i = \begin{cases} x_i & \text{if } |x_i| > \varepsilon/2,\\ 0 & \text{otherwise,}\end{cases} \qquad \bar{y}_i = \begin{cases} y_i & \text{if } |y_i| > \varepsilon/2,\\ 0 & \text{otherwise.}\end{cases}$$

Then $\langle x, y\rangle - \varepsilon \leq \langle \bar{x}, \bar{y}\rangle \leq \langle x, y\rangle$ and this can be computed using space $O(1/\varepsilon)$.

Proof. Observe that $\bar{x}_i \leq x_i$ and $\bar{y}_i \leq y_i$ for $1 \leq i \leq n$. Hence, $\langle \bar{x}, \bar{y}\rangle \leq \langle x, y\rangle$. For the left-hand side define the following sets: $H_u = \{i : u_i > \varepsilon/2\}, L_u = \{i : u_i \leq \varepsilon/2\}$

for $u = x, y$. Then we can write

$$\begin{aligned}\langle x, y\rangle &= \sum_{i\in H_x} x_i y_i + \sum_{i\in L_x} x_i y_i \\ &= \sum_{i\in H_x\cap H_y} x_i y_i + \sum_{i\in H_x\cap L_y} x_i y_i + \sum_{i\in L_x\cap H_y} x_i y_i + \sum_{i\in L_x\cap L_y} x_i y_i \\ &\leq \langle \bar{x}, \bar{y}\rangle + \frac{\varepsilon}{2}\sum_{i\in H_x} x_i + \frac{\varepsilon}{2}\sum_{i\in H_y} y_i + \frac{\varepsilon}{2}\sum_{i\in L_y} y_i \\ &\leq \langle \bar{x}, \bar{y}\rangle + \varepsilon.\end{aligned}$$

Note that the sum can be written this way as the pair H_x, L_x are disjoint, and likewise for H_y, L_y. The first inequality follows from the second line because $i \in H_x \cap H_y$ means x_i and y_i are retained in $\bar{x}, \bar{y}$ so this summation corresponds directly to $\langle \bar{x}, \bar{y}\rangle$. Then for every $i \in H_x \cap L_y$ we must have that $y_i \leq \varepsilon/2$ so is bounded by $\frac{\varepsilon}{2}\sum_{i\in H_x} x_i$. The same argument holds for the remaining two summations in the inequality. Finally, each of the three summations are at most 1 since both x and y have unit 1-norm. The summations over H_y and L_y when combined are at most the norm of y so can be combined such that $\sum_i y_i \leq 1$. This is enough to prove the result. □

The result for unit vectors is sufficient because we can simply normalize a vector, use Lemma E.1 and then rescale by the norm of x and y. This results in $\langle x, y\rangle - \varepsilon\|x\|_1\|y\|_1 \leq \langle \bar{x}, \bar{y}\rangle \leq \langle x, y\rangle$. This result can be used to prove the following theorem.

Theorem E.2. Let $A, B \in \mathbb{R}^{n\times d}$ and let $\varepsilon > 0$. Let A_i denote the ith row of A and B^i denote the ith column of B. For $X = A$ and $X = B$ define:

$$\overline{X}_{ij} = \begin{cases} X_{ij} & \text{if } |X_{ij}| > \frac{\varepsilon}{2}\|X_i\|_1, \\ 0 & \text{otherwise.}\end{cases}$$

Then in entrywise 1-norm:

$$\|AB^T - \overline{A}\overline{B}^T\|_1 \leq \varepsilon\|A\|_1\|B\|_1.$$

Proof. Fix $\varepsilon > 0$. The matrix product takes a row of A with a column of B^T which is simply a row of B. These are both vectors in $\mathbb{R}^d$ so we can apply the transformation as in the Theorem statement, which is equivalent to that in Lemma E.1. By applying the rescaled version of Lemma E.1 we see that:

$$|\langle A_i, B_j\rangle - \langle \overline{A}_i, \overline{B}_j\rangle \leq \varepsilon\|A_i\|_1\|B_j\|_1. \tag{29}$$

Now the norm $\|AB^T - \overline{A}\overline{B}^T\|_1$ is the sum of all summands defined as in Equation 29 over all pairs of i and j. Computing the sum then gives the desired result. □

The argument from Lemma E.1 can easily be adapted to obtain a result for the matrix profuct A^TB. Observe that approximating A^TB is equivalent to approximating inner products between columns of A and columns of B. The modification is that the summary must be applied column-wise instead of row-wise as in Theorem E.2.

