Polylogarithmic Sketches for Clustering

Moses Charikar
Stanford University, CA, USA

Erik Waingarten
Stanford University, CA, USA

Abstract

Given n points in ℓ_p^d, we consider the problem of partitioning points into k clusters with associated centers. The cost of a clustering is the sum of p^{th} powers of distances of points to their cluster centers. For $p \in [1, 2]$, we design sketches of size $\text{poly}(\log(nd), k, 1/\epsilon)$ such that the cost of the optimal clustering can be estimated to within factor $1 + \epsilon$, despite the fact that the compressed representation does not contain enough information to recover the cluster centers or the partition into clusters. This leads to a streaming algorithm for estimating the clustering cost with space $\text{poly}(\log(nd), k, 1/\epsilon)$. We also obtain a distributed memory algorithm, where the n points are arbitrarily partitioned amongst m machines, each of which sends information to a central party who then computes an approximation of the clustering cost. Prior to this work, no such streaming or distributed-memory algorithm was known with sublinear dependence on d for $p \in [1, 2)$.

2012 ACM Subject Classification Theory of computation → Sketching and sampling

Keywords and phrases sketching, clustering

Digital Object Identifier 10.4230/LIPIcs.ICALP.2022.38

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2204.12358`

Funding *Moses Charikar*: Supported by a Simons Investigator Award.
Erik Waingarten: Supported by the National Science Foundation under Award no. 2002201.

1 Introduction

Given a large number of high-dimensional points, is it possible to compress the raw representation into a very compact sketch so that we can understand how clusterable the data is from just this highly compressed representation? Given n points in d dimensions, we consider the problem of approximating the cost of clustering them into k clusters from a compressed representation whose size is polylogarithmic in both n and d.

For $n, d \in \mathbb{N}$, let $P = \{x_1, \ldots, x_n\} \in \mathbb{R}^d$ be any set of points with polynomially bounded entries (i.e., all coordinates may be represented with $O(\log(nd))$ bits). The (k, z)-clustering problem in ℓ_p^d, asks to partition P into at most k clusters $C_1, \ldots, C_k$ so as to minimize

$$\sum_{\ell=1}^{k} \min_{c_\ell \in \mathbb{R}^d} \sum_{x \in C_\ell} \|x - c_\ell\|_p^z. \tag{1}$$

The problem is a generalization of the k-means and k-median problem; in particular, in Euclidean space ($p = 2$), $z = 2$ corresponds to k-means, and $z = 1$ to k-median.

We note that the raw representation of the dataset uses $O(nd \log(nd))$ bits, and that any algorithm which outputs optimal cluster centers $c_1, \ldots, c_k \in \mathbb{R}^d$, or the optimal clustering $C_1, \ldots, C_k$ must utilize $\Omega(kd)$, or $\Omega(n \log k)$ bits of space, respectively. Hence, such algorithms cannot decrease the dependency on both n and d simultaneously. However, this does not rule out an exponential compression, from $O(nd \log(nd))$ bits to $\text{polylog}(nd)$ bits (for constant k and z), for algorithms which approximate the optimal clustering *cost*, which only needs $O(\log(nd))$ bits. In this work, we show that it is indeed possible to design sketches of size

poly$(\log(nd), 1/\epsilon)$ bits which ϵ-approximate the optimal clustering *cost*, despite the fact that we do not have enough information to compute the clusters nor the cluster centers which achieve such cost.

Our results fit into a line of prior work on approximating the cost of optimization problems without necessarily computing an optimal solution. These have been investigated before for various problems and in various contexts, including estimating minimum spanning tree [40, 31, 33], minimum cost matchings and Earth Mover's Distance [40, 5, 3, 13], minimum vertex cover and maximum matching [59, 56, 62, 57, 26, 47, 15] and model-fit [49, 48, 17]. Specifically for clustering problems, the value of the clustering cost is an important statistic; used, for example, in the "elbow method" for determining the number of clusters needed. We will show these sketches may be efficiently maintained on a stream as well as for distributed-memory models, implying polylog(nd)-bit algorithms for these models of computation.

We start by reviewing a set of techniques in the literature to either reduce the dependence on the data set size or the dependence on the dimension.

Coresets

The coreset technique is a "dataset compression" mechanism, aiming to reduce the dependency on n. From the n points $P \subset \mathbb{R}^d$, an algorithm computes a much smaller (weighted) set of points $S \subset \mathbb{R}^d$, $w\colon S \to \mathbb{R}_{\geq 0}$, such that the cost of clustering the weighted points S, w approximates that of P. Following a long line of work [11, 36, 1, 24, 52, 34, 35, 20, 61, 39, 29], the best coreset constructions for (k, z)-clustering in ℓ_p achieve sizes $\tilde{O}(k/\epsilon^4) \cdot \min\{1/\epsilon^{z-2}, k\}$ for a $(1 \pm \epsilon)$-approximation. The ensuing streaming and distributed-memory algorithms maintain a coreset of the input; these algorithms find (approximately) optimal centers $c_1, \ldots, c_k \in \mathbb{R}^d$ and use space complexity $d \cdot \tilde{O}(k/\epsilon^4) \cdot \min\{1/\epsilon^{z-2}, k\} \cdot \text{polylog}(n)$.[1]

Dimension Reduction and Sketching

In addition to constructing coresets, an algorithm may seek to optimize the dependence on d. There is a large body of work studying (oblivious) dimensionality reduction and sketching, where strong compression results are known for computing distances [2, 51, 60, 12, 23, 30, 42, 41, 44, 46, 7, 8, 18]. For example, for $p \in [1, 2]$ there exists a (randomized) sketch $\mathbf{sk}\colon \mathbb{R}^d \to \mathbb{R}^t$ with t much smaller than d such that, for any two vectors $x, y \in \mathbb{R}^d$, an algorithm can approximate $\|x - y\|_p$ from $\mathbf{sk}(x)$ and $\mathbf{sk}(y)$ with high probability. While these results are encouraging, leveraging such sketches for distance computation in order to compress entire optimization problems like (1) is highly nontrivial. The challenge is that (1) implicitly considers distances among infinitely many vectors, and we need to rule out the possibility of spurious low cost solutions in the "sketched" space which do not have an analog in the original space. In particular, prior to this work, no streaming or distributed-memory algorithm was known which could reduce the dependence on d for $p \in [1, 2)$.

There is one setting, of Euclidean space ($p = 2$), where one can sketch vectors while preserving (1). A sequence of works [19, 27, 14, 55] show that applying a Johnson-Lindenstrauss [45] map $\mathbf{\Pi}\colon \mathbb{R}^d \to \mathbb{R}^t$ with $t = O(z^4 \log(k/\epsilon)/\epsilon^2)$, sketches Euclidean vectors to $O(t \log(nd))$ bits and preserves (1) up to $1 \pm \epsilon$. We emphasize that Euclidean space $p = 2$ is special in this regard, because the Johnson-Lindenstrauss map achieves *dimension reduction*, a property known not to hold in ℓ_1 [22, 54, 4]. In particular, d-dimensional vectors $x \in \mathbb{R}^d$ in Euclidean space are sketched to vectors $\mathbf{\Pi}(x) \in \mathbb{R}^t$ in Euclidean space, i.e., one estimates

[1] The $\log n$-factors arise from utilizing the "merge-and-reduce" framework for maintaining coresets on a stream [16, 1], and the fact the coreset constructions are randomized.

$\|x-y\|_2$ by $\|\mathbf{\Pi}(x)-\mathbf{\Pi}(y)\|_2$. Thus the optimization problem (1) for d-dimensional Euclidean space reduces to the same optimization problem for a much smaller dimensional Euclidean space. This can therefore be composed with known coreset constructions. Importantly, the "sketched" space inherits all geometric properties of Euclidean spaces, a key aspect of prior works, and the reason they do not extend beyond Euclidean space. The technical challenge in applying sketches for ℓ_p when $p \neq 2$ is that the "sketched" space is non-geometric.[2]

1.1 Our results

We give a streaming and distributed-memory algorithm for (k,p)-clustering in ℓ_p with space complexity $\mathrm{poly}(\log(nd), k, 1/\epsilon)$ bits.

▶ **Theorem 1** (Streaming (k,p)-Clustering in ℓ_p)**.** *For $p \in [1,2]$, there exists an insertion-only streaming algorithm which processes a set of n points $x_1,\ldots,x_n \in \mathbb{R}^d$ utilizing $\mathrm{poly}(\log(nd), k, 1/\epsilon)$ bits which outputs a parameter $\boldsymbol{\eta} \in \mathbb{R}$ satisfying*

$$(1-\epsilon)\min_{\substack{C_1,\ldots,C_k\\ \text{partition } [n]}}\sum_{\ell=1}^{k}\min_{c_\ell\in\mathbb{R}^d}\sum_{i\in C_\ell}\|x_i-c_\ell\|_p^p \le \boldsymbol{\eta} \le (1+\epsilon)\min_{\substack{C_1,\ldots,C_k\\ \text{partition } [n]}}\sum_{\ell=1}^{k}\min_{c_\ell\in\mathbb{R}^d}\sum_{i\in C_\ell}\|x_i-c_\ell\|_p^p$$

with probability at least 0.9*.*

▶ **Theorem 2** (Distributed-Memory (k,p)-Clustering in ℓ_p)**.** *For $p \in [1,2]$, there exists a public-coin protocol where m machines receive an arbitrary partition of n points $x_1,\ldots,x_n \in \mathbb{R}^d$, each communicates $\mathrm{poly}(\log(md), k, 1/\epsilon)$ bits to a central authority who outputs a parameter $\boldsymbol{\eta} \in \mathbb{R}$ satisfying*

$$(1-\epsilon)\min_{\substack{C_1,\ldots,C_k\\ \text{partition } [n]}}\sum_{\ell=1}^{k}\min_{c_\ell\in\mathbb{R}^d}\sum_{i\in C_\ell}\|x_i-c_\ell\|_p^p \le \boldsymbol{\eta} \le (1+\epsilon)\min_{\substack{C_1,\ldots,C_k\\ \text{partition } [n]}}\sum_{\ell=1}^{k}\min_{c_\ell\in\mathbb{R}^d}\sum_{i\in C_\ell}\|x_i-c_\ell\|_p^p$$

with probability at least 0.9*.*

Both algorithms will follow from applying a coreset and compressing the representation of the coreset points into sketches to recover single-cluster cost. Specifically, the bottleneck for our algorithm will be estimating the cost of (k,p)-clustering in ℓ_p for $k=1$. We give a linear sketch such that given a set of points $x_1,\ldots,x_n \in \mathbb{R}^d$, one may approximate the ℓ_p^p-median cost:

$$\min_{y\in\mathbb{R}^d}\sum_{i=1}^{n}\|x_i-y\|_p^p.$$

Most of the technical work will be devoted to sketching this "ℓ_p^p-median cost" objective. Then, the streaming and distributed-memory algorithm will evaluate the sum of ℓ_p^p-median costs for all possible partitions of the coreset points into k parts. The following theorem gives a linear sketch for approximating the ℓ_p^p-median cost.[3]

[2] For example, the sketched space for ℓ_p with $p \neq 2$ does not satisfy the triangle inequality: it is not the case that for any $x,y,z \in \mathbb{R}^d$, the estimate of $(\mathbf{sk}(x),\mathbf{sk}(y))$ plus the estimate of $(\mathbf{sk}(y),\mathbf{sk}(z))$ is less than the estimate of $(\mathbf{sk}(x),\mathbf{sk}(z))$. On the other hand, for $p=2$, the estimates of $(\mathbf{sk}(x),\mathbf{sk}(y))$, $(\mathbf{sk}(y),\mathbf{sk}(z))$, and $(\mathbf{sk}(x),\mathbf{sk}(z))$ are $\|\mathbf{sk}(x)-\mathbf{sk}(y)\|_2$, $\|\mathbf{sk}(y)-\mathbf{sk}(z)\|_2$, and $\|\mathbf{sk}(x)-\mathbf{sk}(z)\|_2$, so the triangle inequality does hold in the sketched space.

