Bias-Aware Sketches

Jiecao Chen
Indiana University
Bloomington, IN 47405
jiecchen@umail.iu.edu

Qin Zhang
Indiana University
Bloomington, IN 47405
qzhangcs@indiana.edu

ABSTRACT

Linear sketching algorithms have been widely used for processing large-scale distributed and streaming datasets. Their popularity is largely due to the fact that linear sketches can be naturally composed in the distributed model and be efficiently updated in the streaming model. The errors of linear sketches are typically expressed in terms of the sum of coordinates of the input vector excluding those largest ones, or, the mass on the tail of the vector. Thus, the precondition for these algorithms to perform well is that the mass on the tail is small, which is, however, not always the case – in many real-world datasets the coordinates of the input vector have a *bias*, which will generate a large mass on the tail.

In this paper we propose linear sketches that are *bias-aware*. We rigorously prove that they achieve strictly better error guarantees than the corresponding existing sketches, and demonstrate their practicality and superiority via an extensive experimental evaluation on both real and synthetic datasets.

1. INTRODUCTION

Linear sketches, such as Count-Sketch [6] and Count-Median [11], are powerful tools for processing massive, distributed, and real-time datasets. Let $\mathbf{x} = (x_1, \ldots, x_n)^T$ be the input data vector where x_i stands for the frequency of element i. Linear sketching algorithms typically consist of two phases: (1) *Sketching phase.* We apply a linear sketching matrix $\Phi \in \mathbb{R}^{r \times n}$ ($r \ll n$) on $\mathbf{x}$, getting a sketching vector $\Phi\mathbf{x}$ whose dimension is much smaller than $\mathbf{x}$. (2) *Recovery phase.* We use $\Phi\mathbf{x}$ to recover useful information about the input vector $\mathbf{x}$, such as the median coordinate, the number of non-zero coordinates (distinct elements), etc.

Proceedings of the VLDB Endowment, Vol. 10, No. 9

We start by explaining why linear sketches are useful in handling distributed and streaming data. In the distributed computation model, we have t data vectors $\mathbf{x}^1, \ldots, \mathbf{x}^t$ distributed at t sites, which connect to a central coordinator. The goal is for the coordinator to learn the global data vector $\mathbf{x} = \sum_{i \in [t]} \mathbf{x}^i$ communication efficiently. Note that the naive solution that each site sending $\mathbf{x}^i$ to the coordinator is communication expensive if the dimension of $\mathbf{x}$ is large. By linearity we have $\Phi\mathbf{x} = \Phi\mathbf{x}^1 + \ldots + \Phi\mathbf{x}^t$. Thus each site can simply send the local sketching vector $\Phi\mathbf{x}^i$ to the coordinator, and then the coordinator sums up these local sketching vectors to obtain the global sketching vector $\Phi\mathbf{x}$, from which it reconstructs $\mathbf{x}$ using the recovery procedure. The total communication will be the product of t and the dimension of $\Phi\mathbf{x}$, which is much smaller than the dimension of input vector $\mathbf{x}$.

In the streaming model [1], where items arrive one by one in the online fashion, a new incoming item $i \in [n]$ corresponds to updating the input vector $\mathbf{x} \leftarrow \mathbf{x} + \mathbf{e}_i$ where $\mathbf{e}_i$ is an all-0 vector except the i-th coordinate being 1. Again due to linearity, we can easily update the linear sketch as $\Phi\mathbf{x} \leftarrow \Phi\mathbf{x} + \Phi\mathbf{e}_i$. The space usage of the streaming algorithm is simply the dimension of the sketch $\Phi\mathbf{x}$, which is again much smaller than the dimension of $\mathbf{x}$.

We consider in this paper the basic problem that in the recovery phase, we want to best reconstruct the input vector $\mathbf{x}$ using the sketching vector $\Phi\mathbf{x}$. More precisely, our goal is to design a sketching matrix Φ and a recovery procedure $\mathcal{R}(\cdot)$ with the following properties.

- *Accuracy.* $\hat{\mathbf{x}} = \mathcal{R}(\Phi\mathbf{x})$ is close to the original vector $\mathbf{x}$ under certain distance measurement.
- *Compactness.* The size of the sketch (equivalently, r, the number of rows of Φ) is small;
- *Efficiency.* We can compute $\Phi\mathbf{x}$ and $\hat{\mathbf{x}} = \mathcal{R}(\Phi\mathbf{x})$ time-efficiently.

This basic problem has many applications in massive data processing. Once a good approximation to $\mathbf{x}$ is obtained, we can answer a number of statistical queries on the input frequency vector such as *point query*, *frequent*

elements, *range query*, etc. These queries have numerous real-world applications, including Internet data analytics [10], search engines [24], data stream mining [9], streaming and distributed query processing [7, 8, 28], etc.

In this paper we focus on point query, which we believe is the most basic operation: given an index $i \in [n]$, return x_i (the i-th coordinate of the input vector $\mathbf{x}$). Naturally, we would like to minimize the maximum (average) coordinate-wise difference between the recovered vector $\hat{\mathbf{x}} = \mathcal{R}(\Phi\mathbf{x})$ and the original vector $\mathbf{x}$, that is, to minimize $\|\mathbf{x} - \hat{\mathbf{x}}\|_\infty$ ($\frac{1}{n}\|\mathbf{x} - \hat{\mathbf{x}}\|_1$).

Linear Sketches. Before stating our results, we would like to add some background on linear sketches. For a general vector $\mathbf{x} \in \mathbb{R}^n$, it is impossible to recover $\mathbf{x}$ *exactly* from the sketching vector $\Phi\mathbf{x}$ of a much smaller dimension. However, in many cases we are able to recover $\mathbf{x}$ up to some small errors. One such error guarantee, called the ℓ_∞/ℓ_p-guarantee, is that for any $\mathbf{x} \in \mathbb{R}^n$, letting $\hat{\mathbf{x}} = \mathcal{R}(\Phi\mathbf{x})$, the coordinate-wise error of the recovery is bounded by

$$\|\hat{\mathbf{x}} - \mathbf{x}\|_\infty = O(k^{-1/p}) \cdot \mathrm{Err}_p^k(\mathbf{x}), \tag{1}$$

where k is a tradeoff parameter between the sketch size and the accuracy guarantee, and

$$\mathrm{Err}_p^k(\mathbf{x}) = \min_{k\text{-sparse } \mathbf{x}'} \left\|\mathbf{x} - \mathbf{x}'\right\|_p,$$

where we say a vector is *k-sparse* if it contains *at most* k non-zero coordinates. In other words, $\mathrm{Err}_p^k(\mathbf{x})$ is the ℓ_p-norm of the vector containing all coordinates of $\mathbf{x}$ except zero-ing out the k coordinates with the largest absolute values. We often call the k largest coordinates the *head* of $\mathbf{x}$ and the rest $(n-k)$ ones the *tail* of $\mathbf{x}$. Note that if $\mathbf{x}$ is k-sparse, then we are able to recover it exactly since $\mathrm{Err}_p^k(\mathbf{x}) = 0$.

We typically consider $p = 1$ or $p = 2$, since for $p > 2$ there exists strong lower bound: the sketch size has to be at least $\Omega(n^{1-2/p})$. [1] The error guarantee in Equality (1) for $p = 1$ and $p = 2$ can be achieved with high probability by the classical Count-Median algorithm [11] and Count-Sketch algorithm [6] respectively; we will illustrate these two algorithms in details in Section 3.

It is folklore that ℓ_∞/ℓ_1 and ℓ_∞/ℓ_2 guarantees can be converted into ℓ_1/ℓ_1 and ℓ_2/ℓ_2 guarantees respectively (see, for example, Section II of [18]). More precisely, for $p \in \{1, 2\}$ we can derive from Inequality (1) that

$$\|\hat{\mathbf{x}} - \mathbf{x}\|_p = O(1) \cdot \mathrm{Err}_p^k(\mathbf{x}), \tag{2}$$

which gives a more intuitive approximation guarantee on the whole vector instead of individual coordinates.

Bias-Aware Sketches. The question we try to address in this paper is:

What if the coordinates in the input vector $\mathbf{x}$ have a non-trivial bias?

Let us consider an example. Let $k = 2$, $n = 10$, and

$$\mathbf{x} = (\mathbf{3}, 100, 101, \mathbf{500}, 102, 98, 97, 100, 99, 103). \tag{3}$$

We have $\mathrm{Err}_1^k(\mathbf{x}) = 700$, $\mathrm{Err}_2^k(\mathbf{x}) = \sqrt{69428} \approx 263.49$, which are fairly large. It is easy to see that these large errors are due to the fact that most coordinates of $\mathbf{x}$ are close to 100 (intuitively, the bias), which results in a heavy tail. It would be desirable if we can remove this bias first and then perform the sketching and recovery.

In this paper we propose bias-aware sketches that achieve the following performance guarantee. Let $\beta^{(n)}$ be the n-dimensional vector with β at each coordinate. For $p \in \{1, 2\}$, our sketches can recover an $\hat{\mathbf{x}}$ such that

$$\|\hat{\mathbf{x}} - \mathbf{x}\|_\infty = O(k^{-1/p}) \cdot \min_\beta \mathrm{Err}_p^k(\mathbf{x} - \beta^{(n)}). \tag{4}$$

And we define the **bias** of the input data vector $\mathbf{x}$ to be

$$\beta^* = \arg\min_\beta \mathrm{Err}_p^k(\mathbf{x} - \beta^{(n)}). \tag{5}$$

Clearly, the right hand side (RHS) of Inequality (4) is no more than the RHS of Inequality (1) (equal when the best bias β is 0). In the case when all except at most k coordinates of $\mathbf{x}$ are close to a non-zero β, our error bound will be much better than that in (1). For the example mentioned earlier, we have $\min_\beta \mathrm{Err}_1^k(\mathbf{x} - \beta^{(10)}) = 12$ and $\min_\beta \mathrm{Err}_2^k(\mathbf{x} - \beta^{(10)}) = \sqrt{28} \approx 5.29$ ($\arg\min_\beta = 100$; in this example the bias happens to be the same for both $p = 1$ and $p = 2$), which are significantly smaller than those given by Count-Median and Count-Sketch.

Same as Inequality (2), for $p \in \{1, 2\}$ we can derive from (4) that

$$\|\hat{\mathbf{x}} - \mathbf{x}\|_p = O(1) \cdot \min_\beta \mathrm{Err}_p^k(\mathbf{x} - \beta^{(n)}). \tag{6}$$

Remark 1 *Compared with the single bias β, one may want to allow multiple bias values. For example, for the data vector $\mathbf{y} = (200, 100, 50, 50, 50, 50, 100, 100, 100, 10)$, one may want to use two bias values $\beta_1 = 50$ and $\beta_2 = 100$, with 200 and 10 being the outliers. Unfortunately, this* cannot *be done if we want to obtain an $o(n)$ (sub-linear) size sketch where n is the dimension of the input vector, simply because when we have at least two bias values, in the recovery procedure for each of the n coordinates of input vector we need the information of which bias value has been deducted from that coordinate, which costs at least 1 bit.*

Our Contributions. In this paper we have made the following contributions.

1. We have given a rigorously formalization of the bias-aware sketches, which *strictly generalizes* standard linear sketches in the error guarantees.

2. We have proposed bias-aware sketches with rigorous ℓ_∞/ℓ_1 and ℓ_∞/ℓ_2 error guarantees. We have

[1] The proof can be done using the $n^{1/p}$-party set-disjointness hard instance similar to that for p-th frequency moments [4].

also shown how to implement our sketches in the streaming model for fast real-time query.

3. We have implemented our algorithms and verified their effectiveness on both synthetic and real-world datasets. We note that our algorithms significantly outperform the existing algorithms in terms of accuracy for point query.

