Near-optimal (euclidean) metric compression

Near-Optimal (Euclidean) Metric Compression

Piotr Indyk* MIT

Tal Wagner† MIT

Abstract

The metric sketching problem is defined as follows. Given a metric on n points, and $\epsilon > 0$, we wish to produce a small size data structure (sketch) that, given any pair of point indices, recovers the distance between the points up to a $1 + \epsilon$ distortion. In this paper we consider metrics induced by ℓ_2 and ℓ_1 norms whose spread (the ratio of the diameter to the closest pair distance) is bounded by $\Phi > 0$. A well-known dimensionality reduction theorem due to Johnson and Lindenstrauss yields a sketch of size $O(\epsilon^{-2} \log(\Phi n) n \log n)$, i.e., $O(\epsilon^{-2} \log(\Phi n) \log n)$ bits per point. We show that this bound is not optimal, and can be substantially improved to $O(\epsilon^{-2} \log(1/\epsilon) \cdot \log n + \log\log \Phi)$ bits per point. Furthermore, we show that our bound is tight up to a factor of $\log(1/\epsilon)$.

We also consider sketching of general metrics and provide a sketch of size $O(n \log(1/\epsilon) + \log\log \Phi)$ bits per point, which we show is optimal.

*indyk@mit.edu.

†talw@mit.edu.

1 Introduction

Compact representations (or sketches) of high-dimensional data are very useful tools for applications involving data storage, processing and analysis. A prototypical example of this approach is the Johnson-Lindenstrauss theorem [JL84], which states that any set of n points in the Euclidean space of arbitrary dimension can be mapped into a space of dimension $O(\epsilon^{-2} \log n)$ such that the distances between any pair of points are preserved up to a factor of $1 + \epsilon$. The theorem makes it possible to represent each point as a sequence of $O(\epsilon^{-2} \log n)$ *numbers*. The number of *bits* needed to represent a point depends on the precision of the underlying data set. If the coordinates of the high-dimensional points come from the range $\{-\Phi \dots \Phi\}$, and if one uses the "binary" variant of the Johnson-Lindenstrauss theorem due to [Ach03], then each coordinate of the projected points can be represented using $O(\log(n\Phi))$ bits. This yields $O(\epsilon^{-2} \log(n) \log(n\Phi))$ bits per point.

Perhaps surprisingly, it is not known whether this bound is tight for metric sketching, or whether the sketch length can be improved much further. This could be in part because the Johnson-Lindenstrauss theorem itself was not known to be optimal until very recently [LN16] (although it was known to be optimal up to a factor of $\log(1/\epsilon)$ [Alo03]). Still, even an optimal lower bound for dimensionality reduction does not imply a corresponding bound on the sketch length. Another set of related results are the lower bounds from [JW13, MWY13], which show that $\Omega(\epsilon^{-2} n \log(n/\delta) \log \Phi)$ bits are needed to sketch the ℓ_1 or ℓ_2 distances between two sets of n points with probability $1 - \delta$. The latter lower bounds match the "discretized Johnson-Lindenstrauss" upper bound outlined above. However, they apply only to the *one-way communication* variant of the problem, where each point is held by either Alice or Bob, who communicate in order to estimate the distances across the partition. In contrast, in our setting, a single party holds and compresses the *whole* data set (see Section 2 for the formal definition). Thus the lower bounds of [JW13, MWY13] are also inapplicable in our context.

Our results. In this paper we show that the "discretized Johnson-Lindenstrauss" sketching bound is *not* tight. In particular, we describe a new randomized sketching algorithm for n points that enables estimating the distances up to a factor of $1 + \epsilon$ using at most $O(\epsilon^{-2} \log(1/\epsilon) \log n + \log \log \Phi)$ bits per point. This substantially improves over the earlier bound, replacing the $\log(n\Phi)$ term by $\log(1/\epsilon)$, and exponentially reducing the dependence on the precision parameter Φ. Furthermore, we show a lower bound of $O(\epsilon^{-2} \log n + \log \log \Phi)$, which implies that the bounds are tight up to a factor of $\log(1/\epsilon)$. This result, as well as all results that follow, extend to the setting where the coordinates of the points are arbitrary real numbers but the spread of the pointset (the ratio of its diameter to the closest pair distance) is at most Φ.