[3] A related although different work is that of approximating the ℓ_p^p-median (for instance, see Appendix F of [10]). An ℓ_p^p-median is a vector in $\mathbb{R}^d$ which means the sketch outputs d numbers; however, we will sketch the ℓ_p^p-median cost, which is a real number. Hence, our sketch will use $\mathrm{poly}(\log(nd), 1/\epsilon)$ space, as opposed to $\Omega(d)$ space needed to describe an ℓ_p^p-median.

▶ **Theorem 3** (ℓ_p^p-Median Sketch)**.** *For $p \in [1,2]$, there exists a linear sketch which processes a set of n points $x_1, \ldots, x_n \in \mathbb{R}^d$ into a vector $\mathbb{R}^t$ with $t = \text{poly}(\log(nd), 1/\epsilon)$ and outputs a parameter $\boldsymbol{\eta} \in \mathbb{R}$ satsifying*

$$(1-\epsilon) \min_{y \in \mathbb{R}^d} \sum_{i=1}^{n} \|x_i - y\|_p^p \leq \boldsymbol{\eta} \leq (1+\epsilon) \min_{y \in \mathbb{R}^d} \sum_{i=1}^{n} \|x_i - y\|_p^p$$

with probability at least 0.9*.*

There are a few important remarks to make:

- The requirement that $p \leq 2$ is necessary for the exponential compression we desire. For $p > 2$, there are strong lower bounds for sketching distances which show that such sketches require $\Omega(d^{1-2/p})$ space [12]. For $p < 1$, we are not aware of small coresets.
- The focus of this work is on optimizing the space complexity of the sketch, and while we do not explicitly specify the running time of the sketching and streaming algorithms, a naive implementation runs in time $(k \log(nd)/\epsilon)^{(k \log n/\epsilon)^{O(1)}}$. The exponential factor is due to the fact that we evaluate the cost of all possible partitions of the $(k \log(n)/\epsilon)^{O(1)}$-coreset points into k clusters. One could alleviate the exponential dependence to $(k/\epsilon)^{O(1)}$ (as opposed to $(k \log n/\epsilon)^{O(1)}$) by running more sophisticated approximation algorithms [11, 50] on the sketched representation of the coreset.[4] We note that a super-polynomial dependence on k should is unavoidable, because $(1 \pm \epsilon)$-approximations for (k, z)-clustering problems, for *non-constant* k, are NP-hard [9, 53, 28].
- It would be interesting to generalize Theorem 1 to dynamic streams. The reason our algorithm works in the insertion-only model is that we utilize the coreset of [39] with the merge-and-reduce framework [16, 1] which do not support deletions. While there exist dynamic coreset constructions for the streaming model [21, 38], our use of coresets is not entirely black-box. Other dynamic coresets, like [37], focus on update time and do not optimize the space complexity. We must ensure that the algorithm for constructing the coreset does not utilize the d-dimensional representation of the dataset points. The coreset construction of [39] only consider distances between the dataset points, so it suffices for us to only maintain a sketch of the dataset points.
- The fact that $z = p$ in our theorems above is a consequence of our techniques. It is unclear to us whether this assumption is necessary, although our approach hinges on the fact ℓ_p^p is additive over the d coordinates. We leave this as a problem for future work.

A similar, yet importantly different notion of (k, z)-clustering considers *medoid* cost, where the centers of the k clusters $c_1, \ldots, c_k$ are restricted to be dataset points. While seemingly similar to the (k, z)-clustering objective where centers are unrestricted, these two are qualitatively very different from a sketching perspective. In the full version, we show that while a two-pass sketching algorithm may ϵ-approximate the medoid cost, ϵ-approximations for one-pass sketching algorithm require polynomial space.

1.2 Technical Overview

We give an overview of Theorem 3. Once that is established, combining the ℓ_p^p-median sketch with coresets, thereby establishing Theorems 1 and 2 is (relatively) straight-forward (for more details, we refer the reader to the full version). Recall that for $p \in [1,2]$, we will process n points $x_1, \ldots, x_n \in \mathbb{R}^d$, and aim to return an approximation to the ℓ_p^p-median cost:

[4] The one subtlety is that the algorithm should be implemented without explicitly considering the d-dimensional representation of the points. Instead, it should only use the sketches of Theorem 3.

$$\min_{c\in\mathbb{R}^d}\sum_{i=1}^{n}\|x_i-c\|_p^p. \tag{2}$$

It will be useful to assume that the points are centered, i.e., $\sum_{i=1}^n x_i = 0 \in \mathbb{R}^d$ (we can enforce this because our sketches will be linear). The approach will come from the fact that the above optimization problem decomposes into a sum of d independent optimizations, one for each coordinate, and (2) seeks to evaluate the sum. Specifically, we may write

$$\min_{c\in\mathbb{R}^d}\sum_{i=1}^{n}\|x_i-c\|_p^p = \sum_{j=1}^{d}\min_{c_j\in\mathbb{R}}\sum_{i=1}^{n}|x_{ij}-c_j|^p.$$

Furthermore, for any fixed $j\in[d]$, estimating

$$\min_{c_j\in\mathbb{R}}\sum_{i=1}^{n}|x_{ij}-c_j|^p \tag{3}$$

is much more amenable to ℓ_p-sketching. Specifically, we let $x_{\cdot,j}\in\mathbb{R}^n$ be the vector containing the j-th coordinates of all n points, and $\mathbf{1}\in\mathbb{R}^n$ be the all-1's vector. Then, the quantity $\sum_{i=1}^n |x_{ij}-c_j|^p = \|x_{\cdot,j}-c_j\mathbf{1}\|_p^p$, and since the ℓ_p-sketches are linear, an algorithm may maintain $\mathbf{sk}(x_{\cdot,j})\in\mathbb{R}^t$ (for $t=\mathrm{poly}(\log(nd))$) and after processing, could iterate through various values of $c_j\in\mathbb{R}$ to evaluate

$$\left(\sum_{i=1}^{n}|x_{ij}-c_j|^p\right)^{1/p} = \|x_{\cdot,j}-c_j\mathbf{1}\|_p \approx_{1\pm\epsilon} \text{ estimate of } (\mathbf{sk}(x_{\cdot,j}),\mathbf{sk}(c_j\mathbf{1})),$$

and output the smallest value of $c_j\in\mathbb{R}$ found. In order to guarantee an $(1\pm\epsilon)$-approximation of (3), only $\mathrm{poly}(1/\epsilon)$ values of c_j need to be tried (since first evaluating $c_j=0$ will specify the range where the optimal c_j may lie). A simple union bound implies that for any fixed $j\in[d]$, we can prepare a small sketch $\mathbf{sk}(x_{\cdot,j})$ from which we can approximate (3).

In summary, we want to estimate the sum of d minimization problems. Even though each of the d problems could be solved independently with a linear sketch, we do not want to process d linear sketches (as this increases space). In addition, we do not know which of the d minimizations will significantly affect the sum; hence, if we only (uniformly) sampled few $\boldsymbol{j}_1,\ldots,\boldsymbol{j}_t\sim[d]$ and only processed $t\ll d$ sketches along the sampled dimensions, the variance of the estimator may be too large, making it completely useless. The technique we will use was recently developed in [25], building on [6, 43], under the name "ℓ_p-sampling with meta-data." In this paper, we further develop the ideas, and apply them to sketches for clustering in a simple and modular way. We refer the reader to Remark 4 (following this technical overview), where we expand on the comparison to [25].

The goal is to approximate the sum of the d minimization problems by *importance sampling* (see Chapter 9 of [58]). While importance sampling is a well-known technique, it's use in (one-pass) linear sketching algorithms is counter-intuitive, and we are not aware of any linear sketches which use importance sampling in the literature, expect for this and the recent work of [25, 32]. Importance sampling will aim to estimate (2) by sampling with respect to an alternate distribution $\mathcal{D}$. In particular, (2) may be re-written as

$$d\cdot\mathop{\mathbf{E}}_{\boldsymbol{j}\sim[d]}\left[\min_{c_{\boldsymbol{j}}\in\mathbb{R}}\sum_{i=1}^{n}|x_{i\boldsymbol{j}}-c_{\boldsymbol{j}}|^p\right] = d\mathop{\mathbf{E}}_{\boldsymbol{j}\sim\mathcal{D}}[\mathbf{Y}_{\boldsymbol{j}}] \qquad \text{where}$$

$$\mathbf{Y}_{\boldsymbol{j}} \stackrel{\text{def}}{=} \min_{c_{\boldsymbol{j}}\in\mathbb{R}}\sum_{i=1}^{n}|x_{i\boldsymbol{j}}-c_{\boldsymbol{j}}|^p\cdot\frac{1}{\mathbf{Pr}_{\mathcal{D}}[\boldsymbol{j}]}, \tag{4}$$

where $\mathcal{D}$ is a distribution chosen so the variance of the random variable $\mathbf{Y}_{\boldsymbol{j}}$ for $\boldsymbol{j} \sim \mathcal{D}$ is bounded. Once the variance of the random variable is bounded, only a few samples are needed to estimate its expectation in (4). In general, the alternate distribution $\mathcal{D}$ depends on the data in order to decrease the variance; for instance, coordinates $j \in [d]$ whose value of (3) is higher should be sampled more often. Hence, importance sampling inherently interacts with the data in a two-stage process: 1) first, it samples $\boldsymbol{j} \sim \mathcal{D}$ (where the distribution is data-dependent), and 2) second, it evaluates $\mathbf{Y}_{\boldsymbol{j}}$ by using (3) and $\mathbf{Pr}_{\mathcal{D}}[\boldsymbol{j}]$ for the value $\boldsymbol{j} \in [d]$ specified in the first step.

In a two-pass algorithm, the two steps may be implemented sequentially. A *sampling* sketch, like that of [43], is used to sample $\boldsymbol{j} \sim \mathcal{D}$ in the first pass. In the second pass, the algorithm knows the value of the sampled $\boldsymbol{j}$, so it maintains a sketch $\mathbf{sk}(x_{\cdot,\boldsymbol{j}})$ of size t and a sketch $\mathbf{sk}'(\mathbf{Pr}_{\mathcal{D}}[\boldsymbol{j}])$ of size t' (to estimate $\mathbf{Pr}_{\mathcal{D}}[\boldsymbol{j}]$) from which it can evaluate the random variable $\mathbf{Y}_{\boldsymbol{j}}$. The counter-intuitive aspect is that, in this case, we will perform both steps in one-pass:

- We will use an ℓ_p-sampling sketch of [43] to sample from an importance sampling distribution $\mathcal{D}$, and
- Concurrently, we prepare $2d$ linear sketches: d sketches $\mathbf{sk}(x_{\cdot,j})$ to evaluate (3), one for each $j \in [d]$, and d sketches $\mathbf{sk}'(\mathbf{Pr}_{\mathcal{D}}[j])$ to evaluate $\mathbf{Pr}_{\mathcal{D}}[j]$, one for each $j \in [d]$. The non-trivial part is to *sketch the sketches*. by compressing the $2d$ linear sketches into a $O(\text{polylog}(nd))$-bit Count-Min data structure [30].