2. RELATED WORK

The history of data sketch/summary can be traced back to Morris' *approximate counter* [25] and Flajolet and Martin's *probabilistic counting* algorithm [17]. Subsequently, streaming algorithms were extensively investigated since the seminal paper [1] by Alon *et al.* Among them Count-Sketch [6] and Count-Min/Count-Median [11] were found particularly useful in many applications from data analytics and mining to query processing and optimizations. A number of variants of the Count-Min algorithm have also been proposed, such as Count-Min with conservative update [16, 20] and Count-Min-Log with conservative update [27], but these sketches are *not* linear and thus cannot be directly used in the distributed setting. Another closely related algorithm is the Counter-braids [23]. The intent of Counter-braids is to be more bit-efficient than methods which simply use counters. It requires a larger amount of space to execute; and its encoding/decoding procedures are recursive, layer by layer, and thus it cannot answer point query without decoding the whole input vector $\mathbf{x}$. Finally, we would like to emphasize that all of the algorithms mentioned above cannot handle data bias.

Deng *et al.* [13] attempted to remove the bias in the Count-Min algorithm. In the high level, at the time of recovering a coordinate mapped to a hash bucket (see CM-matrix in Definition 1), their algorithm averages the coordinates mapped into all other hash buckets to obtain an estimate of the bias presented in the considered bucket. It turns out that such an estimation is too rough to be useful – their analysis shows that their algorithm can only achieve comparable recovery quality as Count-Sketch.

Yan *et al.* [29] formulated the bias recovery problem in the context of distributed outlier detection. We briefly describe how BOMP works. To sketch a vector $\mathbf{x} \in \mathbb{R}^n$, BOMP first computes $\mathbf{y} = \Phi\mathbf{x}$ where $\Phi = [\phi_1, \ldots, \phi_n] \in \mathbb{R}^{t \times n}$, where each entry of Φ is independently sampled from the Gaussian distribution $\mathcal{N}(0, 1/t)$. In the recovery phase BOMP prepends a new column $\frac{1}{\sqrt{n}} \sum_{i=1}^{n} \phi_i$ to Φ to get $\Phi' = [\frac{1}{\sqrt{n}} \sum_{i=1}^{n} \phi_i, \Phi]$, and then runs OMP (*Orthogonal Matching Pursuit*) on $\mathbf{y}$ and Φ' in $k+1$ iterations to recover $\tilde{\mathbf{x}}$ as an approximation of $\mathbf{x}$. However, their discussion only focused on the biased k-sparse vectors where all coordinates of $\mathbf{x}$ are equal to some unknown value β except at most k "outliers", and did not give a solid theoretical analysis. Moreover, OMP is very time expensive, and cannot answer point query without decoding the whole vector $\mathbf{x}$.

Our work is closely related to the area of compressive sensing. In fact, our linear sketching and recovery algorithms can be seen as natural extensions of the standard compressive sensing sparse recovery algorithms [5, 14, 12]. In the standard sparse recovery setting the bias of the vector is assumed to be 0, which does work well for a number of problems in signal processing but its power is somewhat limited for massive data processing where coordinates in vectors may have non-zero biases. We note that the idea of debiasing can be viewed as a special case of the incoherent dictionary learning [15, 19] – one can add an all-1 vector (normalized by $1/\sqrt{n}$) upon the n standard basis vectors. However, as far as we are concerned, the existing recovery algorithms in incoherent dictionary learning use either linear programming or OMP, which, again, are very time-inefficient on large datasets and do not work for point query.

3. PRELIMINARIES

We summarize the main notations in this paper in Table 1. A quick scan of the table may be useful since some of the notations are not standard (e.g., a vector minus a scalar value: $\mathbf{x} - \beta$).

Table 1: List of notations

$[n]$	$[n] = \{1, 2, \ldots, n\}$
Pr	the probability of
$(\mathbf{x})_i$ or x_i	for $\mathbf{x} \in \mathbb{R}^n$, both $(\mathbf{x})_i$ and x_i represent the i-th coordinate of $\mathbf{x}$
$\|\mathbf{x}\|_p$	$\|\mathbf{x}\|_p = (\sum_i \|x_i\|^p)^{\frac{1}{p}}$ for $\mathbf{x} = (x_1, \ldots, x_n)$; when $p = \infty$, $\|\mathbf{x}\|_\infty = \max_i \|x_i\|$
k-sparse	$\mathbf{x} \in \mathbb{R}^n$ is k-sparse if $\mathbf{x}$ has *at most* k non-zero coordinates
$\mathcal{S}_m(\mathbf{x})$	set of vectors in $\mathbb{R}^m$ obtained by choosing m ($\leq n$) coordinates from $\mathbf{x} \in \mathbb{R}^n$
$\mathrm{Err}_p^k(\mathbf{x})$	$\mathrm{Err}_p^k(\mathbf{x}) = \min_{k\text{-sparse } \mathbf{x}'} \|\mathbf{x} - \mathbf{x}'\|_p$
$\mathbf{x} - \beta$	for $\mathbf{x} \in \mathbb{R}^n, \beta \in \mathbb{R}$, $\mathbf{x} - \beta = (x_1 - \beta, \ldots, x_n - \beta)$
$\mathrm{mean}(\mathbf{x})$	for $\mathbf{x} \in \mathbb{R}^n$, $\mathrm{mean}(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} x_i$
$\mathrm{median}(\mathbf{x})$	for $\mathbf{x} \in \mathbb{R}^n$, $\mathrm{median}(\mathbf{x}) = x_{\frac{n+1}{2}}$ for odd n, $\mathrm{median}(\mathbf{x}) = (x_{\frac{n}{2}} + x_{\frac{n}{2}+1})/2$ for even n
$\mathrm{argmin}_\beta f(\beta)$	$\mathrm{argmin}_\beta f(\beta) = \{\alpha \mid f(\alpha) = \min_\beta f(\beta)\}$
$\sigma^2(\mathbf{x})$	variance of $\mathbf{x} \in \mathbb{R}^n$; $\sigma^2(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mathrm{mean}(\mathbf{x}))^2$
$\sigma^2(Y)$	variance of a random variable Y; $\sigma^2(Y) = \mathbf{E}\left[(Y - \mathbf{E}[Y])^2\right]$
Π	CM-Matrix. See Definition 1
Ψ	CS-Matrix. See Definition 2
Υ	Sampling matrix. See Definition 3

We would like to introduce two classical linear sketches Count-Median and Count-Sketch, which will be used as components in our algorithms.

Count-Median. The Count-Median algorithm [11] is a linear sketch for achieving ℓ_∞/ℓ_1-guarantee. We first introduce the *Count-Median* matrix.

Definition 1 (CM-matrix) *Let $h : [n] \to [s]$ be a hash function. A CM-matrix $\Pi(h) \in \{0,1\}^{s\times n}$ is defined as*

$$\Pi(h)_{i,j} = \begin{cases} 1 & h(j) = i \\ 0 & h(j) \neq i. \end{cases}$$

For a vector $\mathbf{x} \in \mathbb{R}^n$, the following theorem shows that we can recover each coordinate of $\mathbf{x}$ with a bounded error from $\Theta(\log n)$ random sketching vectors $\Pi(h)\mathbf{x}$.

Theorem 1 ([11]) *Set $s = \Theta(k/\alpha)$ for an $\alpha \in (0,1)$ and $d = \Theta(\log n)$. Let $h^1, \ldots, h^d : [n] \to [s]$ be d independent random hash functions, and let $\Pi(h^1), \ldots, \Pi(h^d)$ be the corresponding CM-matrices. Let $\hat{\mathbf{x}} = (\hat{x}_1, \ldots, \hat{x}_n)$ be a vector such that*

$$\hat{x}_j = \underset{i\in[d]}{\text{median}} \left\{ \left(\Pi(h^i)\mathbf{x}\right)_{h^i(j)} \right\}.$$

We have $\mathbf{Pr}\left[\|\hat{\mathbf{x}} - \mathbf{x}\|_\infty \leq \alpha/k \cdot Err_1^k(\mathbf{x})\right] \geq 1 - 1/n$.

Count-Sketch. The Count-Sketch algorithm [6] is a linear sketch for achieving ℓ_∞/ℓ_2-guarantee. It is similar to Count-Median; the main difference is that it introduces random signs in the sketching matrix.

Definition 2 (CS-Matrix) *Let $h : [n] \to [s]$ be a hash function, and $r : [n] \to \{-1, 1\}$ be a random sign function. A CS-matrix $\Psi(h, r) \in \{0,1\}^{s\times n}$ is defined as*

$$\Psi(h,r)_{i,j} = \begin{cases} r(j) & h(j) = i \\ 0 & h(j) \neq i. \end{cases}$$

Similarly, for a vector $\mathbf{x} \in \mathbb{R}^n$, we can recover each coordinate of $\mathbf{x}$ with a bounded error from $\Theta(\log n)$ sketching vectors $\Psi(h, r)\mathbf{x}$.

Theorem 2 ([6]) *Set $s = \Theta(k/\alpha)$ for an $\alpha \in (0,1)$ and $d = \Theta(\log n)$. Let $h^1, \ldots, h^d : [n] \to [s]$ be d independent random hash functions, let $r^1, \ldots, r^d : [n] \to \{-1, 1\}$ be d independent random sign functions, and let $\Psi(h^1, r^1), \ldots, \Psi(h^d, r^d)$ be the corresponding CS-matrices. Let $\hat{\mathbf{x}} = (\hat{x}_1, \ldots, \hat{x}_n)$ be a vector such that*

$$\hat{x}_j = \underset{i\in[d]}{\text{median}} \left\{ r^i(j) \cdot \left(\Psi(h^i, r^i)\mathbf{x}\right)_{h^i(j)} \right\}.$$

We have $\mathbf{Pr}\left[\|\hat{\mathbf{x}} - \mathbf{x}\|_\infty \leq \alpha/\sqrt{k} \cdot Err_2^k(\mathbf{x})\right] \geq 1 - 1/n$.

We will use the following sampling matrix.

Definition 3 (Sampling Matrix) *Let $\Upsilon \in \{0,1\}^{t\times n}$ be a 0/1 matrix by independently setting for each of the t rows exactly one random coordinate to be 1.*

4. BIAS-AWARE SKETCHES

In this section we propose two efficient bias-aware sketches achieving ℓ_∞/ℓ_1-guarantee and ℓ_∞/ℓ_2-guarantee respectively.

4.1 Warm Up

The core of our algorithms is to estimate the bias of the input data. Before presenting our algorithms, we first discuss a few natural approaches that do *not* work, and then illustrate high level ideas of our algorithms.

Using mean as the bias. The first idea is to use the mean of the input vector $\mathbf{x}$. However, this cannot lead to any theoretical error guarantee. Consider the vector $\mathbf{x} = (\infty, \infty, 50, 50, 50, 50, 50, 50, 50, 50)$ where ∞ denotes a very large number, and k is set to be 2. The mean of the coordinates of $\mathbf{x}$ is ∞, but the best bias value is $\beta = 50$ which leads to a tail error 0 (RHS of (4)). Nevertheless, using the mean as the bias may work well in datasets where there are not many extreme values. We will show in our experiments (Section 5) that this is indeed the case for some real-world datasets.

Searching the bias in a post-processing step. Another idea is to search the best bias value β in a post-processing step after performing the existing sketching algorithms such as Count-Sketch and Count-Median, and then subtract it from the original sketch for the recovery. More precisely, we can binary search the best β by computing the RHS of (4) a logarithmic number of times and then picking the best β value that minimize the error $\text{Err}_p^k(\mathbf{x} - \beta^{(n)})$. This idea looks attractive since we can just reuse the existing sketching algorithms. However, such a post-processing does not fit the streaming setting where we want to answer queries in real-time. Indeed, in the streaming model we have to redo the binary search of β for queries coming in different time steps in the streaming process, which makes the individual point query very slow.

Our approaches. In this paper we propose two simple, yet efficient, algorithms to achieve the error guarantee in (4), for $p = 1$ and $p = 2$ respectively. Our algorithms do not need a post-processing step and can thus answer real-time queries in the streaming model. For $p = 1$, we compute by sampling an approximate median (denoted by *med*) of coordinates in $\mathbf{x}$, and use it as the bias. Using the stability of median we can show that *med* is also an approximate median of the vector $\mathbf{x}^*$ obtained from $\mathbf{x}$ by dropping the k "outliers". For $p = 2$, the idea is still to use the mean. However, as we have discussed previously, directly using the mean of all items will not give the desired theoretical guarantee, since the mean can be "contaminated" by the outliers (extreme values). We thus choose to employ a Count-Median sketch and use the mean of the "middle" buckets in the Count-Median sketch as the bias. Both algorithms are conceptually very simple, but the complete analysis turns out to be quite non-trivial. The next two subsections detail our algorithms.