The result for the Euclidean metric is a corollary of a more general result about sketching of *arbitrary* ℓ_p norms. Specifically, we show that any metric induced by a set of n points in a d-dimensional space under an ℓ_p norm can be sketched using $O((d + \log n) \log(1/\epsilon) + \log \log \Phi)$ bits per point. The result for the Euclidean norm is then obtained by letting $d = O(\epsilon^{-2} \log n)$. Furthermore, since the ℓ_∞ norm is universal (any n-point metric space can be embedded into ℓ_∞^n), it also follows that any n-point metric can be sketched using at most $O(n \log(1/\epsilon) + \log \log \Phi)$ bits per point, which we show is tight. This improves over a naive bound of $O(n \log(\log(\Phi)/\epsilon))$ obtained by rounding each distance to the nearest integer power of $1 + \epsilon$ and storing the exponent, after an appropriate scaling.

Related Work. Distance-preserving sketches and data structures have been studied extensively. In what follows we focus on the prior work that is most relevant to the results in this paper.

As described earlier, it is known that one can approximate the distances between n points in the Euclidean space using $O(\epsilon^{-2}\log(n)\log(n\Phi))$ bits per point. If the goal is to only preserve distances in a certain range, say in the interval $[t, 10t]$ for some parameter t, then one can achieve a $O(\epsilon^{-2}\log(n))$ bits bound using distance sketches due to [KOR98]. However, in general, there are $O(\log\Phi)$ different "scales" t to preserve, so this approach does not improve over the aforementioned bound. Similarly, the very recent work [AK16] focused on approximating Euclidean distances between points of norm at most 1, up to an *additive* error of ϵ, and showed a sketch of size $O(\epsilon^{-2}\log n)$ bits per point.

Distance labeling is a general approach to the class of the problems considered in this paper. In particular, it is known [PS89, TZ05] that any metric over n nodes can be represented by roughly $n^{1+\Theta(1/c)}$ numbers while distorting the distances by a factor of c. Note that in order to achieve a near-linear sketch size, the distortion must be almost logarithmic.

Quadtrees are simple and popular data structures for storing point sets (see [Sam88]), often used in practice for low dimensional data, typically 2D or 3D. They are further related to our algorithm, as both are based on a hierarchical clustering of the points. For arbitrary low dimensional points, the quadtree size can still be as large as $n\log\Phi$. Several works [dBÁGB+13, VM14, GGNL+15] showed that if the points are "well clustered" then the quadtree can be compressed into $O(1)$ bits per point, implying a similar bound for storing the metric space. [Hud09] considers a different assumption on the point set, called well-seperatedness. They show that this assumption implies that the quadtree has at most $O(n)$ nodes, which makes it possible to store the metric with distortion $1+\epsilon$ using $O(\log(1/\epsilon) + (\log\Phi)/n)$ bits per point. Without structural assumptions on the point set, their techniques still require $\Omega(\log\Phi)$ bits per point.

Our techniques Our construction is based on a hierarchical clustering of the metric space, which naturally forms a tree T of clusters. The clustering at level ℓ is the partition to connected components induced by drawing edges between points at distance at most 2^ℓ. Each cluster is assigned a representative point called a *center*. Note that unlike other typical variants of hierarchical clustering (eg. quad-trees), the diameter of our clusters is unbounded in terms of ℓ.

The tree size is first reduced to linear by compressing long paths of nodes with only one child. From a distance estimation point of view, this means that if a cluster is very well separated from the rest of the metric (in terms of the ratio between its diameter to the distance to the closest external point), then we can replace it entirely with its center for the purpose of estimating the distances between internal and external points.

After the compression, our aim is to store every point as the displacement from a nearby point that is already stored. To this end, we keep track of the structure of the clusters beyond what is given in T. A cluster C at level ℓ is formed by drawing edges of length up to 2^ℓ between some clusters $C_1, \ldots, C_k$ in level $\ell - 1$. We fix a rooted spanning tree in this graph of clusters, and for every non-root cluster C_j, we store its center as the displacement from the closest point in C_i, its parent cluster in the spanning tree. We call that point the *ingress* of C_j. The distance between the ingress and the center is bounded by $2^\ell + \mathrm{diam}(C_j)$, and does not depend on $\mathrm{diam}(C_i)$. This will ensure we pay (in storage cost) for the diameter of each cluster only once.

The displacement is rounded to a precision that depends on the diameter and the level, roughly $2^{-\ell}\mathrm{diam}(C_j)$, and we show that the total precision cost over all the clusters is linear. By adding the rounded displacement back to the ingress, we obtain an approximation for the center of C_j, which we call a *surrogate*. Our estimate for the distance between two points would be the distance between their surrogates.