The guarantee will be that the ℓ_p-sampling sketch of [43] generates a sample $\boldsymbol{j} \sim \mathcal{D}$, and the Count-Min data structure can recover an approximation

$$\widehat{\mathbf{sk}_1} \approx \mathbf{sk}(x_{\cdot,\boldsymbol{j}}) \qquad \text{and} \qquad \widehat{\mathbf{sk}_2} \approx \mathbf{sk}'(\mathbf{Pr}_{\mathcal{D}}[\boldsymbol{j}]).$$

Furthermore, the sketch evaluation algorithm, which executes on the approximation $\widehat{\mathbf{sk}_1}$ and $\widehat{\mathbf{sk}_2}$, should be able to recover $(1 \pm \epsilon)$-approximations to (3) and $\mathbf{Pr}_{\mathcal{D}}[\boldsymbol{j}]$, so that the ratio of the two is a $(1 \pm 2\epsilon)$-approximation to $\mathbf{Y}_{\boldsymbol{j}}$.

While the above plan provides a general recipe for importance sampling, the idea of "sketching the sketches" may not be applied in a black-box manner. First, the alternate distribution $\mathcal{D}$ should admit a sampling sketch. Second, the sketch evaluation algorithm for $\mathbf{sk}(x_{\cdot,j})$ and $\mathbf{sk}'(\mathbf{Pr}_{\mathcal{D}}[j])$ should be robust to the errors introduced by the Count-Min compression. Bounding the errors introduced by the Count-Min data structure, and ensuring that the approximate sketches $\widehat{\mathbf{sk}_1}$ and $\widehat{\mathbf{sk}_2}$ constitutes the bulk of the technical work. Specifically for us, the plan is executed as follows: when $\sum_{i=1}^{n} x_i = 0 \in \mathbb{R}^d$, every j satisfies (we refer the reader to the full version for a thorough justification)

$$\frac{\min_{c_j \in \mathbb{R}} \sum_{i=1}^{n} |x_{ij} - c_j|^p}{\|x_{\cdot,j}\|_p^p} \in [2^{-p}, 1]. \tag{5}$$

Hence, we will let $\mathcal{D}$ be the distribution supported on $[d]$ given by setting, for each $j \in [d]$,

$$\Pr_{\boldsymbol{j} \sim \mathcal{D}}[\boldsymbol{j} = j] = \frac{\|x_{\cdot,j}\|_p^p}{Z} \qquad \text{where} \qquad Z = \sum_{j=1}^{d} \|x_{\cdot,j}\|_p^p = \sum_{i=1}^{n} \sum_{j=1}^{d} |x_{ij}|^p.$$

Note that (5) implies the variance of $\mathbf{Y}_{\boldsymbol{j}}$ for $\boldsymbol{j} \sim \mathcal{D}$ is appropriately bounded. Furthermore, since $\mathcal{D}$ is an ℓ_p-sampling distribution, the ℓ_p-sampling sketches of [43] are useful for sampling $\boldsymbol{j} \sim \mathcal{D}$. Finally, the approach of [43] is particularly suited for bounding the errors incurred by Count-Min on $\widehat{\mathbf{sk}_1}$ and $\widehat{\mathbf{sk}_2}$, which we overview below.

At a high level, the ℓ_p-sampling sketch of [43] generates a sample $\boldsymbol{j}$ from $[d]$ by identifying a *heavy hitter* in a random scaling of the vector specifying the sampling probabilities. In particular, the algorithm generates $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_d \sim \text{Exp}(1)$ and identifies an entry $j \in [d]$ in the vector

$$\left(\frac{\|x_{\cdot,1}\|_p}{\boldsymbol{u}_1^{1/p}}, \frac{\|x_{\cdot,2}\|_p}{\boldsymbol{u}_2^{1/p}}, \ldots, \frac{\|x_{\cdot,d-1}\|_p}{\boldsymbol{u}_{d-1}^{1/p}}, \frac{\|x_{\cdot,d}\|_p}{\boldsymbol{u}_d^{1/p}} \right) \in \mathbb{R}^d,$$

whose value satisfies

$$\frac{\|x_{\cdot,j}\|_p}{\boldsymbol{u}_j^{1/p}} \gtrsim \left(\sum_{j'=1}^{d} \frac{\|x_{\cdot,j'}\|_p^p}{\boldsymbol{u}_{j'}} \right)^{1/p}, \tag{6}$$

and is the largest among those heavy hitters. For the coordinate $\boldsymbol{j} \in [d]$ recovered by the ℓ_p-sampling sketch [43], the inequality (6) gives a lower bound on how large $1/\boldsymbol{u}_{\boldsymbol{j}}^{1/p}$ will be. In particular, by applying the same transformation to the vector of *sketches*,

$$\left(\frac{\mathbf{sk}(x_{\cdot,1})}{\boldsymbol{u}_1^{1/p}}, \ldots, \frac{\mathbf{sk}(x_{\cdot,d})}{\boldsymbol{u}_d^{1/p}} \right) \in (\mathbb{R}^t)^d \qquad \text{and}$$
$$\left(\frac{\mathbf{sk}'(\mathbf{Pr}_{\mathcal{D}}[1])}{\boldsymbol{u}_1^{1/p}}, \ldots, \frac{\mathbf{sk}'(\mathbf{Pr}_{\mathcal{D}}[d])}{\boldsymbol{u}_d^{1/p}} \right) \in (\mathbb{R}^{t'})^d, \tag{7}$$

the t and t' coordinates corresponding to the sketches $\mathbf{sk}(x_{\cdot,\boldsymbol{j}}) \in \mathbb{R}^t$ and $\mathbf{sk}'(\mathbf{Pr}_{\mathcal{D}}[\boldsymbol{j}]) \in \mathbb{R}^{t'}$ will be heavy hitters of those vectors as well. Namely, with only $\text{poly}(\log(nd), 1/\epsilon)$-bits, the Count-Min data structure will recover the entries of $\mathbf{sk}(x_{\cdot,\boldsymbol{j}})$ and $\mathbf{sk}'(\mathbf{Pr}_{\mathcal{D}}[\boldsymbol{j}])$ up to a small additive error, proportional to the ℓ_1-norm of (7). We know the distribution of sketched vectors (7) (since these are simply ℓ_p-sketches [41]), so we will be able to bound the additive error and show that the sketch evaluation algorithms of $\widehat{\mathbf{sk}_1}$ and $\widehat{\mathbf{sk}_2}$ return the desired $(1 \pm \epsilon)$-approximations.

▶ Remark 4 (Comparison to [25]). The technique, "ℓ_p-sampling with meta-data", arises in [25] in the following context. They seek a linear sketch $\mathbf{sk}\colon \mathbb{R}^d \to \mathbb{R}^t$ which can process a vector $y \in \mathbb{R}^d$ and evaluate a weighted ℓ_1-norm, $\sum_{i=1}^d w_i(y) \cdot |y_i|$, where the weights $w_1(y), \ldots, w_d(y) \in \mathbb{R}_{\geq 0}$ are themselves dependent on the vector y. This arises as an algorithmic step in streaming algorithms for geometric minimum spanning tree and the earth-mover's distance. Mapping the above formulation to our setting, we want to evaluate a weighted ℓ_1-norm as well, where the i-th weight corresponds to $\mathbf{Pr}_{\boldsymbol{j} \sim \mathcal{D}}[\boldsymbol{j} = i]$, and the i-th value seek to sum is $\mathbf{Y}_i$ (as in (4)). The perspective of this technique as importance sampling (as presented in this work) is new. Indeed, the appropriate setting of weights is only apparent once one multiplies and divides the contribution of the j-th coordinate by $\|x_{\cdot,j}\|_p^p$ to define $\mathcal{D}$.

2 Sketching Median Costs

2.1 Statement of Main Lemma

▶ **Theorem 5.** *Fix $n, d \in \mathbb{N}$, as well as $p \in [1, 2]$ and $\epsilon, \delta \in (0, 1)$. There exists a linear sketch using $\text{poly}(\log d, 1/\epsilon, \log(1/\delta))$ space which processes a set of n points $x_1, \ldots, x_n \in \mathbb{R}^d$, and outputs a parameter $\boldsymbol{\eta} \in \mathbb{R}$ which satisfies*

$$\min_{y \in \mathbb{R}^d} \sum_{i=1}^{n} \|y - x_i\|_p^p \leq \boldsymbol{\eta} \leq (1+\epsilon) \min_{y \in \mathbb{R}^d} \sum_{i=1}^{n} \|y - x_i\|_p^p$$

with probability at least $1 - \delta$.

We work with the following representation of a linear sketch. The processed set of n points in $\mathbb{R}^d$ are stacked to form a vector $x \in \mathbb{R}^{nd}$. A linear sketch using space s is a distribution $\mathcal{M}$ supported on $s \times (nd)$ matrices. The theorem states that for any fixed $x_1, \ldots, x_n \in \mathbb{R}^d$, with probability $1 - \delta$ over the draw of $\mathbf{S} \sim \mathcal{M}$, an algorithm with access to the vector $\mathbf{S}x \in \mathbb{R}^s$ and $\mathbf{S}$ can output $\boldsymbol{\eta}$ satisfying the above guarantees.

Linear sketches of the above form imply efficient streaming algorithms, albeit with some subtleties. It is useful to first assume that the streaming algorithm can store its randomness for free (we will address this in the full version) so that it knows the matrix $\mathbf{S}$. In particular, since $\mathbf{S} \in \mathbb{R}^{s \times nd}$ acts on the vector $x \in \mathbb{R}^{nd}$ which vertically stacks $x_1, \ldots, x_n \in \mathbb{R}^d$, the columns of $\mathbf{S}$ may be broken up into n groups of size d, so

$$\mathbf{S} = \begin{bmatrix} \mathbf{S}_1 & \mathbf{S}_2 & \ldots & \mathbf{S}_n \end{bmatrix}, \qquad \text{and} \qquad \mathbf{S}x = \sum_{i=1}^{n} \mathbf{S}_i x_i.$$

In the *insertion-only* model, an algorithm would process the points one-at-a-time, and at time-step j, maintain $\sum_{i=1}^{j} \mathbf{S}_i x_i \in \mathbb{R}^s$. In the *turnstile* model of streaming, there is a subtlety in the implementation; namely, as the algorithm receives insertions and deletions of points in $\mathbb{R}^d$, it must know which index $i \in [n]$ it is considering. The reason is that the algorithm should know which of the sub-matrix $\mathbf{S}_i$ to update the point with.

For our application of the ℓ_p^p-median sketch to (k, p)-clustering in ℓ_p, we consider a weighted ℓ_p^p-median. Namely, for points $x_1, \ldots, x_n \in \mathbb{R}^d$ and weights $\lambda_1, \ldots, \lambda_n \in [0, 1]$ with $\sum_{i=1}^{n} \lambda_i = 1$, the ℓ_p^p-median cost with respect to weights $\lambda_1, \ldots, \lambda_n$ is

$$\min_{y \in \mathbb{R}^d} \sum_{i=1}^{n} \lambda_i \|y - x_i\|_p^p.$$

It is useful to first consider of $\lambda_1 = \cdots = \lambda_n = 1/n$. For general weights, the sketch will receive as input $\mathbf{S} = [\mathbf{S}_1, \ldots, \mathbf{S}_n] \in \mathbb{R}^{s \times (nd)}$, the vector $\sum_{i=1}^{n} \lambda_i^{1/p} \mathbf{S}_i x_i \in \mathbb{R}^s$, and the weights $\lambda_1, \ldots, \lambda_n$.