4.2 Recovery with ℓ_∞/ℓ_1-Guarantee

In this section we give a bias-aware sketch with ℓ_∞/ℓ_1-guarantee. That is, we try to design a sketching matrix $\Phi \in \mathbb{R}^{t\times n}$ ($t \ll n$) such that from $\Phi\mathbf{x}$ we can recover an $\hat{\mathbf{x}}$ satisfying $\|\hat{\mathbf{x}} - \mathbf{x}\|_\infty = O(1/k) \cdot \min_\beta \text{Err}_1^k(\mathbf{x} - \beta)$.

Algorithm 1: ℓ_1-SKETCH($\mathbf{x}$)

Input: $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$
Output: sketch of $\mathbf{x}$ and a set $S \subseteq \{x_1, \ldots, x_n\}$
/* assume $s = c_s k$ for a constant $c_s \geq 4$; $d = \Theta(\log n)$; $h^1, \ldots, h^d : [n] \to [s]$ are common knowledge */
generate a sampling matrix $\Upsilon \in \{0,1\}^{20 \log n \times n}$
$\forall i \in [d],\ \mathbf{y}^i \leftarrow \Pi(h^i)\mathbf{x}$
$S \leftarrow \Upsilon \mathbf{x}$
return $S, \{\mathbf{y}^1, \ldots, \mathbf{y}^d\}$

Algorithm 2: ℓ_1-RECOVER($S, \{\mathbf{y}^1, \ldots, \mathbf{y}^d\}$)

Input: S: a set of randomly sampled coordinates of $\mathbf{x}$; $\{\mathbf{y}^i = \Pi(h^i)\mathbf{x} \mid i \in [d]\}$
Output: $\hat{\mathbf{x}}$ as an approximation of $\mathbf{x}$
/* assume $s = c_s k$ for a constant $c_s \geq 4$; $d = \Theta(\log n)$; $h^1, \ldots, h^d : [n] \to [s]$ are common knowledge */
$\hat{\beta} \leftarrow$ median of coordinates in S
$\forall i \in [d],\ \boldsymbol{\pi}^i \leftarrow$ coordinate-wise sum of columns of $\Pi(h^i)$
$\forall i \in [d],\ \tilde{\mathbf{y}}^i \leftarrow \mathbf{y}^i - \hat{\beta}\boldsymbol{\pi}^i$
/* Run Count-Median recovery */
$\forall j \in [n],\ \hat{z}_j \leftarrow \text{median}_{i \in [d]} \left\{ \left(\tilde{\mathbf{y}}^i\right)_{h^i(j)} \right\}$
$\hat{\mathbf{x}} \leftarrow \hat{\mathbf{z}} + \hat{\beta}$
return $\hat{\mathbf{x}}$

4.2.1 Algorithms

We use ℓ_1-S/R (ℓ_1-Sketch/Recover) to denote our algorithm. Its sketching and recovery procedures are described in Algorithm 1 and Algorithm 2 respectively. For simplicity we assume that the two algorithms can jointly sample hash functions $h^1, \ldots, h^d$ for free (i.e., without any costs). Indeed, we can simply choose 2-wise independent hash functions $g, h^i, r^i (i \in [d])$, each of which can be stored in $O(1)$ space. This will not affect any of our mathematical analysis since we will only need to use the second moment of random variables. Thus the total extra space to store random hash functions can be bounded by $O(d) = O(\log n)$, and is negligible compared with the sketch size $O(k \log n)$. In the distributed model we can ask the coordinator to generate these hash functions and then send to all sites, and in the streaming model we can precompute them at the beginning and store them in the memory.

In the sketching phase of ℓ_1-S/R, we simply use sampling to estimate the best β that minimizes $\text{Err}_1^k(\mathbf{x} - \beta)$. More precisely, we sample $\Theta(\log n)$ coordinates from $\mathbf{x}$ and take the median (denoted by $\hat{\beta}$), which we will show is good for the ℓ_∞/ℓ_1-guarantee. The final (implicit) sketching matrix Φ is a vertical concatenation of $d = \Theta(\log n)$ independent CM-matrix $\Pi(h^i)$'s and the sampling matrix Υ.

In the recovery phase, we use Count-Median to recover $\hat{\mathbf{z}}$ as an approximation to the *de-biased* vector $\mathbf{x} - \hat{\beta}$; consequently $\hat{\mathbf{z}} + \hat{\beta}$ will be a good approximation to $\mathbf{x}$.

The following theorem summarizes the performance of ℓ_1-S/R. One can compare it with Theorem 1 for Count-Median.

Theorem 3 *There exists a bias-aware sketching scheme such that for any $\mathbf{x} \in \mathbb{R}^n$, it computes the sketch Φx, and then recovers an $\hat{\mathbf{x}}$ as an approximation to $\mathbf{x}$ from $\Phi\mathbf{x}$ satisfying the following.*

$$\mathbf{Pr}[\|\hat{\mathbf{x}} - \mathbf{x}\|_\infty \leq C_1/k \cdot \min_\beta Err_1^k(\mathbf{x} - \beta)] \geq 1 - C_2/n, \quad (7)$$

where $C_1, C_2 > 0$ are two universal constants. The sketch can be constructed in time $O(n \log n)$; the sketch size is bounded by $O(k \log n)$; the recovery can be done in time $O(n \log n)$.

As mentioned in the introduction, we can convert ℓ_∞/ℓ_1 guarantee to ℓ_1/ℓ_1 guarantee.

Corollary 1 *The $\hat{\mathbf{x}}$ recovered in Theorem 3 also guarantees that with probability $1 - O(1/n)$, we have*

$$\|\hat{\mathbf{x}} - \mathbf{x}\|_1 = O(1) \cdot \min_\beta Err_1^k(\mathbf{x} - \beta).$$

4.2.2 Analysis

Correctness.

Let $\bar{\beta}$ be any β that minimizes the ℓ_1-norm error $\text{Err}_1^k(\mathbf{x} - \beta)$. Let $\mathbf{x}^*$ be the vector obtained by dropping the k coordinates from $\mathbf{x}$ that deviate the most from $\bar{\beta}$. We first show:

Lemma 1 *Given $\mathbf{x} \in \mathbb{R}^n$, pick any $\bar{\beta} \in \text{argmin}_\beta Err_1^k(\mathbf{x} - \beta)$. Let $\mathbf{x}^* \in \mathcal{S}_{n-k}(\mathbf{x})$ be the vector obtained by dropping the k coordinates that deviate the most from $\bar{\beta}$, we must have*

$$\|\mathbf{x}^* - \bar{\beta}\|_1 = \|\mathbf{x}^* - \text{median}(\mathbf{x}^*)\|_1. \quad (8)$$

PROOF. For convenience we assume that $(n - k)$ is odd, and then $\|\mathbf{x}^* - \beta\|_1$ reaches the minimum only when $\beta = \text{median}(\mathbf{x}^*)$. It is easy to verify that our lemma also holds when $(n - k)$ is even. Under this assumption, we only need to show $\bar{\beta} = \text{median}(\mathbf{x}^*)$.

We prove by contradiction. Suppose $\bar{\beta} \neq \text{median}(\mathbf{x}^*)$, then

$$\text{Err}_1^k(\mathbf{x} - \text{median}(\mathbf{x}^*)) \leq \|\mathbf{x}^* - \text{median}(\mathbf{x}^*)\|_1 < \text{Err}_1^k(\mathbf{x} - \bar{\beta}),$$

contradicting the definition of $\bar{\beta}$. □

Lemma 1 gives a more intuitive understanding of the best β that minimizes $\text{Err}_1^k(\mathbf{x} - \beta)$, and it connects to the idea that the median of coordinates works. But we are not quite there yet since in (8) we need the *exact* median of a vector $\mathbf{x}^*$ that we do *not* know before figuring out $\bar{\beta}$. To handle this we need the followings two lemmas.

The first lemma says that a value that is *close* (but not necessary equal) to the median of coordinates of $\mathbf{x}^*$ can be used to approximate the best $\bar{\beta}$.

Lemma 2 *Given a vector $\mathbf{x} \in \mathbb{R}^m$ with its coordinates sorted non-decreasingly: $x_1 \le x_2 \le \ldots \le x_m$, for any j such that $\frac{m}{4} < j < \frac{3m}{4}$, we have*

$$\sum_{i\in[m]} |x_i - x_j| \le 2 \cdot \min_{\beta} \sum_{i\in[m]} |x_i - \beta|.$$

PROOF. For simplicity we assume m is odd; the even case can be handled similarly. Let $t = (m+1)/2$ be the index of the median coordinate. If $j = t$ then we are done. Otherwise, w.l.o.g., we assume $j < t$. We have

$$\begin{aligned}
&\left(\sum_{i=1}^{m} |x_i - x_j|\right) - \left(\sum_{i=1}^{m} |x_i - x_t|\right) \\
=\ & \sum_{i=1}^{t-j} i \cdot (x_{t-i+1} - x_{t-i}) \\
=\ & -(t-j)\cdot x_j + \sum_{i=j+1}^{t} x_i \\
=\ & \sum_{i=j+1}^{t} (x_i - x_j) \\
\le\ & \sum_{i=1}^{m/4} (x_t - x_i) \quad \left(\text{since } \frac{m}{4} < j < t = \frac{m+1}{2}\right) \\
\le\ & \sum_{i=1}^{m} |x_i - x_t|.
\end{aligned}$$

The lemma follows. □

The second lemma says that the median of $O(\log n)$ randomly sampled coordinates of $\mathbf{x}$ is close to the median of coordinates of the unknown vector $\mathbf{x}^*$.

Lemma 3 *Given a vector $\mathbf{x} \in \mathbb{R}^n$ with its coordinates sorted non-decreasingly: $x_1 \le x_2 \le \ldots \le x_n$, if we randomly sample with replacement $t = 20 \log n$ coordinates from $\mathbf{x}$, then with probability at least $1 - 1/n$ the median of the t samples falls into the range $[x_{n/2-n/6}, x_{n/2+n/6}]$.*

PROOF. Let $X_1, \ldots, X_t$ be the samples we pick. The median of them does not fall into the range

$$[x_{n/2-n/6}, x_{n/2+n/6}]$$

if and only if one of the following events happens,

- $\mathcal{E}_1$: at least half of the samples larger than $x_{n/2+n/6}$.
- $\mathcal{E}_2$: at least half of the samples smaller than $x_{n/2-n/6}$.

We first bound the probability that $\mathcal{E}_1$ happens. Let Y_i be the random variables such that $Y_i = 1$ if $X_i > x_{n/2+n/6}$, and $Y_i = 0$ otherwise. We have $\mathbf{E}\left[\sum_{i=1}^{t} Y_i\right] = t/3$. By a Chernoff bound, we have

$$\begin{aligned}
&\mathbf{Pr}\left[\sum_{i=1}^{t} Y_i - \frac{t}{3} > \frac{t}{6}\right] \\
\le\ & \exp\left(-\frac{t}{12}\right) \\
<\ & 1/(2n). \qquad (t = 20\log n)
\end{aligned}$$

Similarly we can show that the probability that $\mathcal{E}_2$ happens is at most $1/(2n)$. The lemma follows. □

Now we are ready to prove the theorem.

PROOF. (of Theorem 3) W.l.o.g. we assume the coordinates of $\mathbf{x}$ are sorted as $x_1 \le x_2 \le \ldots \le x_n$. To simplify the discussion, we assume t at Line 3 of Algorithm 1 is odd. The even case can be verified similarly.