The above description is oversimplified, and our actual choice of ingresses and surrogates is

more careful. First, the closest point to C_j in C_i may have been lost in the preceding compression, so we might need to settle for another nearby point. Second, we need to store the displacement not from the ingress itself, but rather from the surrogate of the ingress, since the latter is what will actually be available to us during estimation. This means the surrogates are defined recursively, and we need to prevent error from accummulating. Ultimately we show that within given parts of the tree (called *subtrees*), we can recover the surrogates up to a fixed (unknown) shift, and this gives us satisfactory estimates for the original distances.

2 Preliminaries and Formal Statements

For an integer n, let X be a fixed set of n labeled points. Throughout we will use the convention $X = \{1, \ldots, n\}$. In the metric sketching problem, our goal is to design a sketch from which the distance between any pair of points can be approximately recovered, given their labels. Formally, let $\mathcal{D}_X$ be a family of metrics on X. For an integer $b > 0$, we define a *b-bit sketching scheme* for $\mathcal{D}_X$ as a pair of algorithms $(\mathsf{Summ}, \mathsf{Est})$ such that

- Summ is a (possibly randomized) summary algorithm, mapping a metric $D \in \mathcal{D}_X$ to a bitstring of length b.
- Est is an estimation algorithm, which takes the output of Summ on a metric $D \in \mathcal{D}_X$, and a pair of labels $x, y \in X$, and outputs an estimate for $D(x, y)$.

The sketching scheme has *distortion* k if for every $D \in \mathcal{D}_X$, if S_D is the output of a successful execution of Summ on D, then for every $x, y \in X$,

$$D(x, y) \leq \mathsf{Est}(S_D, x, y) \leq k \cdot D(x, y).$$

We denote,

$$b_k(\mathcal{D}_X) := \inf\{b : \mathcal{D}_X \text{ has a } b\text{-bit sketching scheme with distortion } k\}.$$

We are interested in upper bounds on $b_{1+\epsilon}(\mathcal{D}_X)$ for commonly arising metric families $\mathcal{D}_X$, and most importantly Euclidean metrics.

To state our results, we recall some basic definitions. The *spread* of a metric D on X is the ratio $\frac{\max_{x,y \in X} D(x,y)}{\min_{x,y \in X, x \neq y} D(x,y)}$. For $p \geq 1$, the *ℓ_p-norm* of a point $v \in \mathbb{R}^d$ is $\|v\|_p := \left(\sum_{i=1}^d |v_i|^p\right)^{1/p}$. The *$\ell_\infty$-norm* is $\|v\|_\infty := \max_i |v_i|$. For $1 \leq p \leq \infty$, a metric D on X is called a *d-dimensional ℓ_p-metric* if there is a map $f : X \to \mathbb{R}^d$ such that $D(x, y) = \|f(x) - f(y)\|_p$ for every $x, y \in X$.

Our main theorem is an upper bound for general ℓ_p metrics. Let $\mathcal{D}_p(n, d, \Phi)$ denote the family of all d-dimensional ℓ_p-metrics on X with spread at most Φ. (The dependence on X is omitted in order to keep the notation simple. Recall it is an arbitrary set of n labels.)

Theorem 2.1. *For every $1 \leq p \leq \infty$,*

$$b_{1+\epsilon}(\mathcal{D}_p(n, d, \Phi)) = O(n(d + \log n)\log(1/\epsilon) + n \log\log \Phi).$$

The summary and estimation algorithms are deterministic and run in time $\mathrm{poly}(n, d, \epsilon^{-1}, \log \Phi)$.

or Euclidean metrics ($p = 2$), we can first apply the Johnson-Lindenstrauss theorem on the pointset (distorting the distances by at most $(1 + \epsilon)$) and then apply Theorem 2.1.

Theorem 2.2 (Euclidean metrics)**.**

$$b_{1+\epsilon}(\mathcal{D}_2(n,d,\Phi)) = O(\epsilon^{-2}\log(1/\epsilon)\cdot n\log n + n\log\log\Phi).$$

Note that since the Johnson-Lindenstrauss theorem is randomized, then so is the resulting summary algorithm for Euclidean metrics. This means that with probility $1/\mathrm{poly}(n)$, it may output a sketch that distorts the distances by more than a $(1+\epsilon)$ factor. However, this does not effect the sketch size nor the running time.