Centering Points

There is a straight-forward way to process the points so as to assume they are centered. Specifically, the average point may be subtracted from every point by applying a linear map, and since our sketch is linear, subtracting the average point may be incorporated into the sketch. For weights $\lambda_1, \ldots, \lambda_n \in [0, 1]$ satisfying $\sum_{i=1}^{n} \lambda_i = 1$, we consider the linear map

$$(x_1, \ldots, x_n) \mapsto \left(x_1 - \sum_{i=1}^{n} \lambda_i x_i, \ldots, x_n - \sum_{i=1}^{n} \lambda_i x_i \right) \in \mathbb{R}^{nd}.$$

Hence, we assume, without loss of generality, that the points $x_1, \ldots, x_n \in \mathbb{R}^d$ satisfy

$$\sum_{i=1}^{n} \lambda_i x_i = 0 \in \mathbb{R}^d. \tag{8}$$

The centering is useful for deriving the following set of inequalities, which will be useful for our sketching procedures. Suppose we denote $y \in \mathbb{R}^d$ as the point which minimizes $\sum_{i=1}^{n} \lambda_i \|y - x_i\|_p^p$. Then, for every $j \in [d]$,

$$\sum_{i=1}^{n} \lambda_i |y_j - x_{ij}|^p \leq \sum_{i=1}^{n} \lambda_i |x_{ij}|^p \leq 2^p \sum_{i=1}^{n} \lambda_i |y_j - x_{ij}|^p.$$

Importantly for us, every $j \in [d]$ satisfies

$$2^{-p} \leq \frac{\min_{y_j \in \mathbb{R}} \sum_{i=1}^n \lambda_i |y_j - x_{ij}|^p}{\sum_{i=1}^n \lambda_i |x_{ij}|^p} \leq 1. \tag{9}$$

We let $\mathcal{D}$ be the distribution supported on $[d]$ given by setting, for each $j \in [d]$,

$$\mathop{\mathbf{Pr}}_{\boldsymbol{j} \sim \mathcal{D}}[\boldsymbol{j} = j] = \frac{1}{Z} \sum_{i=1}^n \lambda_i |x_{ij}|^p \qquad \text{where} \qquad Z = \sum_{j'=1}^d \sum_{i=1}^n \lambda_i |x_{ij'}|^p = \sum_{i=1}^n \lambda_i \|x_i\|_p^p.$$

Then, the quantity we want to estimate may be equivalently re-written as:

$$\sum_{j=1}^d \min_{y_j \in \mathbb{R}} \sum_{i=1}^n \lambda_i |y_j - x_{ij}|^p = Z \cdot \mathop{\mathbf{E}}_{\boldsymbol{j} \sim \mathcal{D}} \left[\frac{\min_{y_{\boldsymbol{j}} \in \mathbb{R}} \sum_{i=1}^n \lambda_i |y_{\boldsymbol{j}} - x_{i\boldsymbol{j}}|^p}{\sum_{i=1}^n \lambda_i |x_{i\boldsymbol{j}}|^p} \right], \tag{10}$$

where the value within the expectation is bounded between 2^{-p} and 1. Furthermore, the quantity Z will be sketched with an ℓ_p-sketch, and a sample $\boldsymbol{j} \sim \mathcal{D}$ will be drawn with an ℓ_p-sampling sketch. Hence, the plan is to produce $t = O(1/\epsilon^2)$ samples of $\boldsymbol{j}_1, \ldots, \boldsymbol{j}_t \sim \mathcal{D}$, and produce a sketch to evaluate the numerator inside the expectation, and the denominator inside the expectation. Taking an empirical average of the samples to estimate the expectation, and multiplying it by the estimate of Z will give the desired estimator.

▶ **Lemma 6** (Main Lemma). *For any $n, d \in \mathbb{N}$, $p \in [1, 2]$ and $\epsilon, \delta \in (0, 1)$, let $s = \mathrm{poly}(\log d, 1/\epsilon, 1/\delta)$.*[5] *There exists a distribution $\mathcal{S}$ over $s \times (nd)$ matrices, and an algorithm such that for any n vectors $x_1, \ldots, x_n \in \mathbb{R}^d$ and any $\lambda_1, \ldots, \lambda_n \in [0, 1]$ with $\sum_{i=1}^n \lambda_i = 1$ and $\sum_{i=1}^n \lambda_i x_i = 0$, the following occurs:*

- *We sample $\mathbf{S} = [\mathbf{S}_1, \ldots, \mathbf{S}_n] \sim \mathcal{S}$, and we give the algorithm as input $\mathbf{S}$, $\sum_{i=1}^n \mathbf{S}_i(\lambda_i^{1/p} x_i)$, and $\lambda_1, \ldots, \lambda_n$.*
- *The algorithm outputs a tuple of three numbers $(\boldsymbol{j}, \boldsymbol{\alpha}, \boldsymbol{\beta}) \in [d] \times \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0}$. With probability at least $1 - \delta$ over the draw of $\mathbf{S} \sim \mathcal{S}$, we have the following two inequalities:*

$$(1-\epsilon) \left(\sum_{i=1}^n \lambda_i |x_{i\boldsymbol{j}}|^p \right)^{1/p} \leq \boldsymbol{\alpha} \leq (1+\epsilon) \left(\sum_{i=1}^n \lambda_i |x_{i\boldsymbol{j}}|^p \right)^{1/p},$$

$$(1-\epsilon) \min_{z \in \mathbb{R}} \left(\sum_{i=1}^n \lambda_i |x_{i\boldsymbol{j}} - z|^p \right)^{1/p} \leq \boldsymbol{\beta} \leq (1+\epsilon) \min_{z \in \mathbb{R}} \left(\sum_{i=1}^n \lambda_i |x_{i\boldsymbol{j}} - z|^p \right)^{1/p}.$$

- *Furthermore, the distribution of the random variable $\boldsymbol{j}$ is $\epsilon 2^{-p}$-close in total variation distance to $\mathcal{D}$.*

Proof of Theorem 5 assuming Lemma 6. Given Lemma 6, the proof of Theorem 5 is straight-forward. We fix $\lambda_1 = \cdots = \lambda_n = 1/n$, and we first handle the centering. We will utilize Lemma 6 which requires vectors $x_1, \ldots, x_n$ to satisfy $\sum_{i=1}^n \lambda_i x_i = 0$; hence, we sketch the vectors $x'_1, \ldots, x'_n$ given by $x'_i = x_i - \sum_{h=1}^n \lambda_h x_h$, which are now centered. By linearity, this is equivalent to maintaining the vector

$$\sum_{i=1}^n \lambda_i^{1/p} \mathbf{S}_i (x_i - \sum_{h=1}^n \lambda_h x_h) = \sum_{i=1}^n \left(\lambda_i^{1/p} \mathbf{S}_i - \lambda_i \sum_{h=1}^n \lambda_h^{1/p} \mathbf{S}_h \right) x_i \in \mathbb{R}^s.$$

[5] See the full version for the specific polynomial bounds.

We take $t = \omega(1/\epsilon^2)$ independent sketches from Lemma 6 with accuracy parameter $\epsilon/2$ and error probability $\delta = o(1/t)$. This, in turn, gives us t independent samples $(\boldsymbol{j}_1, \boldsymbol{\alpha}_1, \boldsymbol{\beta}_1)$, $\ldots$, $(\boldsymbol{j}_t, \boldsymbol{\alpha}_1, \boldsymbol{\beta}_t)$. By taking a union bound over the t executions of Lemma 6, with high probability, every $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_1, \ldots, \boldsymbol{\beta}_t$ satisfy

$$\boldsymbol{\alpha}_\ell^p \approx_{(1+\epsilon p/2)} \sum_{i=1}^n \lambda_i |x_{i\boldsymbol{k}}|^p \qquad \text{and} \qquad \boldsymbol{\beta}_\ell^p \approx_{(1+\epsilon p/2)} \min_{z\in\mathbb{R}} \sum_{i=1}^n \lambda_i |x_{i\boldsymbol{k}} - z|^p,$$

and $\boldsymbol{j}_1, \ldots, \boldsymbol{j}_t$ are independent draws from a distribution $\mathcal{D}'$ which is $\epsilon 2^{-p}$-close to $\mathcal{D}$. For estimating Z, we use an ℓ_p-sketch to accuracy $\epsilon/2$ and failure probability $\delta = o(1)$. For example, the sketch for Z may proceed by applying an ℓ_p-sketch [41] to the stacked vector $x' \in \mathbb{R}^{nd}$ where

$$x'_{ij} = \lambda_i^{1/p} \cdot x_{ij},$$

so that the ℓ_p norm of x' is exactly $Z^{1/p}$. Let $\widehat{\mathbf{Z}}$ be the estimate for the Z. For our estimate $\boldsymbol{\eta}$ that we will output, we set

$$\boldsymbol{\eta} = \widehat{\mathbf{Z}} \cdot \frac{1}{t} \sum_{\ell=1}^t \operatorname{minmax}\left\{2^{-p}, \left(\frac{\boldsymbol{\beta}_\ell}{\boldsymbol{\alpha}_\ell}\right)^p, 1\right\},$$

where $\operatorname{minmax}(l, x, u)$ is l if $x \le l$, u if $u \ge x$, and x otherwise. To see why our estimator approximates (10), we have $\widehat{\mathbf{Z}}$ is a $(1 \pm \epsilon/2)$-approximation of Z. The latter quantity is the empirical average of t i.i.d random variables, each of which is bounded by 2^{-p} and 1. In particular, we have that with probability at least $1 - o(1)$, Chebyshev's inequality, and the conditions of $\boldsymbol{\beta}_\ell$ and $\boldsymbol{\alpha}_\ell$,

$$\mathop{\mathbf{E}}_{\boldsymbol{j}\sim\mathcal{D}'}\left[\frac{\min_{z\in\mathbb{R}} \sum_{i=1}^n \lambda_i |x_{i\boldsymbol{j}} - z|^p}{\sum_{i=1}^n \lambda_i |x_{i\boldsymbol{j}}|^p}\right] \approx_{(1+2\epsilon p)} \frac{1}{t}\sum_{\ell=1}^t \operatorname{minmax}\left\{2^{-p}, \left(\frac{\boldsymbol{\beta}_\ell}{\boldsymbol{\alpha}_\ell}\right)^p, 1\right\}.$$

It remains to show that

$$\mathop{\mathbf{E}}_{\boldsymbol{j}\sim\mathcal{D}'}\left[\frac{\min_{z\in\mathbb{R}} \sum_{i=1}^n \lambda_i |x_{i\boldsymbol{j}} - z|^p}{\sum_{i=1}^n \lambda_i |x_{i\boldsymbol{j}}|^p}\right] \approx_{(1\pm\epsilon)} \mathop{\mathbf{E}}_{\boldsymbol{j}\sim\mathcal{D}}\left[\frac{\min_{z\in\mathbb{R}} \sum_{i=1}^n \lambda_i |x_{i\boldsymbol{j}} - z|^p}{\sum_{i=1}^n \lambda_i |x_{i\boldsymbol{j}}|^p}\right].$$

This follows from two facts: (1) $\mathcal{D}'$ and $\mathcal{D}$ are $\epsilon 2^{-p}$ close, since the random variable is at most 1, the expectations are off by at most an additive $\epsilon 2^{-p}$-factor, and (2) both quantities above are the average of random variables which are at least 2^{-p}, so an additive $\epsilon 2^{-p}$ error is less than a multiplicative $(1 \pm \epsilon)$-error.