Let $\hat{\beta}$ be the median of the t samples in S (Line 1 in Algorithm 2). Let $\alpha \in \operatorname{argmin}_\beta \mathrm{Err}_1^k(\mathbf{x} - \beta)$. Let $\mathbf{x}^*$ be the vector obtained by dropping the k coordinates from $\mathbf{x}$ that deviate the most from α.

By Lemma 3, $\hat{\beta} \in [x_{n/2-n/6}, x_{n/2+n/6}]$ holds with probability $1 - 1/n$. Note that we can assume that $k = O(n/\log n)$ (otherwise the sketch can just be $\mathbf{x}$ itself which has size $O(k \log n)$). We thus have

$$\mathbf{Pr}\left[(\mathbf{x}^*)_{\frac{(n-k)}{4}} \le \hat{\beta} \le (\mathbf{x}^*)_{\frac{3(n-k)}{4}}\right] > 1 - \frac{1}{n}. \quad (9)$$

Applying Lemma 2 to $\mathbf{x}^*$ (with $m = n - k$), with probability at least $(1 - 1/n)$ it holds that

$$\begin{aligned}
\mathrm{Err}_1^k(\mathbf{x} - \hat{\beta}) &\le \|\mathbf{x}^* - \hat{\beta}\|_1 \\
&\le 2 \cdot \min_{\beta} \|\mathbf{x}^* - \beta\|_1 \quad \text{(by (9) and Lemma 2)} \\
&= 2 \cdot \|\mathbf{x}^* - \mathrm{median}(\mathbf{x}^*)\|_1 \\
&= 2 \cdot \|\mathbf{x}^* - \alpha\|_1 \qquad \text{(by Lemma 1)} \\
&= 2 \cdot \min_{\beta} \mathrm{Err}_1^k(\mathbf{x} - \beta), \qquad (10)
\end{aligned}$$

where the last equality holds due to the definitions of α and $\mathbf{x}^*$. By Theorem 1 (property of Count-Median) and Line 4 of Algorithm 2 we have

$$\mathbf{Pr}\left[\|\hat{\mathbf{z}} - (\mathbf{x} - \hat{\beta})\|_\infty = O\left(\frac{1}{k}\right) \cdot \mathrm{Err}_1^k(\mathbf{x} - \hat{\beta})\right] \ge 1 - \frac{1}{n}.$$

Since at Line 5 we set $\hat{\mathbf{x}} = \hat{\mathbf{z}} + \hat{\beta}$, we have

$$\mathbf{Pr}\left[\|\hat{\mathbf{x}} - \mathbf{x})\|_\infty = O\left(\frac{1}{k}\right) \cdot \mathrm{Err}_1^k(\mathbf{x} - \hat{\beta})\right] \ge 1 - \frac{1}{n}. \quad (11)$$

Inequality (7) of Theorem 3 follows from (10) and (11). □

Complexities. Since CM-matrix only has one nonzero entry in each column, using sparse matrix representation we can compute $\Pi(h^i)\mathbf{x}$ ($i \in [d]$) in $O(n)$ time. Thus the sketching phase can be done in time $O(nd) = O(n \log n)$.

The sketch size is $O(k \log n)$ since each $\Psi(h^i)\mathbf{x}$ ($i \in [d]$) has size $O(k)$.

Algorithm 3: ℓ_2-SKETCH($\mathbf{x}$)

Input: $\mathbf{x} \in \mathbb{R}^n$
Output: the sketch of $\mathbf{x}$
/* assume $s = c_s k$ for a constant $c_s \geq 4$; $d = \Theta(\log n)$; $g, h^1, \ldots, h^d : [n] \to [s]$; $r^1, \ldots, r^d : [n] \to \{-1, 1\}$ are common knowledge */
1 $\mathbf{w} \leftarrow \Pi(g)\mathbf{x}$
2 $\forall i \in [d],\ \mathbf{y}^i \leftarrow \Psi(h^i, r^i)\mathbf{x}$
3 **return** $\mathbf{w}, \{\mathbf{y}^1, \ldots, \mathbf{y}^d\}$

Algorithm 4: ℓ_2-RECOVER($\mathbf{w}, \{\mathbf{y}^1, \ldots, \mathbf{y}^d\}$)

Input: $\mathbf{w} = \Pi(g)\mathbf{x}$; $\{\mathbf{y}^i = \Psi(h^i, r^i)\mathbf{x} \mid i \in [d]\}$
Output: $\hat{\mathbf{x}}$ as an approximation of $\mathbf{x}$
/* assume $s = c_s k$ for a constant $c_s \geq 4$; $d = \Theta(\log n)$; $g, h^1, \ldots, h^d : [n] \to [s]$; $r^1, \ldots, r^d : [n] \to \{-1, 1\}$ are common knowledge */
1 $\boldsymbol{\pi} \leftarrow$ coordinate-wise sum of columns of $\Pi(g)$
2 w.l.o.g. assume $w_1/\pi_1 \leq \ldots \leq w_s/\pi_s$; set $\hat{\beta} = \sum_{i=s/2-k}^{s/2+k-1} w_i \Big/ \sum_{i=s/2-k}^{s/2+k-1} \pi_i$
3 $\forall i \in [d],\ \boldsymbol{\psi}^i \leftarrow$ coordinate-wise sum of columns of $\Psi(h^i, r^i)$
4 $\forall i \in [d],\ \tilde{\mathbf{y}}^i \leftarrow \mathbf{y}^i - \hat{\beta}\boldsymbol{\psi}^i$
/* Run the Count-Sketch recovery */
5 $\forall j \in [n],\ \hat{z}_j \leftarrow \text{median}_{i \in [d]} \left\{ r^i(j) \cdot \left(\tilde{\mathbf{y}}^i\right)_{h^i(j)} \right\}$
6 $\hat{\mathbf{x}} \leftarrow \hat{\mathbf{z}} + \hat{\beta}$
7 **return** $\hat{\mathbf{x}}$

In the recovery phase, the dominating cost is the computation of coordinates in $\hat{\mathbf{z}}$, for each of which we need $O(d) = O(\log n)$ time. Thus the total cost is $O(n \log n)$.

4.3 Recovery with ℓ_∞/ℓ_2-Guarantee

In this section we give a bias-aware sketch with ℓ_∞/ℓ_2-guarantee. That is, we try to design a sketching matrix $\Phi \in \mathbb{R}^{t \times n}$ ($t \ll n$) such that from $\Phi\mathbf{x}$ we can recover an $\hat{\mathbf{x}}$ satisfying $\|\hat{\mathbf{x}} - \mathbf{x}\|_\infty = O(1/\sqrt{k}) \cdot \min_\beta \text{Err}_2^k(\mathbf{x} - \beta)$.

4.3.1 Algorithms

We use ℓ_2-S/R (ℓ_2-Sketch/Recover) to denote our algorithm. Its sketching and recovery procedures are described in Algorithm 3 and Algorithm 4 respectively.

We again assume that the sketching algorithm and the recovery algorithm can jointly sample (1) independent random hash functions $g, h^1, \ldots, h^d : [n] \to [s]$ and (2) independent random signed functions $r^1, \ldots, r^d : [n] \to \{-1, 1\}$ without any costs.

In our algorithms we first use the CM-matrix to obtain a good approximation $\hat{\beta}$ of the β that minimizes $\text{Err}_2^k(\mathbf{x} - \beta)$, and then use the Count-Sketch algorithm to recover $\hat{\mathbf{z}}$ as an approximation to the *de-biased* vector $\mathbf{x} - \hat{\beta}$; and consequently $\hat{\mathbf{z}} + \hat{\beta}$ will be a good approximation to $\mathbf{x}$. The final (implicit) sketching matrix $\Phi \in \mathbb{R}^{s(d+1) \times n}$ in Algorithm 3 is a vertical concatenation of a CM-matrix $\Pi(g)$ and $d = \Theta(\log n)$ independent CS-matrices $\Psi(h^i, r^i)$'s.

In Algorithm 4, to approximate the best β we first sum up all the columns of $\Pi(g)$, giving a vector $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_s)$ (Line 1). Let $\mathbf{w} = \Pi(g)\mathbf{x} \in \mathbb{R}^s$. W.l.o.g. assume that $w_1/\pi_1 \leq \ldots \leq w_s/\pi_s$. We estimate β by

$$\hat{\beta} = \sum_{i=s/2-k}^{s/2+k-1} w_i \Big/ \sum_{i=s/2-k}^{s/2+k-1} \pi_i \ .$$

The intuition of this estimation is the following. First note that w_i/π_i ($i \in [s]$) is the average of coordinates of $\mathbf{x}$ that are hashed into the i-th coordinate(bucket) of sketching vector $\Pi(g)\mathbf{x}$. In the case that there is no "outlier" coordinate of $\mathbf{x}$ that is hashed into the i-th bucket of $\Pi(g)\mathbf{x}$, then w_i/π_i ($i \in [s]$) should be close to the best bias β. Since there are at most k outliers, if we choose $s \geq 4k$ then most of these s buckets in $\Pi(g)$ will *not* be "contaminated" by outliers.

The next idea is to sort the buckets according to the average of coordinates of $\mathbf{x}$ hashed into it (*i.e.*, w_i/π_i), and then choose the $2k$ buckets around the median and take the average of coordinates hashed into those buckets (Line 2). We can show that the average of coordinates of $\mathbf{x}$ that are hashed into these $2k$ "median" buckets is a good estimation of the best β. Note that there could still be outliers hashed into the median $2k$ buckets, but we are able to prove that such outliers will not affect the estimation of β by much. After getting an estimate of β we de-bias the sketching vector $\mathbf{y}$ (Line 3 and 4) for the next step recovery (Line 5 and 6).

The following theorem summarizes the performance of ℓ_2-S/R. One can compare it with Theorem 2 for Count-Sketch.

Theorem 4 *There exists a bias-aware sketching scheme such that for any $\mathbf{x} \in \mathbb{R}^n$, it computes $\Phi\mathbf{x}$, and then recovers an $\hat{\mathbf{x}}$ as an approximation to $\mathbf{x}$ from $\Phi\mathbf{x}$ satisfying the following:*

$$\mathbf{Pr}[\|\hat{\mathbf{x}} - \mathbf{x}\|_\infty \leq C_1/\sqrt{k} \cdot \min_\beta Err_2^k(\mathbf{x} - \beta)] \geq 1 - C_2/n, \tag{12}$$

where $C_1, C_2 > 0$ are two universal constants. The sketch can be constructed in time $O(n \log n)$; the sketch size is bounded by $O(k \log n)$; the recovery can be done in time $O(n \log n)$.

As mentioned in the introduction, we can convert ℓ_∞/ℓ_2 guarantee to ℓ_2/ℓ_2 guarantee.

Corollary 2 *The $\hat{\mathbf{x}}$ recovered in Theorem 4 also guarantees that with probability $1 - O(1/n)$, we have*

$$\|\hat{\mathbf{x}} - \mathbf{x}\|_2 = O(1) \cdot \min_\beta Err_2^k(\mathbf{x} - \beta).$$

4.3.2 Analysis

Correctness.

Similar to the ℓ_1 case, we first replace the somewhat obscure expression $\min_\beta \mathrm{Err}_2^k(\mathbf{x}-\beta)$ in Theorem 4 with another one which is more convenient to use.

Lemma 4 *For any $\mathbf{x} \in \mathbb{R}^n$ and $k < n$, let $\mathbf{x}^*$ be a vector in $\mathcal{S}_{n-k}(\mathbf{x})$ that has the minimum variance. It holds that*

$$\left(\min_\beta Err_2^k(\mathbf{x}-\beta)\right)^2 = (n-k)\sigma^2(\mathbf{x}^*) = \|\mathbf{x}^* - \mathrm{mean}(\mathbf{x}^*)\|_2^2. \quad (13)$$

Furthermore, $\mathbf{x}^*$ is equivalent to the vector obtained by dropping the k coordinates from $\mathbf{x}$ that deviate the most from $\mathrm{mean}(\mathbf{x}^*)$.