Theorem E.3. Let $A, B \in \mathbb{R}^{n\times d}$ and let $\varepsilon > 0$. Then there exists a deterministic algorithm which uses $O(1/\varepsilon)$ space and outputs $\overline{A}$ and $\overline{B}$ which satisfy:

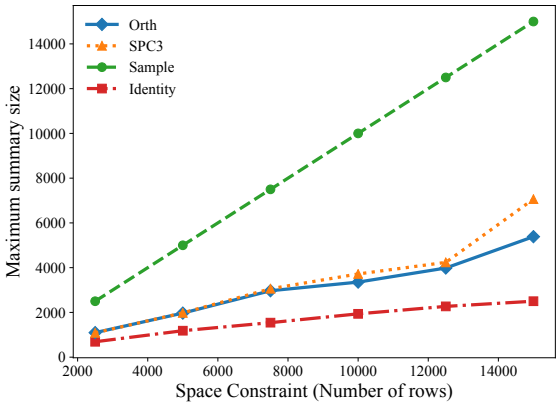
$$\|A^TB - \overline{A}^T\overline{B}\|_1 \leq \varepsilon\|A\|_1\|B\|_1.$$

Proof. For a matrix X let X_i denote the ith row and X^j denote the jth column. Let $\|X^j_{:i}\|_1$ denote the 1-norm of column j of X up to and including row i. It is clear that this norm is monotonic as more rows are seen in the stream. In particular, $\|X^j_{:n}\|_1 = \|X\|_1$. Therefore, the algorithm can be modified as follows: upon seeing a row i, if $|X_{ij}| > \varepsilon/2 \cdot \|X^j_{:i}\|_1$ then keep the entry X_{ij} and otherwise set $X_{oj} = 0$. It is sufficient to consider only the last row. At this stage all rows which have not exceeded the running threshold upon seeing a particular row will have been ignored and only those which exceed $\varepsilon/2 \cdot \|X^j_{:n-1}\|_1$ will be stored. Then by increasing the threshold upon seeing row n only the X_{ij} which exceed $\|X^j_{:n}\|_1 = \|X\|_1$ will be kept and this is exactly the same set of rows as had the summary been applied given full access to the rows.

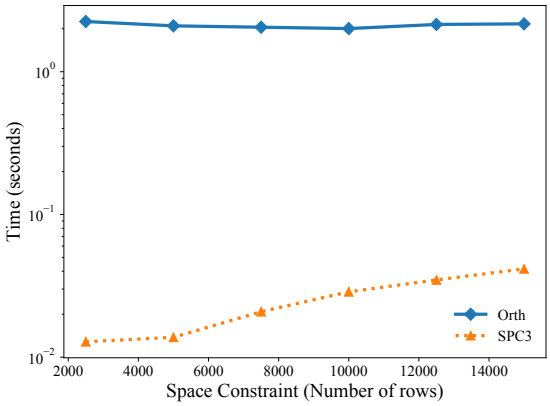
Hence, we may apply the result from Lemma E.1 on the columns of A and B as described above. It is then straightforward to show in a similar way to the lemma that the claim of the theorem holds. □

F. Further Experimental results

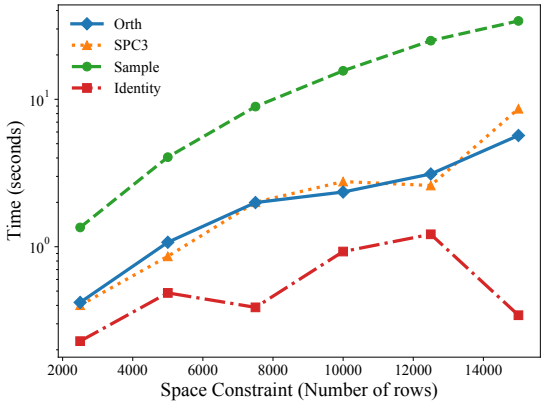
Here we illustrate the remaining experimental results on the YearPredictionMSD dataset which include the space and time plots. The experimental setup is the same as outline in Section 7.



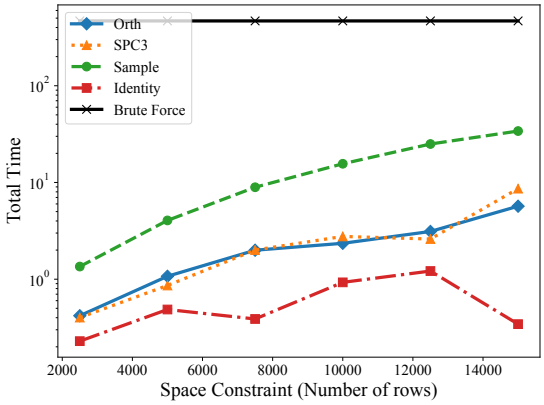
(a) Summary Size



(b) Update Time



(c) Query Time



(d) Total Time

Figure 4: Remaining plots for YearPredictionMSD data.