The above gives an estimate which is a $1 \pm \epsilon$-approximation with probability $1 - o(1)$, in order to boost the probability of success to $1 - \delta$, we simply repeat $O(\log(1/\delta))$ times and output the median estimate. ◀

The remainder of the section is organized as follows. We give in the (next) Subsection 2.2, the necessary sketches for obtaining $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ for a fixed coordinate j. Then, in the following Subsection 2.3, we show how we combine various sketches from Subsection 2.2 for different $j \in [d]$ to obtain $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ up to some additive error. Finally, the proof of Lemma 6 appears in the full version, where we apply a randomized transformation to the input so that the additive error from Subsection 2.3 is a multiplicative error for the specific sampled $\boldsymbol{j}$.

2.2 Sketch for Optimizing a Single Coordinate

In this subsection, we give linear sketches which are useful for optimizing over a single coordinate. Specifically, given the n vectors $x_1, \ldots, x_n \in \mathbb{R}^d$ and $j \in [d]$, we consider the k-th coordinate of the n vectors $x_{1j}, x_{2j}, \ldots, x_{nj} \in \mathbb{R}$. Hence, the linear sketches in this section will act on vectors in $\mathbb{R}^n$, corresponding to the j-th coordinates of the points, and will give approximations to

$$\sum_{i=1}^{n} \lambda_j |x_{ij}|^p \quad \text{(Corollary 8)} \quad \text{and} \quad \min_{y_j \in \mathbb{R}} \sum_{i=1}^{n} \lambda_j |y_j - x_{ij}|^p. \quad \text{(Lemma 9)}$$

The lemma statements also consider an additive error term, $\text{err} \in \mathbb{R}_{\geq 0}$, which will be necessary when combining these sketches in Subsection 2.3; however, it may be helpful to consider $\text{err} = 0$ on first reading.

▶ **Lemma 7.** *For any $n \in \mathbb{N}$, $p \in [1,2]$ and $\epsilon, \delta \in (0,1)$, let $s = O(\log(1/\delta)/\epsilon^2)$. There exists a distribution $\mathcal{M}$ over $s \times n$ matrices, and an algorithm such that for any $x \in \mathbb{R}^n$, $\lambda_1, \ldots, \lambda_n \in [0,1]$, and $y \in \mathbb{R}$, the following occurs:*

- *We sample $\mathbf{S} \sim \mathcal{M}$ as well as a random vector $\boldsymbol{\chi} = (\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_s) \in \mathbb{R}^s$ where each is an i.i.d p-stable random variable. For any $\text{err} \in \mathbb{R}_{\geq 0}$, we give the algorithm as input $\mathbf{S}$, $\mathbf{S}(\lambda^{1/p} \circ x) + \text{err} \cdot \boldsymbol{\chi}$, the parameters $\lambda_1, \ldots, \lambda_n$, and y.*[6]
- *The algorithm outputs a parameter $\widehat{\boldsymbol{\eta}} \in \mathbb{R}_{\geq 0}$, which depends on $\mathbf{S}, \mathbf{S}(\lambda^{1/p} \circ x) + \text{err} \cdot \boldsymbol{\chi}$, the parameters $\lambda_1, \ldots, \lambda_n$, and y which satisfies with probability at least $1 - \delta$ over $\mathbf{S}$ and $\boldsymbol{\chi}$,*

$$(1-\epsilon)\left(\sum_{i=1}^{n} \lambda_i |x_i - y|^p + \text{err}^p\right)^{1/p} \leq \widehat{\boldsymbol{\eta}} \leq (1+\epsilon)\left(\sum_{i=1}^{n} \lambda_i |x_i - y|^p + \text{err}^p\right)^{1/p}. \quad (11)$$

Furthermore, for every $j \in [s]$, the random variables $(\mathbf{S}(\lambda^{1/p} \circ x))_j \in \mathbb{R}$ are distributed as $\|\lambda^{1/p} \circ x\|_p \cdot \boldsymbol{\chi}_j$, where $\boldsymbol{\chi}_j$ are independent, p-stable random variables.

Proof. We notice that this simply corresponds to an ℓ_p-sketch of the vector $z \in \mathbb{R}^n$, which is given by letting each $z_i = \lambda^{1/p} \circ (x_i - y)$, so that the ℓ_p-sketch of [41] would accomplish this task. Since the algorithm receives $\mathbf{S}$, $\mathbf{S}(\lambda^{1/p} \circ x) + \text{err} \cdot \chi$ and $y \in \mathbb{R}$, the algorithm may compute $\mathbf{S}(\lambda^{1/p} \circ y \cdot \mathbf{1})$, where $\mathbf{1} \in \mathbb{R}^n$ is an all-1's vector, and evaluate the sketch

$$\mathbf{S}(\lambda^{1/p} \circ x) + \text{err} \cdot \boldsymbol{\chi} - \mathbf{S}(\lambda^{1/p} \circ y \cdot \mathbf{1}) = \mathbf{S}(\lambda^{1/p} \circ (x - y \cdot \mathbf{1})) + \text{err} \cdot \boldsymbol{\chi}$$

by linearity. Furthermore, note that the error simply corresponds to an ℓ_p-sketch of the vector $z' \in \mathbb{R}^{n+1}$ which sets $z'_i = z_i$ for $i \neq n+1$ and $z'_{n+1} = \text{err}$. ◀

▶ **Corollary 8.** *For any $n \in \mathbb{N}$, $p \in [1,2]$ and $\epsilon, \delta \in (0,1)$, let $s = O(\log(1/\delta)/\epsilon^2)$. There exists a distribution $\mathcal{M}$ over $s \times n$ matrices, and an algorithm such that for any $x \in \mathbb{R}^n$, and any $\lambda_1, \ldots, \lambda_n \in [0,1]$, the following occurs:*

- *We sample $\mathbf{S} \sim \mathcal{M}$ and a random vector $\boldsymbol{\chi} = (\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_s) \in \mathbb{R}^s$ of i.i.d p-stable random variables. For any $\text{err} \in \mathbb{R}_{\geq 0}$. We give the algorithm as input $\mathbf{S}$ and $\mathbf{S}(\lambda^{1/p} \circ x) + \text{err} \cdot \boldsymbol{\chi}$.*

[6] The notation $\lambda^{1/p} \circ x \in \mathbb{R}^n$ denotes the Hadamard product, where $(\lambda^{1/p} \circ x)_i = \lambda_i^{1/p} \cdot x_i$.

- *With probability at least* $1-\delta$ *over* $\mathbf{S}$ *and* $\boldsymbol{\chi}$*, the algorithm outputs a parameter* $\widehat{\boldsymbol{\gamma}} \in \mathbb{R}_{\geq 0}$*, which depends on* $\mathbf{S}$*, and* $\mathbf{S}(\lambda^{1/p} \circ x)$ *which satisfies*

$$(1-\epsilon)\left(\sum_{i=1}^{n} \lambda_i |x_i|^p + \mathrm{err}^p\right)^{1/p} \leq \widehat{\boldsymbol{\gamma}} \leq (1+\epsilon)\left(\sum_{i=1}^{n} \lambda_i |x_i|^p + \mathrm{err}^p\right)^{1/p}.$$

Furthermore, for every $j \in [s]$*, the random variables* $(\mathbf{S}(\lambda^{1/p} \circ x))_j \in \mathbb{R}$ *are independent and distributed as* $\|\lambda^{1/p} \circ x\|_p \cdot \boldsymbol{\chi}_j$*, where* $\boldsymbol{\chi}_j$ *are independent, p-stable random variables.*

Proof. We apply Lemma 7 to the vector $x \in \mathbb{R}^n$ with $y = 0$. ◀

▶ **Lemma 9.** *For any* $n \in \mathbb{N}$*,* $p \in [1,2]$ *and* $\epsilon, \delta \in (0,1)$*, let* $s = O(\log(1/(\epsilon\delta))/\epsilon^2)$*. There exists a distribution* $\mathcal{M}$ *over* $s \times n$ *matrices, and an algorithm such that for any* $x \in \mathbb{R}^n$ *and any* $\lambda_1, \ldots, \lambda_n \in [0,1]$ *with* $\sum_{i=1}^n \lambda_i = 1$*, whenever* $\sum_{i=1}^n \lambda_i x_i = 0$*, the following occurs:*

- *We sample* $\mathbf{S} \sim \mathcal{M}$ *and a random vector* $\boldsymbol{\chi} = (\boldsymbol{\chi}_1, \ldots, \boldsymbol{\chi}_s)$ *of i.i.d p-stable random variables. For any* $\mathrm{err} \in \mathbb{R}_{\geq 0}$*. We give the algorithm as input* $\mathbf{S}$*,* $\mathbf{S}(\lambda^{1/p} \circ x) + \mathrm{err} \cdot \boldsymbol{\chi}$*, the parameters* $\lambda_1, \ldots, \lambda_n$*, and a parameter* $\gamma \in \mathbb{R}_{\geq 0}$ *satisfying*

$$(1-\epsilon)\left(\sum_{i=1}^{n} \lambda_i |x_i|^p\right)^{1/p} \leq \gamma \leq (1+\epsilon)\left(\sum_{i=1}^{n} \lambda_i |x_i|^p\right)^{1/p}.$$

- *The algorithm outputs a parameter* $\widehat{\boldsymbol{\beta}} \in \mathbb{R}_{\geq 0}$ *which satisfies*

$$(1-\epsilon)\min_{z\in\mathbb{R}}\left(\sum_{i=1}^{n} \lambda_i |x_i - z|^p + \mathrm{err}^p\right)^{1/p} \leq \widehat{\boldsymbol{\beta}} \leq (1+\epsilon)\min_{z\in\mathbb{R}}\left(\sum_{i=1}^{n} \lambda_i |x_i - z|^p + \mathrm{err}^p\right)^{1/p}. \tag{12}$$

Furthermore, for every $j \in [s]$*, the random variables* $(\mathbf{S}(\lambda^{1/p} \circ x))_j$ *are independent and distributed as* $\|\lambda^{1/p} \circ x\|_p \cdot \boldsymbol{\chi}_j$*, where* $\boldsymbol{\chi}_j$ *are independent, p-stable random variables.*

Proof. We will utilize the sketch from Lemma 7, while varying the y's to find the minimum. Specifically, let $t = 16 \cdot 2^p/\epsilon$, and let the distribution $\mathcal{M}$ be the same as that of Lemma 7 instantiated with error probability $1 - t\delta$ and accuracy parameter $\epsilon/2$. We discretize the interval $[-4\gamma, 4\gamma]$ into t, evenly-spaced out points $y_1, \ldots, y_t \subset [-4\gamma, 4\gamma]$ such that $y_{\ell+1} - y_\ell = 8\gamma/t$. We utilize the algorithm in Lemma 7 to obtain estimates $\widehat{\boldsymbol{\eta}}_1, \ldots, \widehat{\boldsymbol{\eta}}_t$ satisfying (11) with $y_1, \ldots, y_t$, respectively. Then, we output

$$\widehat{\boldsymbol{\beta}} = \min_{\ell\in[t]} \widehat{\boldsymbol{\eta}}_\ell.$$