Proof. First, by the definition of $\mathbf{x}^*$ we have

$$(n-k)\sigma^2(\mathbf{x}^*) = \min_{\mathbf{x}' \in \mathcal{S}_{n-k}(\mathbf{x})} \|\mathbf{x}' - \mathrm{mean}(\mathbf{x}')\|_2^2.$$

By the definition of $\mathrm{Err}_2^k(\cdot)$, we have

$$\min_\beta \mathrm{Err}_2^k(\mathbf{x}-\beta) \geq \min_{\mathbf{x}' \in \mathcal{S}_{n-k}(\mathbf{x})} \min_\beta \|\mathbf{x}' - \beta\| = \min_{\mathbf{x}' \in \mathcal{S}_{n-k}(\mathbf{x})} \|\mathbf{x}' - \mathrm{mean}(\mathbf{x}')\|_2.$$

Thus to prove (13), it suffices to show that

$$\min_\beta \mathrm{Err}_2^k(\mathbf{x}-\beta) \leq \min_{\mathbf{x}' \in \mathcal{S}_{n-k}(\mathbf{x})} \|\mathbf{x}' - \mathrm{mean}(\mathbf{x}')\|_2.$$

Since the order of the coordinates in $\mathbf{x}$ do not matter, w.l.o.g. we assume $\mathbf{x}^* = (x_1, x_2, \ldots, x_{n-k})$. Let $\gamma = \mathrm{mean}(\mathbf{x}^*)$, and write $x_i = \gamma + \Delta_i$, or equivalently, $\Delta_i = x_i - \gamma$. Note that if

$$\min_{i \in [n]\setminus[n-k]} |\Delta_i| \geq \max_{i \in [n-k]} |\Delta_i|, \quad (14)$$

then we are done because

$$\min_\beta \mathrm{Err}_2^k(\mathbf{x}-\beta)^2 \leq \mathrm{Err}_2^k(\mathbf{x}-\gamma)^2 = \sum_{i \in [n-k]} \Delta_i^2 = \|\mathbf{x}^* - \gamma\|_2^2. \quad (15)$$

Now we assume (14) is false. Again w.l.o.g., we assume $|\Delta_1| = \max_{i \in [n-k]} |\Delta_i|$ and $|\Delta_n| = \min_{i \in [n]\setminus[n-k]} |\Delta_i|$, then $|\Delta_1| > |\Delta_n|$. Let $\mathbf{x}' = (x_2, x_3, \ldots, x_{n-k-1}, x_{n-k}, x_n) \in \mathcal{S}_{n-k}$, that is, $\mathbf{x}'$ is obtained by dropping x_i from $\mathbf{x}^*$ and then appending x_n, we have

$$\begin{aligned}
\|\mathbf{x}' - \mathrm{mean}(\mathbf{x}')\|_2^2 &\leq \|\mathbf{x}' - \mathrm{mean}(\mathbf{x}^*)\|_2^2 \\
&= \|\mathbf{x}' - \gamma\|_2^2 \quad \text{(by definition of } \gamma) \\
&= \Delta_n^2 + \sum_{i=2}^{n-k} \Delta_i^2 \quad \text{(by definition of } \Delta_i) \\
&< \Delta_1^2 + \sum_{i=2}^{n-k} \Delta_i^2 \\
&= \|\mathbf{x}^* - \mathrm{mean}(\mathbf{x}^*)\|_2,
\end{aligned}$$

which contradicts the definition of $\mathbf{x}^*$. Hence (14) holds, and consequently (13) holds.

On the other hand, (14) also implies that $x_{n-k+1}, \ldots, x_n$ are the k coordinates of $\mathbf{x}$ that deviate the most from $\gamma = \mathrm{mean}(\mathbf{x}^*)$. □

We then show (using Lemma 4) that a good approximation of $\mathrm{mean}(\mathbf{x}^*)$ is also a good approximation of the best β.

Lemma 5 *For any $\mathbf{x} \in \mathbb{R}^n$ and $k < n$, let $\mathbf{x}^*$ be a vector in $\mathcal{S}_{n-k}(\mathbf{x})$ that has the minimum variance. For any α such that $|\mathrm{mean}(\mathbf{x}^*) - \alpha|^2 \leq C \cdot \sigma^2(\mathbf{x}^*)$ for any constant $C > 0$, we have*

$$Err_2^k(\mathbf{x}-\alpha)^2 = O\left(\min_\beta Err_2^k(\mathbf{x}-\beta)^2\right).$$

Proof. W.l.o.g. we again assume $\mathbf{x}^* = (x_1, \ldots, x_{n-k})$. Define $f(b) \triangleq \|\mathbf{x}^* - b\|_2^2$. Let $\gamma = \mathrm{mean}(\mathbf{x}^*)$. By Lemma 4 we have

$$f(\gamma) = (n-k)\sigma^2(\mathbf{x}^*) = \|\mathbf{x}^* - \gamma\|_2^2 = \min_\beta \mathrm{Err}_2^k(\mathbf{x}-\beta)^2. \quad (16)$$

Write $\alpha = \gamma + \Delta$ and thus $\Delta^2 \leq C\sigma^2(\mathbf{x}^*)$,

$$\begin{aligned}
\mathrm{Err}_2^k(\mathbf{x}-\alpha)^2 &\leq \|\mathbf{x}^* - \alpha\|_2^2 \\
&= f(\alpha) = f(\gamma + \Delta) \\
&= \sum_{i \in [n-k]} ((x_i - \gamma) - \Delta)^2 \\
&= (n-k)\Delta^2 + \sum_{i=1}^{n-k} (x_i - \gamma)^2 - 2\Delta \sum_{i=1}^{n-k} (x_i - \gamma) \\
&\leq (n-k) \cdot C\sigma^2(\mathbf{x}^*) + \|\mathbf{x}^* - \gamma\|_2^2 + 0 \\
&= O\left(\min_\beta \mathrm{Err}_2^k(\mathbf{x}-\beta)^2\right). \quad \text{(by (16))}
\end{aligned}$$

We are done. □

The next lemma is crucial. It shows that the approximation $\hat{\beta}$ obtained in the recovery algorithm (Algorithm 4) is a good approximation of $\mathrm{mean}(\mathbf{x}^*)$.

Lemma 6 *Let $\hat{\beta}$ be given at Line 2 of Algorithm 4. If $s = c_s k$ for a sufficiently large constant $c_s \geq 4$, it holds that*

$$\mathbf{Pr}\left[\left(\hat{\beta} - \mathrm{mean}(\mathbf{x}^*)\right)^2 = O\left(\sigma^2(\mathbf{x}^*)\right)\right] = 1 - O\left(\frac{1}{n}\right)$$

for any $\mathbf{x}^*$ in $\mathcal{S}_{n-k}(\mathbf{x})$ that has the minimum variance.

Before proving Lemma 6, we need a bound on the difference between the average of all coordinates of a vector and the average of a subset of coordinates.

Lemma 7 *Let $\mathbf{x} = \{x_1, \ldots, x_m\} \in \mathbb{R}^m$ be a vector. Let S be a subset of $\mathbf{x}$'s coordinates of size $|S| = \Theta(m)$. Let $\mu = \frac{1}{m}\sum_{i \in [m]} x_i$, and $\mu' = \frac{1}{|S|}\sum_{i \in S} x_i$. Then we have*

$$\left|\mu' - \mu\right|^2 = O\left(\sigma^2(\mathbf{x})\right).$$

PROOF. W.l.o.g., let $x_1 \le \ldots \le x_m$. Let $\alpha \in (0,1)$ be an arbitrary constant. Let

$$\mu_1 = \frac{1}{\alpha m} \sum_{i \in [\alpha m]} x_i \quad \text{and} \quad \mu_2 = \frac{1}{(1-\alpha)m} \sum_{i \in [m] \backslash [\alpha m]} x_i.$$

We only need to prove two extreme cases where $S = [\alpha m]$ or $S = [m] \backslash [\alpha m]$ (in both cases $|S| = \Theta(n)$ since $\alpha = (0,1)$ is an arbitrary constant):

$$|\mu_1 - \mu|^2 \le \frac{1}{\alpha} \cdot \sigma^2(\mathbf{x}) \quad \text{and} \quad |\mu_2 - \mu|^2 \le \frac{1}{1-\alpha} \cdot \sigma^2(\mathbf{x}). \tag{17}$$

For simplicity (and w.o.l.g.), we assume $\mu = 0$, since we can always define $y_i = x_i - \mu$ and prove on y_i's. We can write the variance of $\mathbf{x}$ as

$$\begin{aligned} \sigma^2(\mathbf{x}) &= \frac{1}{m} \left(\sum_{i \in [\alpha m]} x_i^2 + \sum_{i \in [m] \backslash [\alpha m]} x_i^2 \right) \\ &\ge \left(\alpha m \mu_1^2 + (1-\alpha) m \mu_2^2 \right) / m \quad \text{(Cauchy-Schwarz)} \\ &= \alpha \mu_1^2 + (1-\alpha) \mu_2^2. \end{aligned}$$

(17) follows straightforwardly. □

PROOF. (of Lemma 6) Fix any $\mathbf{x}^* = (x_{i_1}, \ldots, x_{i_{n-k}})$ in $\mathcal{S}_{n-k}(\mathbf{x})$ that has the minimum variance. Let O be the set of the top-k indices i in $\mathbf{x}$ that maximize $|x_i - \text{mean}(\mathbf{x}^*)|$. By Lemma 4 we have that $\mathbf{x}^*$ can be obtained from $\mathbf{x}$ by dropping coordinates indexed by O.

We call an index $i \in [s]$ in the sketching vector $\mathbf{w} = \Pi(g)\mathbf{x}$ *contaminated* if there is at least one $o \in O$ such that $g(o) = i$. W.l.o.g., we assume $w_1/\pi_1 \le \ldots \le w_s/\pi_s$ where $\boldsymbol{\pi}$ is defined at Line 1 of Algorithm 4. Let

$$I = \{i \mid s/2 - k \le i < s/2 + k\}$$

be the $2k$ "median" indices of $\mathbf{w}$, and

$$\bar{I} = \{i \mid i < s/2 - k \vee i \ge s/2 + k\}$$

be the rest of indices in $\mathbf{w}$. Since $|O| = k$ and $s \ge 4k$, there are *at most* k coordinates in I that are contaminated, and *at least* k coordinates in $\bar{I}$ that are *not* contaminated.

The approximation to the bias β at Line 2 of Algorithm 4 can be written as

$$\hat{\beta} = \frac{\sum_{i \in I} w_i}{\sum_{i \in I} \pi_i}. \tag{18}$$

Let $O' = I \cap g(O)$ be the indices in I that are contaminated, and let J be an arbitrary subset of $\bar{I}$ with size $|O'|$. Define

$$\gamma_1 = \min_J \frac{\sum_{i \in I \cup J \backslash O'} w_i}{\sum_{i \in I \cup J \backslash O'} \pi_i} \quad \text{and} \quad \gamma_2 = \max_J \frac{\sum_{i \in I \cup J \backslash O'} w_i}{\sum_{i \in I \cup J \backslash O'} \pi_i}. \tag{19}$$

It is easy to see that $\gamma_1 \le \hat{\beta} \le \gamma_2$: since $s \ge 4k$, one can always find a subset $J \subseteq \bar{I}$ of size $|O'|$ such that for any $j \in J, o \in O'$ we have $w_j/\pi_j \ge w_o/\pi_o$, and replacing O' with J only increases the RHS of (18); On the other hand one can also find a subset $J \subseteq \bar{I}$ of size $|O'|$ such that for any $j \in J, o \in O'$ we have $w_j/\pi_j \le w_o/\pi_o$, and replacing O' with J only decreases the RHS of (18).

We now show that both γ_1 and γ_2 deviate at most $O(\sigma(\mathbf{x}^*))$ from $\text{mean}(\mathbf{x}^*)$, and consequently $\hat{\beta}$, which is sandwiched by γ_1 and γ_2, deviates from $\text{mean}(\mathbf{x}^*)$ by at most $O(\sigma(\mathbf{x}^*))$. Consider the set $G = g^{-1}(I \cup J \backslash O')$. First, by definitions of I, J and O' we have $G \subseteq \{i_1, \ldots, i_{n-k}\}$; and thus $\{x_j \mid j \in G\}$ are coordinates in $\mathbf{x}^*$. Second, since $|I \cup J \backslash O'| = \Theta(k)$ and g is a random mapping from $[n]$ to $[s]$, by a Chebyshev inequality we have $|G| = \Theta(n)$ with probability at least $1 - O(1/n)$.[2] For any $J \subseteq \bar{I}$ of size $|O'|$, let

$$\gamma_J = \frac{1}{|G|} \sum_{j \in G} x_j = \frac{\sum_{i \in I \cup J \backslash O'} w_i}{\sum_{i \in I \cup J \backslash O'} \pi_i}.$$

By Lemma 7, we have

$$|\gamma_J - \text{mean}(\mathbf{x}^*)| = O(\sigma(\mathbf{x}^*)). \tag{20}$$

Since Inequality (20) applies to any $J \subseteq \bar{I}$ of size $|O'|$, we have $|\gamma - \text{mean}(\mathbf{x}^*)| = O(\sigma(\mathbf{x}^*))$ for any $\gamma \in \{\gamma_1, \gamma_2\}$. □

Finally we prove Theorem 4 using Lemma 5 and 6; we show that the obtained $\hat{\beta}$ is a good approximation of the best β that minimizes $\text{Err}_2^k(\mathbf{x} - \beta)$.

PROOF. (of Theorem 4) Let $\mathbf{x}^*$ be a vector in $\mathcal{S}_{n-k}(\mathbf{x})$ that has the minimum variance. At Line 4-5 in Algorithm 4 the Count-Sketch recovery algorithm is used to compute $\hat{\mathbf{z}}$ as an approximation to $\mathbf{x} - \hat{\beta}$. By Theorem 2 we have

$$\mathbf{Pr}\left[\|\hat{\mathbf{z}} - (\mathbf{x} - \hat{\beta})\|_\infty = O\left(\frac{1}{\sqrt{k}}\right) \cdot \text{Err}_2^k(\mathbf{x} - \hat{\beta}) \right] \ge 1 - \frac{1}{n}.$$

Since at Line 6 we set $\hat{\mathbf{x}} = \hat{\mathbf{z}} + \hat{\beta}$, it holds that

$$\mathbf{Pr}\left[\|\hat{\mathbf{x}} - \mathbf{x}\|_\infty = O\left(\frac{1}{\sqrt{k}}\right) \cdot \text{Err}_2^k(\mathbf{x} - \hat{\beta}) \right] \ge 1 - \frac{1}{n}. \tag{21}$$

By Lemma 6,

$$\mathbf{Pr}\left[|\hat{\beta} - \text{mean}\, \mathbf{x}^*| = O\left(\sigma(\mathbf{x}^*)\right) \right] = 1 - O\left(\frac{1}{n}\right).$$

Plugging it to Lemma 5 we have with probability at least $(1 - O(1/n))$ that

$$\text{Err}_2^k(\mathbf{x} - \hat{\beta}) = O\left(\min_\beta \text{Err}_2^k(\mathbf{x} - \beta) \right). \tag{22}$$

Inequality (12) in Theorem 4 follows from (21) and (22). □

[2] More precisely, define for each $i \in [n]$ a random variable Y_i, which is 1 if $g(i) \in I \cup J \backslash O'$ and 0 otherwise. Since $|I \cup J \backslash O'| = \Theta(k)$ and $s = \Theta(k)$, we have $\mathbf{E}[Y_i] = \Theta(1)$, and $\mathbf{Var}[Y_i] \le \mathbf{E}[Y_i^2] = O(1)$. Next note that $|G| = \sum_{i \in [n]} Y_i$. We thus can apply a Chebyshev inequality on Y_i's and conclude that $|G| = \Theta(n)$ with probability $1 - O(1/n)$.

Complexities. Since each CS-Matrix or CM-matrix only has one non-zero entry in each column, using sparse matrix representation we can compute $\Psi(h^i, r^i)\mathbf{x}$ ($i \in [d]$) or $\Pi(g)\mathbf{x}$ in $O(n)$ time. Thus the sketching phase can be done in time $O(nd) = O(n \log n)$.

The sketch size is $O(k \log n)$, simply because $\Pi(g)\mathbf{x}$ and each $\Psi(h^i, r^i)\mathbf{x}$ ($i \in [d]$) has size $O(k)$.

In the recovery phase, the dominating cost is the computation of coordinates in $\hat{\mathbf{z}}$, for each of which we need $O(d) = O(\log n)$ time. Thus the total cost is $O(n \log n)$.

4.4 Streaming Implementations

We now discuss how to maintain the bias (estimation) β at any time step in the streaming setting. This is useful since we would like to answer individual point queries efficiently without decoding the whole vector $\mathbf{x}$; to this end we need to first maintain β efficiently.

For the ℓ_∞/ℓ_1 guarantee we can easily maintain a good approximation of β with $O(\log \log n)$ time per update: we can simply keep the $\Theta(\log n)$ sampled coordinates sorted (e.g., using a balanced binary search tree) during the streaming process, and use their median as an approximation of β at any time step. For the ℓ_∞/ℓ_2 guarantee, the recovery procedure in Algorithm 4 takes the average of items in the middle $2k$ of the s sorted buckets $w_1/\pi_1 \leq \ldots \leq w_s/\pi_s$. This can again be done in $O(\log s) = O(\log k + \log \log n)$ time per update using a balanced binary search tree. An alternative implementation using *biased heaps* is described Algorithm 5. The idea of the algorithm is very simple: we use heaps to keep track of $\sum_{i\in A} w_i, \sum_{i\in C} w_i, \sum_{i\in A} \pi_i, \sum_{i\in C} \pi_i$, where A is the set of the top $(s/2 - k)$ coordinates and C is the set of the bottom $(s/2 - k)$ coordinates (the order is defined by w_i/π_i). Using these sums together with $\sum_{i\in[s]} w_i$ and $\|\pi\|_1$ we can well estimate the bias.

The full streaming algorithm for ℓ_∞/ℓ_2 guarantee is described in Algorithm 6, which is similar to Algorithm 4 but has been augmented to fit the streaming model. The one for ℓ_∞/ℓ_1 guarantee can be done similarly, and we omit here.

Finally we comment on how to generate and store random hash functions in the streaming setting. In fact, we can simply choose hash functions $g, h^i, r^i (i \in [d])$ to be 2-wise independent (and each will use $O(1)$ space to store). This will not affect any of our analysis since we only need to use the second moment of random variables (same in the proofs for Theorem 1 and Theorem 2 for the CM-sketch and CS-sketch, see [6, 11]). Thus the total extra space to store random hash functions can be bounded by $O(d) = O(\log n)$, and is negligible compared with the sketch size $O(k \log n)$.

Algorithm 5: Bias-Heap

Input: s (# of rows of CM-matrix Π), and vector $\boldsymbol{\pi}$ (coordinate-wise sum of columns of matrix Π)

create s nodes, each of which is associated with a (key, value, id) triple $(w_i/\pi_i, w_i, i)$ where $w_i = 0$ and π_i is the i-th coordinate of $\boldsymbol{\pi}$

`/* key is the priority of nodes in the heap */`

set $k \leftarrow s/4$

initialize a min-heap A for nodes 1 to $s/2 - k - 1$

initialize a max-heap B for nodes $s/2 - k$ to s

initialize a max-heap C for nodes $s/2 + k$ to s

initialize a min-heap D for nodes 1 to $s/2 + k - 1$

$\pi_A \triangleq \sum_{i\in\{\text{all id in } A\}} \pi_i$; $\pi_C \triangleq \sum_{i\in\{\text{all id in } C\}} \pi_i$

$w_A \triangleq \sum_{i\in\{\text{all id in } A\}} w_i$; $w_C \triangleq \sum_{i\in\{\text{all id in } C\}} w_i$

$w \leftarrow 0$

`/* process updates or queries */`

case *upon receiving an update* (j, Δ) **do**

  $w \leftarrow w + \Delta$

  find node with id j in two of heaps A, B, C, D and update its w_j by adding Δ; update the corresponding key and maintain the heap properties

  if *the key of the top node in A is smaller than that of the top of B* **then**

    swap their tops and maintain heap properties

  if *the key of the top node of C is larger than that of the top of D* **then**

    swap their tops and maintain heap properties

  update π_A, π_C, w_A, w_C if necessary

case *upon receiving a query of the bias* β **do**

  return $\frac{w - w_A - w_C}{\|\boldsymbol{\pi}\|_1 - \pi_A - \pi_C}$

5. EXPERIMENTS

In this section we give our experimental studies.

5.1 The Setup

Reference Algorithms. We compare ℓ_1-S/R and ℓ_2-S/R with Count-Sketch (CS) algorithm and Count-Median Sketch (CM) algorithm, as well as non-linear sketches Count-Min with conservative update (CM-CU) [16, 20] and Count-Min-Log with conservative update (CML-CU) [27]. For CML-CU, we set the base to be 1.00025.

We would like to mention that the Count-Min algorithm, which was proposed in the same paper [11] as Count-Median, is very similar to Count-Median; they share the same sketching matrix. In fact, Count-Median can be thought as a generalization of Count-Min [11]. On the other hand, CM-CU is an improvement upon Count-Min and has strictly better performance. We thus do not compare our algorithms with Count-Min in our experiments.

Finally, we also compare ℓ_1-S/R and ℓ_2-S/R with two simple algorithms that just use the *mean* of *all* coordinates in $\mathbf{x}$ as the bias (denoted by ℓ_1-mean and ℓ_2-mean respectively). See sec. 5.4 for details. As mentioned earlier, using the mean of all the coordinates as the bias

Algorithm 6: Streaming Algorithm with ℓ_∞/ℓ_2 Guarantee

/* $s = c_s k$ `for a constant` $c_s \geq 4$; $d = \Theta(\log n)$;
$g, h^1, \ldots, h^d : [n] \to [s]$;
$r^1, \ldots, r^d : [n] \to \{-1, 1\}$ `are the same as in Algorithm 4;`
$\boldsymbol{\pi} \leftarrow \sum_{j \in [n]} j$`-th column of` $\Pi(g)$ */

$\forall i \in [d], \boldsymbol{\psi}^i \leftarrow$ coordinate-wise sum of columns of the CS-matrix $\Psi(h^i, r^i)$

initialize $\mathbf{y}^1, \ldots \mathbf{y}^d$ to be all-zero vectors of length s

initialize Bias-Heap in Algorithm 5 with s and $\boldsymbol{\pi}$

/* `process updates or queries` */

case *upon an update* (e_i, Δ) **do**

  $\forall t \in [d]$, $y^t_{h^t(i)} \leftarrow y^t_{h^t(i)} + r^t(i) \cdot \Delta$

  update the Bias-Heap with $(g(e_i), \Delta)$

case *upon receiving a query for computing* x_i **do**

  query Bias-Heap to get $\hat{\beta}$

  $z \leftarrow$ median $\left\{ r^t(i) \cdot \left(y^t_{h^t(i)} - \psi^t_{h^t(i)} \cdot \hat{\beta} \right) \;\middle|\; t \in [d] \right\}$

  return $z + \hat{\beta}$

cannot give us any theoretical guarantees – for example, it will perform badly on datasets where the top-k largest coordinates are significantly larger than all the rest coordinates. However, this simple heuristic does work well on some real-world datasets. Thus they may be interesting to practitioners.

Datasets. We compare the algorithms using a set of real and synthetic datasets.

- `Gaussian`. Each entry of $\mathbf{x}$ is independently sampled from the Gaussian distribution $\mathcal{N}(b, \sigma^2)$ where b is the *bias*. In our experiments, we fix $n = 500,000,000$, $\sigma = 15$ and vary the value of b.

- `Gaussian-2`. This dataset is used to compare ℓ_1-S/R, ℓ_2-S/R, ℓ_1-mean and ℓ_2-mean. Each entry of $\mathbf{x}$ is independently sampled from the Gaussian distribution $\mathcal{N}(100, 15^2)$. In our experiments, we fix $n = 5,000,000$. To verify our theorems, we shift several coordinates. See Figure 8 for details.