Since we amplified the error probability to less than $t\delta$, we may assume, by a union bound, that all estimates $\{\boldsymbol{\eta}_\ell\}_{\ell\in[t]}$ satisfy (11) with y_ℓ with probability at least $1-\delta$. First, for any $\ell \in [t]$,

$$\min_{z\in\mathbb{R}}\left(\sum_{i=1}^{n} \lambda_i |x_i - z|^p\right)^{1/p} \leq \left(\sum_{i=1}^{n} \lambda_i |x_i - y_\ell|^p\right)^{1/p},$$

and therefore, the lower bound in (12) is implied by (11). To prove the upper bound in (12), denote $z \in \mathbb{R}$ as the true minimizer of $(\sum_{i=1}^n \lambda_i |x_i - z|^p)^{1/p}$. By the fact $\sum_{i=1}^n \lambda_i = 1$ and the triangle inequality, we have

$$|z| = \left(\sum_{i=1}^n \lambda_i |z|^p\right)^{1/p} \leq \left(\sum_{i=1}^n \lambda_i |x_i - z|^p\right)^{1/p} + \left(\sum_{i=1}^n \lambda_i |x_i|^p\right)^{1/p} \leq 2\left(\sum_{i=1}^n \lambda_i |x_i|^p\right)^{1/p},$$

so $|z| \leq 2(1+\epsilon)\gamma \leq 4\gamma$, and thus $z \in [-4\gamma, 4\gamma]$. Let $\ell \in [t]$ be such that $|y_t - z| \leq 4\gamma/t$. Then, again by the triangle inequality and the fact $\sum_{i=1}^n \lambda_i x_i = 0$,

$$\begin{aligned}\left(\sum_{i=1}^n \lambda_i |x_i - y_t|^p\right)^{1/p} &\leq \left(\sum_{i=1}^n \lambda_i |x_i - z|^p\right)^{1/p} + 4\gamma/t \\ &\leq (1 + 4(1+\epsilon)2^p/t)\left(\sum_{i=1}^n \lambda_i |x_i - z|^p\right)^{1/p}.\end{aligned}$$

By the setting of t, $(\sum_{i=1}^n \lambda_i |x_i - y_t|^p)^{1/p} \leq (1+\epsilon/2)(\sum_{i=1}^n \lambda_i |x_i - z|^p)^{1/p}$, and by (11), we obtain the desired upper bound. ◀

2.3 Grouping Single Coordinate Sketches

In this subsection, we show how to compress d linear sketches (one for each coordinate) from Subsection 2.2. In the lemma that follows, the parameter $m \in \mathbb{N}$ should be considered the sketch size of the sketches in Subsection 2.2, and the linear sketch will take the d sketches from Subsection 2.2 (represented as a vector $\mathbb{R}^{dm}$). Each of the d linear sketches have each coordinate of $\mathbb{R}^m$ distributed as an i.i.d scaled p-stable random variable (specified by the last sentence in Corollary 8 and Lemma 9). Thus, we write the d sketches as $\boldsymbol{\Psi}_1 v_1, \ldots, \boldsymbol{\Psi}_d v_d \in \mathbb{R}^m$, where $v_j \in \mathbb{R}$ is a scaling, and $\boldsymbol{\Psi}_1, \ldots, \boldsymbol{\Psi}_d \in \mathbb{R}^{m \times n}$ are i.i.d p-stable matrices.

▶ **Lemma 10** (p-stable Sketch Compression via Count-Min). *Let $d, m \in \mathbb{N}$, $\epsilon, \delta \in (0,1)$, and let $t = O(\log(d/\delta))$. There exists a distribution $\mathcal{C}$ over $(10tm/\epsilon^p) \times (dm)$ matrices, and an algorithm such that for any $v \in \mathbb{R}^d$, the following occurs:*

- *We sample $\mathbf{C} \sim \mathcal{C}$ and a $(dm) \times d$ matrix $\boldsymbol{\Psi}$, where $\boldsymbol{\Psi} = \mathrm{diag}(\boldsymbol{\Psi}_1, \ldots, \boldsymbol{\Psi}_d)$, and each $\boldsymbol{\Psi}_j = (\boldsymbol{\chi}_{j1}, \ldots, \boldsymbol{\chi}_{jm}) \in \mathbb{R}^m$ are independent p-stable random vectors.[7] The algorithm receives as input $\mathbf{C}$ and $\mathbf{C}\boldsymbol{\Psi}v$.*
- *The algorithm outputs, for each $j \in [d]$, a sequence of t vectors $\widehat{\boldsymbol{z}}_j^{(1)}, \ldots, \widehat{\boldsymbol{z}}_j^{(t)} \in \mathbb{R}^m$ which satisfy, for each $t' \in [t]$,*

$$\widehat{\boldsymbol{z}}_j^{(t')} = v_j \boldsymbol{\Psi}_j + \mathbf{err}_j^{(t')} \cdot \boldsymbol{\chi}_j^{(t')}. \tag{13}$$

where $\boldsymbol{\chi}_j^{(t')} \in \mathbb{R}^m$ is a vector of independent p-stable random variables, and $\mathbf{err}_j^{(t')} \in \mathbb{R}_{\geq 0}$ only depends on $\mathbf{C}$. With probability at least $1 - \delta$ over $\mathbf{C}$, for every $j \in [d]$

$$\left|\left\{t' \in [t] : \mathbf{err}_j^{(t')} \leq \epsilon \|v\|_p\right\}\right| \geq t/2. \tag{14}$$

[7] Hence, the vector $\boldsymbol{\Psi}v \in \mathbb{R}^{dm}$ is given by vertically stacking d vectors of the form $v_j \boldsymbol{\Psi}_j \in \mathbb{R}^m$.

Proof. The matrix $\mathbf{C}$ is a Count-Min matrix which given a vector $u \in \mathbb{R}^{dm}$ given by vertically stacking d vectors $u_1, \ldots, u_d \in \mathbb{R}^m$ repeats the following process: for each $t' \in [t]$, we sample a hash function $\boldsymbol{h}_{t'} : [d] \to [10/\epsilon^p]$, and for each $\ell \in [10/\epsilon^p]$ store the vector

$$\boldsymbol{b}_{t',\ell} \stackrel{\text{def}}{=} \sum_{j \in [d]} \mathbf{1}\{\boldsymbol{h}_{t'}(j) = \ell\} \cdot u_j \in \mathbb{R}^m.$$

In particular, the output $\mathbf{C}u$ consists of stacking $t \cdot 10/\epsilon^p$ vectors $(\boldsymbol{b}_{t',\ell} \in \mathbb{R}^m : t' \in [t], \ell \in [10/\epsilon^p])$, which gives the desired bound of $10mt/\epsilon^p$ on the output dimension of $\mathbf{C}$. For each $j \in [d]$ and $t' \in [t]$, the algorithm lets $\boldsymbol{\ell} = \boldsymbol{h}_{t'}(j)$ and sets

$$\widehat{\boldsymbol{z}}_j^{(t')} = \boldsymbol{b}_{t',\ell} = v_j \boldsymbol{\Psi}_j + \sum_{j' \in [d] \setminus \{j\}} \mathbf{1}\{\boldsymbol{h}_{t'}(j') = \boldsymbol{\ell}\} \cdot v_{j'} \cdot \boldsymbol{\Psi}_{j'}.$$

We now apply the p-stability property to the right-most summand, to notice that

$$\mathbf{err}_j^{(t')} = \left(\sum_{j' \in [d] \setminus \{j\}} \mathbf{1}\{\boldsymbol{h}_{t'}(j') = \boldsymbol{\ell}\} \cdot v_{j'}^p \right)^{1/p},$$

which only depends on $\mathbf{C}$. Furthermore, the inner most summand is at most $\epsilon^p/10 \sum_{j' \in [d] \setminus \{j\}} v_{j'}^p$ in expectation. By Markov's inequality, each $\mathbf{err}_j^{(t')} \leq \epsilon \|v\|_p$ with probability at least $9/10$. Since $t = O(\log(d/\delta))$, the probability that (14) is not satisfied for each $j \in [d]$ is at most δ/d by a Chernoff bound, so that a union bound gives the desired guarantees. ◀

The above lemma allows us to compress d many p-stable sketches into $O(\log(d/\delta)/\epsilon^p)$ many p-stable sketches, albeit with some error. Since the p-stable sketches that we will use (from Corollary 8 and Lemma 9) are exactly of the form $\boldsymbol{\Psi}v$ for some vector v, Lemma 10 will allow us to compress them. Namely, we will consider d sketches from Corollary 8 and Lemma 9 and utilize Lemma 10; for each $j \in [d]$, we will be able to recover t noisy versions of the sketch of Corollary 8 and Lemma 9 for coordinate j. Importantly, the noise is of the form an error times a p-stable random variable, and these are the kinds of errors that Corollary 8 and Lemma 9 can easily handle.

▶ **Lemma 11** (p-stable Sketch Recovery for Sample). *For $n, d \in \mathbb{N}$, $p \in [1, 2]$ and $\epsilon, \delta \in (0, 1)$, let $s = O(\log^2(d/\delta)/\epsilon^{2+p})$. There exists a distribution $\mathcal{R}$ over $s \times (nd)$ matrices, and an algorithm such that for any vectors $y_1, \ldots, y_n \in \mathbb{R}^d$ and weights $\lambda_1, \ldots, \lambda_n \in [0, 1]$ with $\sum_{i=1}^n \lambda_i = 1$, the following occurs with probability at least $1 - \delta$:*

- *We sample $\mathbf{S} = [\mathbf{S}_1, \ldots, \mathbf{S}_n] \sim \mathcal{R}$ and we give the algorithm as input $\mathbf{S}$, and the vector $\sum_{i=1}^n \mathbf{S}_i(\lambda_i^{1/p} y_i) \in \mathbb{R}^s$.*
- *The algorithm outputs d numbers $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_d \in \mathbb{R}_{\geq 0}$ such that each $j \in [d]$ satisfies*

$$(1-\epsilon)\left(\sum_{i=1}^n \lambda_i |y_{ij}|^p\right)^{1/p} - \text{err} \leq \boldsymbol{\alpha}_j \leq (1+\epsilon)\left(\sum_{i=1}^n \lambda_i |y_{ij}|^p\right)^{1/p} + \text{err},$$

where $\text{err} \in \mathbb{R}_{\geq 0}$ is an additive error satisfying

$$\text{err} \leq \epsilon \left(\sum_{j=1}^d \sum_{i=1}^n \lambda_i |y_{ij}|^p\right)^{1/p}.$$

Proof. We combine Corollary 8 and Lemma 10. We describe the distribution $\mathcal{R}$ over $s \times (nd)$ matrices by giving a procedure for sampling $\mathbf{S} \sim \mathcal{R}$. $\mathbf{S}$ can be naturally expressed as a concatenation of matrices $\mathbf{S} = [\mathbf{S}_1, \ldots, \mathbf{S}_n]$.

- We let $\mathcal{M}$ be the distribution over $s_0 \times n$ matrices of Corollary 8 with error probability at most $\delta/(2dt)$ and accuracy ϵ (so that we may union bound over d sketches later) so that $s_0 = O(\log(dt/\delta)/\epsilon^2)$. We take d independent samples $\mathbf{S}'_1, \ldots, \mathbf{S}'_d \sim \mathcal{M}$.
- For each $j \in [d]$, we let P_j be the $n \times (nd)$ matrix where given the vector $y' \in \mathbb{R}^{nd}$ given by vertically stacking $\lambda_1^{1/p} y_1, \ldots, \lambda_n^{1/p} y_n \in \mathbb{R}^d$, sets $y'_{\cdot,j} = P_k j'$, where $y'_{\cdot,j} = \lambda^{1/p} \circ (y_{i,j})_{i \in [n]} \in \mathbb{R}^n$. Let P be the $(nd) \times (nd)$ matrix which stacks these matrices vertically.
- We sample $\mathbf{C} \sim \mathcal{C}$ as in Lemma 10 with $m = s_0$, where we set the accuracy parameter $\epsilon/2$ and the failure probability $\delta/2$. We let

$$\mathbf{S} = \mathbf{C} \cdot \mathrm{diag}(\mathbf{S}'_1, \ldots, \mathbf{S}'_d) \cdot P.$$

Intuitively, we will apply our sketch $\mathbf{S}$ on the vector the matrix $\mathbf{S}$ may be interpreted as first applying d sketches of Corollary 8 to the vectors $(\lambda^{1/p} \circ y_{\cdot,1}), \ldots, (\lambda^{1/p} \circ y_{\cdot,d}) \in \mathbb{R}^n$, and then applying $\mathbf{C}$ from Lemma 10. The algorithm for producing the estimates $\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_d$ proceeds by applying the algorithm of Lemma 10 to obtain, for each $j \in [d]$ a sequence of t vectors $\widehat{\boldsymbol{z}}_j^{(1)}, \ldots, \widehat{\boldsymbol{z}}_j^{(t)} \in \mathbb{R}^{s_0}$. We apply the algorithm of Corollary 8 to each of the t vectors to obtain estimates $\boldsymbol{\alpha}_j^{(1)}, \ldots, \boldsymbol{\alpha}_j^{(t)} \in \mathbb{R}_{\geq 0}$, and we let $\boldsymbol{\alpha}_j = \mathrm{median}\{\boldsymbol{\alpha}_j^{(t')} : t' \in [t]\}$.

To see why this works, consider the collection of d vectors

$$\boldsymbol{z}_j = \mathbf{S}'_j(\lambda^{1/p} \circ y_{\cdot,j}) \in \mathbb{R}^{s_0},$$

and notice that by Corollary 8, every $j \in [d]$ and $\ell \in [s]$, $\boldsymbol{z}_{j,\ell} \sim (\sum_{i=1}^n \lambda_i |y_{i,j}|^p)^{1/p} \cdot \boldsymbol{\chi}_{j,\ell}$, where $\boldsymbol{\chi}_{k,\ell}$ are independent, p-stable random variables. Indeed, if we write $v \in \mathbb{R}^d$ as the vector which sets

$$v_j = \left(\sum_{i=1}^n \lambda_i |y_{i,j}|^p \right)^{1/p},$$

then vertically stacking the vectors $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_d \in \mathbb{R}^{s_0}$ gives a vector which is equivalently distributed as $\boldsymbol{\Psi} v$, where $\boldsymbol{\Psi}$ is the matrix from Lemma 10. In particular, with probability at least $1 - \delta/2$, the algorithm of Lemma 10 outputs dt vectors $(\widehat{\boldsymbol{z}}_j^{(t')} : j \in [d], t' \in [t])$ which satisfy

$$\widehat{\boldsymbol{z}}_j^{(t')} = \mathbf{S}'_k(\lambda^{1/p} \circ y_{\cdot,j}) + \mathbf{err}_j^{(t')} \cdot \boldsymbol{\chi}_{j,t'}. \tag{15}$$

Hence, with probability at least $1 - \delta/(2dt)$, the algorithm of Corollary 8 applied to $\widehat{\boldsymbol{z}}_j^{(t')}$ outputs an estimate $\boldsymbol{\alpha}_j^{(t')}$ satisfying $(1-\epsilon)(v_j^p + (\mathbf{err}_j^{(t')})^p)^{1/p} \leq \boldsymbol{\alpha}_j^{(t')} \leq (1+\epsilon)(v_j^p + (\mathbf{err}_j^{(t')})^p)^{1/p}$, and therefore, we have that each $\boldsymbol{\alpha}_j^{(t')}$ satisfies

$$(1-\epsilon)v_j - \mathbf{err}_j^{(t')} \leq \boldsymbol{\alpha}_j^{(t')} \leq (1+\epsilon)v_j + 2 \cdot \mathbf{err}_j^{(t')}.$$

Since at least $t/2$ of $t' \in [t]$ satisfies $\mathbf{err}_j^{(t')} \leq \epsilon/2 \cdot \|v\|_p$, the median $\boldsymbol{\alpha}_j^{(t')}$ satisfies the desired error guarantee. Applying a union bound over all dt applications of Corollary 8 and Lemma 10 gives the desired guarantees. ◀

▶ **Lemma 12** (p-stable Sketch Recovery for Optimizer)**.** *For $n, d \in \mathbb{N}$, $p \in [1,2]$ and $\epsilon, \delta \in (0,1)$, let $s = O(\log(d/\delta) \cdot \log(\log d/(\epsilon\delta))/\epsilon^{2+p})$. There exists a distribution $\mathcal{O}$ over $s \times (nd)$ matrices, and an algorithm such that for any vectors $y_1, \ldots, y_n \in \mathbb{R}^d$ and any set of weights $\lambda_1, \ldots, \lambda_n \in [0,1]$ where $\sum_{i=1}^n \lambda_i = 0$, whenever $\sum_{i=1}^n \lambda_i y_i = 0$, the following occurs with probability at least $1 - \delta$:*

- *We sample $\mathbf{S} = [\mathbf{S}_1, \ldots, \mathbf{S}_n] \sim \mathcal{O}$ and we give the algorithm as input $\mathbf{S}$, $\sum_{i=1}^n \mathbf{S}_i(\lambda_i^{1/p} y_i)$, the parameters $\lambda_1, \ldots, \lambda_n$, an index $j_0 \in [d]$, and a parameter $\gamma \in \mathbb{R}_{\geq 0}$ satisfying*

$$(1-\epsilon)\left(\sum_{i=1}^n \lambda_i |y_{ij_0}|^p\right)^{1/p} \leq \gamma \leq (1+\epsilon)\left(\sum_{i=1}^n \lambda_i |y_{ij_0}|^p\right)^{1/p}.$$

- *The algorithm outputs a parameter $\widehat{\boldsymbol{\beta}} \in \mathbb{R}_{\geq 0}$ which satisfies*

$$(1-\epsilon)\min_{z\in\mathbb{R}}\left(\sum_{i=1}^n \lambda_i |y_{ij_0} - z|^p\right)^{1/p} - \mathrm{err} \leq \widehat{\boldsymbol{\beta}} \leq (1+\epsilon)\min_{z\in\mathbb{R}}\left(\sum_{i=1}^n \lambda_i |y_{ij_0} - z|^p\right)^{1/p} + \mathrm{err},$$

where $\mathrm{err} \in \mathbb{R}_{\geq 0}$ is an additive error satisfying

$$\mathrm{err} \leq \epsilon\left(\sum_{j=1}^d \sum_{i=1}^n \lambda_i |y_{i,j}|^p\right)^{1/p}.$$

Proof. The proof follows similarly to that of Lemma 11; the only difference is that instead of using the sketch of Corollary 8, we use the sketch of Lemma 9. For completeness, we describe the distribution $\mathcal{O}$ over $s \times (nd)$ matrices by giving a procedure for sampling $\mathbf{S} \sim \mathcal{O}$:

- We let $\mathcal{M}$ be the distribution over $s_0 \times n$ matrices from Lemma 9 with accuracy ϵ and failure probability $\delta/(2t)$, where $s_0 = O(\log(t/(\epsilon\delta))/\epsilon^2)$. We take d independent samples $\mathbf{S}'_1, \ldots, \mathbf{S}'_d \sim \mathcal{M}$. Note that even though we take d independent samples, we will only require that the sketch t evaluations of the $\mathbf{S}_{j_0} \sim \mathcal{M}$ succeed (hence, we amplify the error probability to $\delta/(2t)$, as opposed to $\delta/(2td)$ as in Lemma 11).
- We sample $\mathbf{C} \sim \mathcal{C}$ as in Lemma 10 with $m = s_0$, where we set the accuracy parameter $\epsilon/2$ and failure probability $\delta/2$. Recalling the definition of P (see Item 2 in the proof of Lemma 11, we let

$$\mathbf{S} = \mathbf{C} \cdot \mathrm{diag}(\mathbf{S}'_1, \ldots, \mathbf{S}'_d) \cdot P.$$

Similarly to the proof of Lemma 11, $\mathbf{S}$ may be interpreted as applying the sketch of Lemma 10 to d vectors in $\mathbb{R}^{s_0}$, each $j \in [d]$ of which is an independent sketch $\mathbf{S}'_j(\lambda^{1/p} \circ y_{\cdot,j}) \in \mathbb{R}^{s_0}$, where $\mathbf{S}'_j \sim \mathcal{M}$ is the sketch of Lemma 9. Again, we consider the collection of d vectors $\boldsymbol{z}_j = \mathbf{S}'_j(\lambda^{1/p} \circ y_{\cdot,j}) \in \mathbb{R}^{s_0}$, for all $j \in [d]$, and by Lemma 9, every $j \in [d]$ has $\boldsymbol{z}_j \sim \|\lambda^{1/p} \circ y_{\cdot,j}\|_p \cdot \boldsymbol{\Psi}_j \in \mathbb{R}^{s_0}$, where $\boldsymbol{\Psi}_j$ is an independent, p-stable random vector. Writing $v \in \mathbb{R}^d$ by $v_j = \|\lambda^{1/p} \circ y_{\cdot,j}\|_p$, and we apply the algorithm of Lemma 10 and focus on the t vectors $\widehat{\boldsymbol{z}}_{j_0}^{(1)}, \ldots, \widehat{\boldsymbol{z}}_{j_0}^{(t)} \in \mathbb{R}^{s_0}$ which satisfy

$$\widehat{\boldsymbol{z}}_{j_0}^{(t')} = \mathbf{S}'_{j_0}(\lambda^{1/p} \circ y_{\cdot,j_0}) + \mathbf{err}_{j_0}^{(t')} \cdot \boldsymbol{\chi}_{j,t'}.$$

We apply the algorithm of Lemma 9 to each of the vectors $\widehat{\boldsymbol{z}}_{j_0} \in \mathbb{R}^s$, while giving as input the parameter γ to obtain the estimate $\widehat{\boldsymbol{\beta}}^{(1)}, \ldots, \widehat{\boldsymbol{\beta}}^{(t)}$. Then, we set $\widehat{\boldsymbol{\beta}} = \mathrm{median}\{\widehat{\boldsymbol{\beta}}^{(t')} : t' \in [t]\}$. Similarly to the proof of Lemma 11, $\widehat{\boldsymbol{\beta}}$ provides the desired approximation guarantees. ◀

References

1 Pankaj K. Agarwal, Sariel Har-Peled, and Kasturi R. Varadarajan. Geometric approximation via coresets. *Combinatorial and computational geometry*, 2005.

2 Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999.

3 Alexandr Andoni, Khanh Do Ba, Piotr Indyk, and David Woodruff. Efficient sketches for earth-mover distance, with applications. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2009)*, 2009.