- `WorldCup` [2]. This dataset consists of all the requests made to all resources of the 1998 World Cup Web site between April 30, 1998 to July 26, 1998. We picked all the requests made to all resources on May 14, 1998. We construct $\mathbf{x}$ from those requests where each coordinate is the number of requests made in a particular second. The dimension of $\mathbf{x}$ is therefore $24 \times 3600 = 86,400$. There are about $3,200,000$ requests.

- `Wiki` [22]. This dataset contains pageviews to the English-language Wikipedia from March 16, 2015 to April 25, 2015. The number of pageviews of each second is recorded. We model the data as a vector $\mathbf{x}$ of length about $3,500,000$ (we added up mobile views and desktop views if they have the same timestamp). There are about $13,000,000,000$ pageviews.

- `Higgs` [3]. The dataset was produced by Monte Carlo simulations. There are 28 kinematic properties measured by the particle detectors in the accelerator. We model the fourth feature as a vector $\mathbf{x}$ of size $11,000,000$. The vector is non-negative.

- `Meme` [21]. This dataset includes memes from the `memetracker.org`. We model the vector $\mathbf{x}$ as the lengths of memes. Each coordinate of $\mathbf{x}$ can be thought as the number of words of a specific meme. The dimension of $\mathbf{x}$ is 210,999,824.

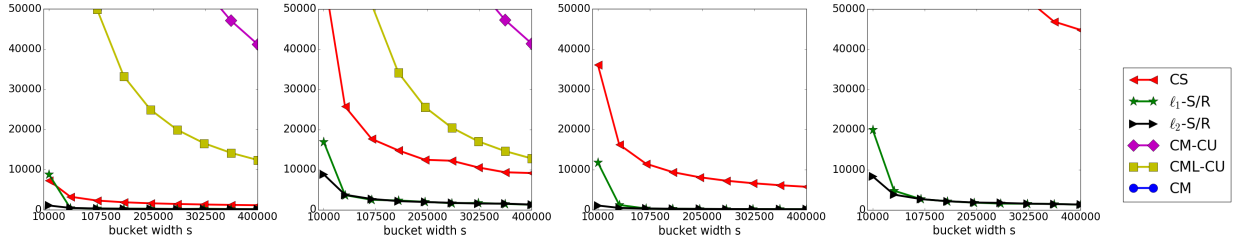
- `Hudong` [26]. There are "related to" links between articles of the Chinese online encyclopaedia Hudong.[3] This dataset contains about 2,452,715 articles, and 18,854,882 edges. Each edge (a, b) indicates that in article a, there is a "related to" link pointing to article b. Such links can be added or removed by users. We consider edges as a data stream, arriving in the order of editing time. Let $\mathbf{x}$ be the out-degree of those articles, and x_i is the number of "related to" links in article i. This dataset will be used to test our algorithms in the streaming model where we dynamically maintain a sketch for $\mathbf{x}$.

Measurements. We measure the effectiveness of the tested algorithms by the tradeoffs between sketch size and the recovery quality. We also compare the running time of these algorithms in the streaming setting.

For ℓ_1-S/R and ℓ_2-S/R, we use $d = 9$ copies of CS/CM-matrices of dimensions $s \times n$ (see Algorithm 1 and Algorithm 3). Theoretically we only need $O(\log n)$ extra words for ℓ_1-S/R to estimate the bias, but in our implementation we use s (typically much larger than $\log n$) extra words for both ℓ_1-S/R and ℓ_2-S/R, which makes it easier to compare the accuracies of ℓ_1-S/R and ℓ_2-S/R. Moreover, it also allows us to get more accurate and stable bias estimation for ℓ_1-S/R. For CM, CS, CM-CU and CML-CU, we set $d = 10$ so that all algorithms use $10s$ words. We will then vary s to get multiple sketch-size versus accuracy tradeoffs.

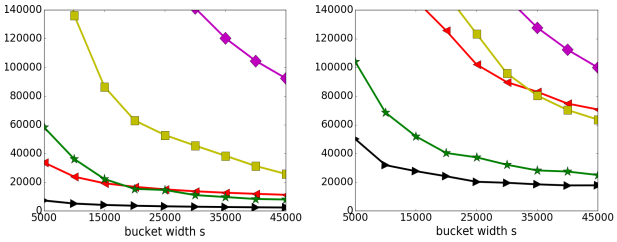
For point query we use the following two measurements: (1) average error $\frac{1}{n} \|\mathbf{x} - \hat{\mathbf{x}}\|_1$, and (2) maximum error $\|\mathbf{x} - \hat{\mathbf{x}}\|_\infty$. Recall that $\hat{\mathbf{x}}$ is the approximation of $\mathbf{x}$ given by the recovery scheme.

Computation Environments. All algorithms were implemented in C++. All experiments were run in a server with 32GB RAM and an Intel Xeon E5-2650 v2 8-core processor; the operating system is Red Hat Enterprise Linux 6.7.



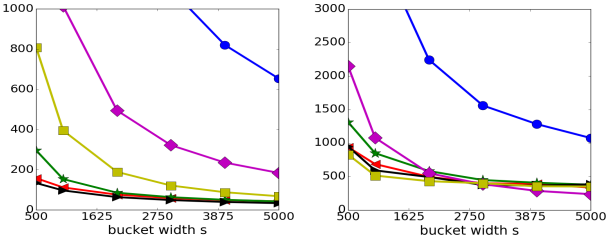
(a) $b = 100$, ave. error (b) $b = 100$, max. error (c) $b = 500$, ave. error (d) $b = 500$, max. error (e) Legend

Figure 1: `Gaussian` dataset; $n = 500,000,000$ and $\sigma = 15$. Some curves for CM, CM-CU, CML-CU cannot be presented since the errors are too large



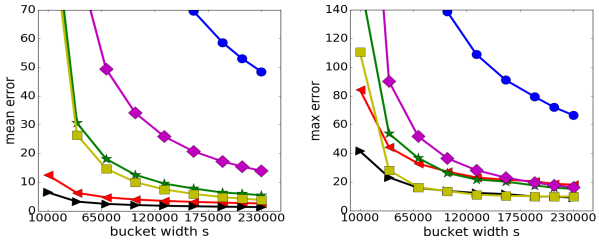
(a) Average error (b) Maximum error

Figure 2: `Wiki` dataset; $n = 3,513,600$. The curve for CM cannot be presented since the errors are too large



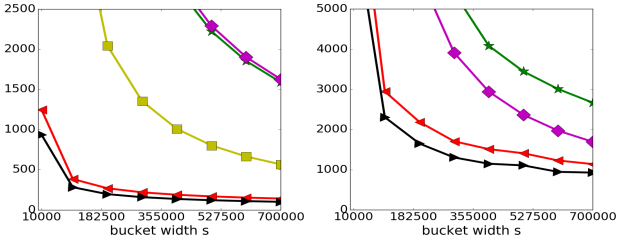
(a) Average error (b) Maximum error

Figure 3: `WorldCup` dataset; $n = 86,400$



(a) Average error (b) Maximum error

Figure 4: `Higgs` dataset; $n = 11,000,000$.



(a) Average error (b) Maximum error

Figure 5: `Meme` dataset; $n = 210,999,824$. Some curves for CM and CML-CU cannot be presented since the errors are too large

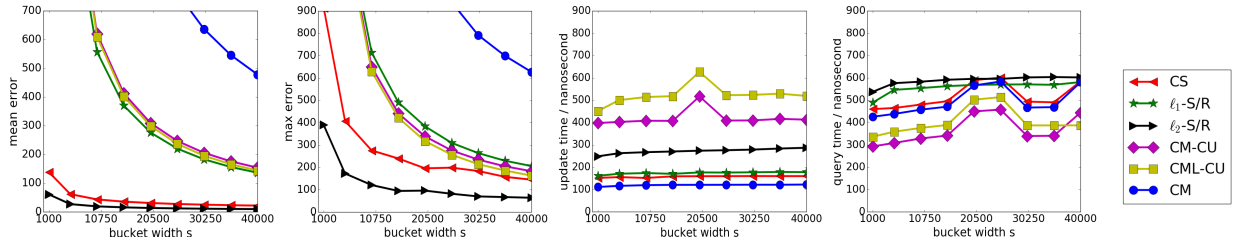
5.2 Accuracy for Point Query

`Gaussian` dataset with $n = 500,000,000$. Figure 1a and Figure 1b show the average and maximum errors of ℓ_1-S/R, ℓ_2-S/R, CM, CS, CM-CU and CML-CU respectively on `Gaussian` dataset with $n = 500$ million, $\sigma = 15$ and $b = 100$.

First note that ℓ_1-S/R and ℓ_2-S/R have similar average/maximum errors when we increase s. An explanation for this phenomenon is that in ℓ_2-S/R we use random signs $+1, -1$ to reduce/cancel the noise (contributed by colliding coordinates) in each hashing bucket, while in ℓ_1-S/R we do not. But in `Gaussian` the "perturbation" of each x_i around the bias is symmetric, and thus both algorithms achieve good cancellations. When s is small, the error of ℓ_2-S/R is slightly smaller than that of ℓ_1-S/R, this might because ℓ_1-S/R can not estimate the bias accurately. On the other hand, both ℓ_1-S/R and ℓ_2-S/R outperform other algorithms significantly. As a comparison, the errors of ℓ_1-S/R and ℓ_2-S/R are less than 1/5 of CS, 1/20 of CML-CU, 1/50 of CM-CU and 1/200 of CM.

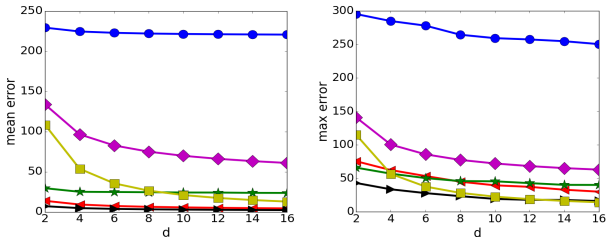
In Figure 1c and Figure 1d, we increase the value of b to 500. As we can observe from those figures that

[3] `http://www.hudong.com/`



(a) Average error (b) Maximum error (c) Update time (d) Query time (e) Legend

Figure 6: `Hudong` dataset; $n = 2,232,285$, there are $18,854,882$ updates in total



(a) Average error (b) Maximum error

Figure 7: `Higgs` dataset for fixed s; $n = 11,000,000$. We fix $s = 50000$ and vary d. The depth d here is for ℓ_1-S/R and ℓ_2-S/R; for CS, CM, CM-CU and CML-CU, the depth is $d + 1$.

the average and maximum errors of ℓ_1-S/R and ℓ_2-S/R are not affected by the value of b, which can be fully predicted from our theoretical results. On the contrary, the errors of CM, CS, CM-CU and CML-CU increase significantly when we increase b.

`Wiki` dataset. Figure 2 shows the accuracies of different algorithms on `Wiki`. We have observed that when varying the sketch size, ℓ_2-S/R always achieves the best recovery quality. For example, when sketch size is $s = 20,000$, the average error of ℓ_2-S/R is smaller than $1/10$ of the average errors of other algorithms. For average error, ℓ_1-S/R and CS perform similarly but the maximum error of CS is typically 2+ times larger than that of ℓ_1-S/R. The performance of CM, CM-CU and CML-CU are much worse than ℓ_1-S/R and ℓ_2-S/R.

`WorldCup` dataset. Figure 3 shows the accuracies of different algorithms on `WorldCup`. While ℓ_2-S/R still achieves the smallest average error, CS and ℓ_1-S/R follow closely. Again CM, CM-CU and CML-CU perform significantly worse than others. For maximum error, CS, CM-CU, CML-CU ℓ_1-S/R and ℓ_2-S/R have similar errors; CM gives significantly (typically 4+ times) larger errors than other algorithms.