4 Alexandr Andoni, Moses Charikar, Ofer Neiman, and Huy L. Nguyen. Near linear lower bound for dimension reduction in l1. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS '2011)*, 2011.

5 Alexandr Andoni, Piotr Indyk, and Robert Krauthgamer. Earth mover distance over high-dimensional spaces. In *Proceedings of the 19th ACM-SIAM Symposium on Discrete Algorithms (SODA '2008)*, pages 343–352, 2008.

6 Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Polylogarithmic approximation for edit distance and the asymmetric query complexity. In *Proceedings of the 51st Annual IEEE Symposium on Foundations of Computer Science (FOCS '2010)*, 2010.

7 Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Streaming algorithms from precision sampling. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS '2011)*, 2011.

8 Alexandr Andoni, Robert Krauthgamer, and Ilya Razenshteyn. Sketching and embedding are equivalent for norms. In *Proceedings of the 47th ACM Symposium on the Theory of Computing (STOC '2015)*, pages 479–488, 2015. Available as `arXiv:1411.2577`.

9 Pranjal Awasthi, Moses Charikar, Ravishankar Krishnaswamy, and Ali Kemal Sinop. The hardness of approximation of euclidean k-means. In *31st International Symposium on Computational Geometry (SoCG 2015)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2015.

10 Arturs Backurs, Piotr Indyk, Ilya Razenshteyn, and David P. Woodruff. Nearly-optimal bounds for sparse recovery in generic norms, with applications to k-median sketching. In *Proceedings of the 27th ACM-SIAM Symposium on Discrete Algorithms (SODA '2016)*, pages 318–337, 2016. Available as `arXiv:1504.01076`.

11 Mihai Badoiu, Sariel Har-Peled, and Piotr Indyk. Approximate clustering via core-sets. In *Proceedings of the 34th ACM Symposium on the Theory of Computing (STOC '2002)*, 2002.

12 Ziv Bar-Yossef, T.S. Jayram, Ravi Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68(4):702–732, 2004.

13 Arturs Bačkurs and Piotr Indyk. Better embeddings for planar earth-mover distance over sparse sets. In *Proceedings of the 41st International Colloquium on Automata, Languages and Programming (ICALP '2014)*, 2014.

14 Luca Becchetti, Marc Bury, Vincent Cohen-Addad, Fabrizio Grandoni, and Chris Schwiegelshohn. Oblivious dimension reduction for k-means: Beyond subspaces and the johnson-lindenstrauss lemma. In *Proceedings of the 51th ACM Symposium on the Theory of Computing (STOC '2019)*, 2019.

15 Soheil Behnezhad. Time-optimal sublinear algorithms for matching and vertex cover. In *Proceedings of the 62nd Annual IEEE Symposium on Foundations of Computer Science (FOCS '2021)*, 2021.

16 Jon Louis Bentley and James B Saxe. Decomposable searching problems i. static-to-dynamic transformation. *Journal of Algorithms*, 1(4):301–358, 1980.

17 Guy Blanc, Neha Gupta, Jane Lange, and Li-Yang Tan. Estimating decision tree learnability with polylogarithmic sample complexity. In *Proceedings of Advances in Neural Information Processing Systems 33 (NeurIPS '2020)*, 2020.

18 Jaroslaw Blasiok, Vladimir Braverman, Stephen R. Chestnut, and Robert Krauthgamerand Lin F. Yang. Streaming symmetric norms via measure concentration. In *Proceedings of the 50th ACM Symposium on the Theory of Computing (STOC '2017)*, 2017.

19 Christos Boutsidis, Anastasios Zouzias, and Petros Drineas. Random projections for k-means clustering. In *Proceedings of Advances in Neural Information Processing Systems 23 (NeurIPS '2010)*, 2010.

20 Vladimir Braverman, Dan Feldman, and Harry Lang. New frameworks for offline and streaming coreset constructions. *arXiv preprint*, 2016. `arXiv:1612.00889`.

21 Vladimir Braverman, Gereon Frahling, Harry Lang, Christian Sohler, and Lin F Yang. Clustering high dimensional dynamic data streams. In *Proceedings of the 34th International Conference on Machine Learning (ICML '2017)*, 2017.

22 Bo Brinkman and Moses Charikar. On the impossibility of dimension reduction in ℓ_1. *Journal of the ACM*, 52(5):766–788, 2005.

23 Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Theoretical Computer Science*, 312(1):3–15, 2004.

24 Ke Chen. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM Journal on Computing*, 39(3):923–947, 2009.

25 Xi Chen, Rajesh Jayaram, Amit Levi, and Erik Waingarten. New streaming algorithms for high dimensional emd and mst. In *Proceedings of the 54th ACM Symposium on the Theory of Computing (STOC '2022)*, 2022.

26 Yu Chen, Sampath Kannan, and Sanjeev Khanna. Sublinear algorithms and lower bounds for metric tsp cost estimation. In *47th International Colloquium on Automata, Languages, and Programming (ICALP 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

27 Michael B. Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Mădălina Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *Proceedings of the 47th ACM Symposium on the Theory of Computing (STOC '2015)*, 2015.

28 Vincent Cohen-Addad and CS Karthik. Inapproximability of clustering in lp metrics. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 519–539. IEEE, 2019.

29 Vincent Cohen-Addad, David Saulpic, and Chris Schwiegelshohn. A new coreset framework for clustering. In *Proceedings of the 53rd ACM Symposium on the Theory of Computing (STOC '2021)*, 2021.

30 Graham Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

31 Artur Czumaj, Funda Engün, Lance Fortnow, Avner Magen, Ilan Newman, Ronitt Rubinfeld, and Christian Sohler. Approximating the weight of the euclidean minimum spanning tree in sublinear time. *SIAM Journal on Computing*, 2005.

32 Artur Czumaj, Shaofeng H.-C. Jiang, Robert Krauthgamer, Pavel Veselý, and Mingwei Yang. Streaming facility location in high dimension via new geometric hashing. *arXiv preprint*, 2022. `arXiv:2204.02095`.

33 Artur Czumaj and Chirstian Sohler. Estimating the weight of metric minimum spanning trees in sublinear time. *SIAM Journal on Computing*, 2009.

34 Dan Feldman and Michael Langberg. A unified framework for approximating and clustering data. In *Proceedings of the 43rd ACM Symposium on the Theory of Computing (STOC '2011)*, 2011.

35 Dan Feldman, Melanie Schmidt, and Christian Sohler. Turning big data into tiny data: constant-size coresets for k-means, pca and projective clustering. In *Proceedings of the 24th ACM-SIAM Symposium on Discrete Algorithms (SODA '2013)*, 2013.

36 Sariel Har-Peled and Soham Mazumdar. On coresets for k-means and k-median clustering. In *Proceedings of the 36th ACM Symposium on the Theory of Computing (STOC '2004)*, 2004.

37 Monika Henzinger and Sagar Kale. Fully-dynamic coresets. In *28th Annual European Symposium on Algorithms (ESA 2020)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020.

38 Wei Hu, Zhao Song, Lin F. Yang, and Peilin Zhong. Nearly optimal dynamic k-means clustering for high-dimensional data. *arXiv preprint*, 2019. `arXiv:1802.00459`.

39 Lingxiao Huang and Nisheeth K. Vishnoi. Coresets for clustering in euclidean spaces: importance sampling is nearly optimal. In *Proceedings of the 52nd ACM Symposium on the Theory of Computing (STOC '2020)*, 2020.

40 Piotr Indyk. Algorithms for dynamic geometric problems over data streams. In *Proceedings of the 36th ACM Symposium on the Theory of Computing (STOC '2004)*, 2004.

41 Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *Journal of the ACM*, 53(3):307–323, 2006.

42 Piotr Indyk and David Woodruff. Optimal approximations of the frequency moments of data streams. In *Proceedings of the 37th ACM Symposium on the Theory of Computing (STOC '2005)*, pages 202–208, 2005.

43 Rajesh Jayaram and David Woodruff. Perfect lp sampling in a data stream. In *Proceedings of the 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2018)*, 2018.

44 Thathachar S. Jayram and David Woodruff. The data stream complexity of cascaded norms. In *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2009)*, 2009.

45 William Johnson and Joram Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. In *Conference in modern analysis and probability (New Haven, Conn., 1982)*, volume 26 of *Contemporary Mathematics*, pages 189–206. AMS, 1984.

46 Daniel M. Kane, Jelani Nelson, and David P. Woodruff. On the exact space complexity of sketching and streaming small norms. In *Proceedings of the 21st ACM-SIAM Symposium on Discrete Algorithms (SODA '2010)*, 2010.

47 Michael Kapralov, Slobodan Mitrović, Ashkan Norouzi-Fard, and Jakab Tardos. Space efficient approximation to maximum matching size from uniform edge samples. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1753–1772. SIAM, 2020.

48 Weihao Kong, Emma Brunskill, and Gregory Valiant. Sublinear optimal policy value estimation in contextual bandits. In *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS '2020)*, 2020.

49 Weihao Kong and Gregory Valiant. Estimating learnability in the sublinear data regime. In *Proceedings of Advances in Neural Information Processing Systems 31 (NeurIPS '2018)*, pages 5455–5464, 2018.

50 Amit Kumar, Yogish Sabharwal, and Sandeep Sen. A simple linear time $(1+\epsilon)$-approximation algorithm for k-means clustering in any dimension. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2004)*, 2004.

51 Eyal Kushilevitz, Rafail Ostrovsky, and Yuval Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM Journal on Computing*, 30(2):457–474, 2000.

52 Michael Langberg and Leonard J. Schulman. Universal epsilon-approximators for integrals. In *Proceedings of the 21st ACM-SIAM Symposium on Discrete Algorithms (SODA '2010)*, 2010.

53 Euiwoong Lee, Melanie Schmidt, and John Wright. Improved and simplified inapproximability for k-means. *Information Processing Letters*, 120:40–43, 2017.

54 James R. Lee and Assaf Naor. Embedding the diamond graph in L_p and dimension reduction in L_1. *Geometric and Functional Analysis*, 14(4):745–747, 2004.

55 Konstantin Makarychev, Yuri Makarychev, and Ilya Razenshteyn. Performance of johnson-lindenstrauss transform for k-means and k-medians clustering. In *Proceedings of the 51th ACM Symposium on the Theory of Computing (STOC '2019)*, 2019.

56 Huy N Nguyen and Krzysztof Onak. Constant-time approximation algorithms via local improvements. In *2008 49th Annual IEEE Symposium on Foundations of Computer Science*, pages 327–336. IEEE, 2008.

57 Krzysztof Onak, Dana Ron, Michal Rosen, and Ronitt Rubinfeld. A near-optimal sublinear-time algorithm for approximating the minimum vertex cover size. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 1123–1131. SIAM, 2012.

58 Art B. Owen. Monte carlo theory, methods, and examples, 2013.

59 Michal Parnas and Dana Ron. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. *Theoretical Computer Science*, 381(1-3):183–196, 2007.

60 Michael Saks and Xiaodong Sun. Space lower bounds for distance approximation in the data stream model. In *Proceedings of the 34th ACM Symposium on the Theory of Computing (STOC '2002)*, 2002.

61 Christian Sohler and David Woodruff. Strong coresets for k-median and subspace approximation. In *Proceedings of the 59th Annual IEEE Symposium on Foundations of Computer Science (FOCS '2018)*, 2018.

62 Yuichi Yoshida, Masaki Yamamoto, and Hiro Ito. An improved constant-time approximation algorithm for maximum matchings. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 225–234, 2009.