`Higgs` dataset. Figure 4 shows the accuracies of different algorithms on `Higgs`. It can be observed that for average error, ℓ_2-S/R again achieves the smallest error. The average error of CS is typically larger than that of ℓ_2-S/R and is much smaller than that of other algorithms. For maximum error, CML-CU has similar accuracy as ℓ_2-S/R for large s. The maximum errors of all other algorithms are larger than that of ℓ_2-S/R. CM again has the worst performance.

`Meme` dataset. Figure 5 shows the accuracies of different algorithms on `Meme`. We can again observe that ℓ_2-S/R achieves the best recovery quality. The errors of CS are about 30% larger than that of ℓ_2-S/R. Both ℓ_2-S/R and CS outperform other algorithms significantly.

5.3 Effects of Sketch Depth

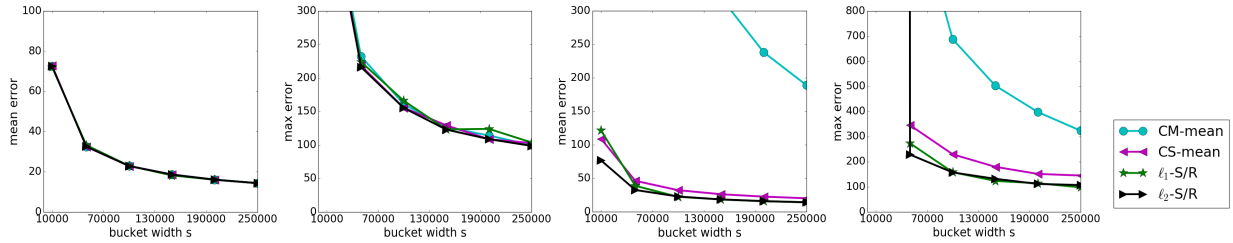
To see how the sketch depth d affects the accuracy of the sketch, we conduct experiments as follows: we fix the sketch size s and vary the sketch depth d. We only present the results for `Higgs` and similar results can be observed in other datasets.

It can be observed from Figure 7 that for all algorithms we tested, increasing d will improve the accuracy. One can also observe that CML-CU is more sensitive to the value of d than other algorithms. In terms of accuracy, ℓ_2-S/R still outperforms other algorithms and for the maximum error, CML-CU follows closely when d is large.

5.4 Comparisons with Mean Heuristics

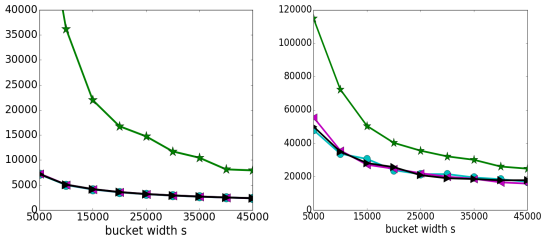
We also compare our algorithms with ℓ_1-mean and ℓ_2-mean. In Figure 8a-8b, we use the dataset whose entries are sampled from $\mathcal{N}(100, 15^2)$. It can be observed that all algorithms have similar performance and this is because all of ℓ_1-S/R, ℓ_2-S/R, ℓ_1-mean and ℓ_2-mean can estimate the bias ($b = 100$) well. In the dataset used in Figure 8c-8d, we shift 500 entries by $100,000$. A direct consequence is that the mean of the vector is no longer an accurate estimation of the bias. It can be observed that errors of both ℓ_1-mean and ℓ_2-mean increase significantly.

We also conduct experiments on `Wiki` dataset, one can observe that ℓ_2-S/R, ℓ_1-mean and ℓ_2-mean have similar performance and all of them outperform ℓ_1-S/R.



(a) Average error (b) Maximum error (c) Average error (d) Maximum error (e) Legend

Figure 8: `Gaussian-2` dataset; Fig. 8a-8b, the dataset is not shifted. Fig. 8c-8d, 500 entries are shifted by $100,000$



(a) Average error (b) Maximum error

Figure 9: `Wiki` dataset

5.5 Distributed and Streaming Implementations

As mentioned in the introduction, it is straightforward to implement our bias-aware sketches in the distributed model by making use of the linearity. Moreover, their performance in the distributed model can be *fully* predicted by the centralized counterparts – the total communication will just be the number of sites times the size of the sketch, and the time costs at the sites and the coordinator will be equal to the sketching time and recovery time respectively.[4] Therefore, our experiments in the centralized model can also speak for that in the distributed model.

We implemented our bias-aware sketches in the streaming model. We have run our algorithms on the streaming dataset `Hudong` where edges are added in the streaming fashion. We update the sketch at each step, and recover the entire $\hat{\mathbf{x}}$ after feeding in the whole dataset. To measure the running time, we first process the whole data stream and calculate the average update time. We then recover the whole vector and calculate the average query time.

[4]Regarding the random hash functions, the coordinator can simply generate $g, h^1, \ldots, h^d : [n] \to [s]$; $r^1, \ldots, r^d : [n] \to \{-1, 1\}$ at the beginning and send to each site, which only incurs an extra of $O(\log n)$ communication on each channel and is thus negligible.

Accuracy for Point Query. Figure 6a and Figure 6b show that the recovery errors of CS are 2+ times larger than that of ℓ_2-S/R. The others algorithms have even larger errors. In both Figures the results of CML-CU and CM-CU are very close and their curves overlap with each other. The performance of ℓ_1-S/R is also quite similar to CML-CU and CM-CU.

Update/Recover Running Time. It can be seen from Figure 6c and Figure 6d that all of the six tested algorithms have similar processing time per update and per point query. The time cost per update of ℓ_1-S/R is about 50% more than CM, and that of ℓ_2-S/R is within a factor of 2 of CS. We thus conclude that the additional components (such as the Bias-Heap) used in ℓ_1-S/R and ℓ_2-S/R only generate small overheads.

5.6 Summary of Experimental Results

We now summarize our experimental results. We have observed that in terms of recovery quality, ℓ_1-S/R strictly outperforms CM, and ℓ_2-S/R strictly outperforms CS. In general ℓ_2-S/R is much better than ℓ_1-S/R, especially when the noise around the bias is not symmetric. Note that this is similar to the phenomenon that the error of CS is almost always smaller than that of CM in practise, and is consistent to the theoretical fact that if $n \gg k$, and the tail coordinates of $\mathbf{y} = \mathbf{x} - \beta^{(n)}$ follows some long tail distribution, than the error $\frac{1}{k}\mathrm{Err}_1^k(\mathbf{y})$ is much larger than $\frac{1}{\sqrt{k}}\mathrm{Err}_2^k(\mathbf{y})$.

In almost all datasets we have tested, ℓ_2-S/R outperforms CML-CU and CM-CU, the latter two are considered as improved versions of the Count-Min sketch.

The sketch depth d also affects the accuracy of a sketch. Larger d leads to better performance. It is also observed that some algorithms (e.g. CML-CU) are more sensitive to d than others.

As for running time (update/query), the differences between ℓ_1-S/R, ℓ_2-S/R, CS, CM, CM-CU and CML-CU are not significant. The overhead introduced by the components used to estimate the bias is fairly low in both ℓ_1-S/R and ℓ_2-S/R.

6. CONCLUSION

In this paper we formulated the bias-aware sketching and recovery problem, and proposed two algorithms that strictly generalize the widely used Count-Sketch and Count-Median algorithms. Our bias-aware sketches, due to their linearity, can be easily implemented in the streaming and distributed computation models. We have also verified their effectiveness experimentally, and showed the advantages of our bias-aware sketches over Count-Sketch, Count-Median and the improved versions of Count-Min in both synthetic and real-world datasets.

7. ACKNOWLEDGMENT

Jiecao Chen and Qin Zhang are supported in part by NSF CCF-1525024 and IIS-1633215.

8. REFERENCES

[1] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *STOC*, pages 20–29. ACM, 1996.

[2] M. Arlitt and T. Jin. World cup web site access logs, august 1998. *URL http://ita. ee. lbl. gov/html/contrib/WorldCup. html*, 1998.

[3] P. Baldi, P. Sadowski, and D. Whiteson. Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5, 2014.

[4] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. In *FOCS*, pages 209–218, 2002.

[5] E. J. Candès, J. K. Romberg, and T. Tao. Robust uncertainty principles: exact signal reconstruction from highly incomplete frequency information. *IEEE Transactions on Information Theory*, 52(2):489–509, 2006.

[6] M. Charikar, K. C. Chen, and M. Farach-Colton. Finding frequent items in data streams. In *ICALP*, pages 693–703, 2002.

[7] G. Cormode. Sketch techniques for approximate query processing. *Foundations and Trends in Databases. NOW publishers*, 2011.

[8] G. Cormode and M. Garofalakis. Sketching streams through the net: Distributed approximate query tracking. In *VLDB*, pages 13–24. VLDB Endowment, 2005.

[9] G. Cormode and M. Hadjieleftheriou. Methods for finding frequent items in data streams. *VLDB J.*, 19(1):3–20, 2010.

[10] G. Cormode, T. Johnson, F. Korn, S. Muthukrishnan, O. Spatscheck, and D. Srivastava. Holistic udafs at streaming speeds. In *SIGMOD*, pages 35–46. ACM, 2004.

[11] G. Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, 2005.

[12] G. Cormode and S. Muthukrishnan. Combinatorial algorithms for compressed sensing. In *SIROCCO*, pages 280–294, 2006.

[13] F. Deng and D. Rafiei. New estimation algorithms for streaming data: Count-min can do more. Technical report, 2007.

[14] D. L. Donoho. Compressed sensing. *IEEE Transactions on Information Theory*, 52(4):1289–1306, 2006.

[15] D. L. Donoho, M. Elad, and V. N. Temlyakov. Stable recovery of sparse overcomplete representations in the presence of noise. *IEEE Trans. Information Theory*, 52(1):6–18, 2006.

[16] C. Estan and G. Varghese. New directions in traffic measurement and accounting. *Computer Communication Review*, 32(1):75, 2002.

[17] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31(2):182–209, 1985.

[18] A. C. Gilbert and P. Indyk. Sparse recovery using sparse matrices. *Proceedings of the IEEE*, 98(6):937–947, 2010.

[19] A. C. Gilbert, S. Muthukrishnan, and M. Strauss. Approximation of functions over redundant dictionaries using coherence. In *SODA*, pages 243–252, 2003.

[20] A. Goyal, H. D. III, and G. Cormode. Sketch algorithms for estimating point queries in NLP. In *EMNLP-CoNLL*, pages 1093–1103, 2012.

[21] J. F. E. IV, F. Fogelman-Soulié, P. A. Flach, and M. J. Zaki, editors. *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009*. ACM, 2009.

[22] O. Keyes. Wiki-Pageviews, english wikipedia pageviews by second. `http://datahub.io/dataset/english-wikipedia-pageviews-by-second`, April, 2015.

[23] Y. Lu, A. Montanari, B. Prabhakar, S. Dharmapurikar, and A. Kabbani. Counter braids: a novel counter architecture for per-flow measurement. In *SIGMETRICS*, pages 121–132, 2008.

[24] S. Melnik, A. Gubarev, J. J. Long, G. Romer, S. Shivakumar, M. Tolton, and T. Vassilakis.

Dremel: interactive analysis of web-scale datasets. *Communications of the ACM*, 54(6):114–123, 2011.

[25] R. Morris. Counting large numbers of events in small registers. *Communications of the ACM*, 21(10):840–842, 1978.

[26] X. Niu, X. Sun, H. Wang, S. Rong, G. Qi, and Y. Yu. Zhishi.me – weaving Chinese linking open data. In *Proc. Int. Semantic Web Conf.*, pages 205–220, 2011.

[27] G. Pitel and G. Fouquier. Count-Min-Log sketch: Approximately counting with approximate counters. *ArXiv e-prints*, Feb. 2015.

[28] D. Van Gucht, R. Williams, D. P. Woodruff, and Q. Zhang. The communication complexity of distributed set-joins with applications to matrix multiplication. In *PODS*, pages 199–212. ACM, 2015.

[29] Y. Yan, J. Zhang, B. Huang, X. Sun, J. Mu, Z. Zhang, and T. Moscibroda. Distributed outlier detection using compressive sensing. In *SIGMOD*, pages 3–16. ACM, 2015.