We complement this upper bound by a matching lower bound up to the $\log(1/\epsilon)$ term, see Theorem 6.1.

Another notable special case is ℓ_1-metrics. By known embedding results, both our upper and lower bounds on $\mathcal{D}_2(n,d,\Phi)$ hold for $\mathcal{D}_1(n,d,\Phi)$ as well, see Appendix A for details.

For general metrics, we obtain tight upper and lower bounds. Let $\mathcal{D}_{\mathrm{all}}(n,\Phi)$ be the family of all metrics on X with spread at most Φ. Since any metric on n points is an n-dimensional ℓ_∞-metric, we obtain the following corollary from Theorem 2.1.

Theorem 2.3 (general metrics)**.**

$$b_{1+\epsilon}(\mathcal{D}_{\mathrm{all}}(n,\Phi)) = O(n^2\log(1/\epsilon) + n\log\log\Phi).$$

We show a tight lower bound in Theorem 6.2.

3 Summary Algorithm

In this section we begin the proof of Theorem 2.1, by describing and analyzing the summary algorithm. In Section 4 we describe and analyze the estimation algorithm, and in Section 5 we discuss the running times of both algorithms.

Let D be a d-dimensional ℓ_p-metric on X, and let $f: X\to\mathbb{R}^d$ denote the available embedding, meaning $D(x,y) = \|f(x)-f(y)\|_p$ for all $x,y\in X$. Throughout the proof we write $\|\cdot\|$ for the ℓ_p-norm $\|\cdot\|_p$, omitting the subscript. We assume by normalization that $\min_{x,y\in X, x\neq y}\|f(x)-f(y)\| = 1$, and thus Φ is an upper bound on the diameter.

We use the following variant of hierarchically well-separated trees (HSTs). [Bar96]

Definition 3.1. *Let T be a rooted, edge-weighted tree. Number the levels in T bottom-up, starting with the deepest leaf which is defined to belong to the level 0. Denote by $\ell(v)$ the level of every node v in T.*

We say T is a k-hierarchically well-separated tree (k-HST) *if for every node v in T, each edge connecting v to a child has weight $k^{\ell(v)}$.*

3.1 Building the Tree

We now construct a 2-HST T from X in a bottom-up manner. For $i = 0,1,\ldots,\log\mathrm{diam}(X)+1$,

- Let $G_i(X,E_i)$ be the (unweighted) graph in which $x,y\in X$ are neighbors if $\|f(x)-f(y)\| < 2^i$. (Note that $E_0=\emptyset$, by our assumption that the minimum pairwise distance is 1.)
- For every connected component C in G_i add a tree node v, and let $C(v) := C$.
- The connected components of G_i form a partition of X, and if $i>0$, the partition at level $i-1$ is a refinement of the partition at level i. Add the corresponding tree edges. This means that for all tree nodes v,u at levels i and $i-1$ respectively, such that $C(u)\subset C(v)$, we attach v as the parent of u.

Notation and terminology. For every $v \in T$, we denote its level in T by $\ell(v)$. The *degree* of v is its number of children. The set $C(v)$ is the *cluster* associated with v. We denote its diameter by

$$\Delta(v) := \mathrm{diam}(C(v)).$$

Observe that the leaves in T correspond bijectively to points in X, in the sense that for every $x \in X$ there is a unique leaf whose associated cluster is $\{x\}$. We denote that leaf by $\mathrm{leaf}(x)$.

Observation 3.2. *If $x, y \in X$ are at different components of the partition induced by the level-i nodes of T, then $\|f(x) - f(y)\| \geq 2^i$.*

3.2 Compressing the Tree

As constructed above, T has n leaves and up to $\log \Phi + 2$ levels, and since it may contain degree-1 nodes its total size can be as large as $O(n \log \Phi)$. We wish to make it smaller by compressing long paths of degree-1 nodes.

A *maximal 1-path* in T is a downward path $v_0, v_1, \ldots, v_k$ such that $v_1, \ldots, v_{k-1}$ are degree-1 nodes, and v_0 and v_k are not degree-1 nodes (v_k may have degree 0). For every such path in T, if $k > \log(\frac{\Delta(v_k)}{2^{\ell(v_k)}}) + \log(1/\epsilon)$, we replace the path from v_1 to v_k with a *long edge* directly connecting v_1 to v_k. (Note that the edge v_1 remains a degree-1 node in the tree.) We mark it as long and store with it the original path length, k. Non-long edges will be called *short edges*.

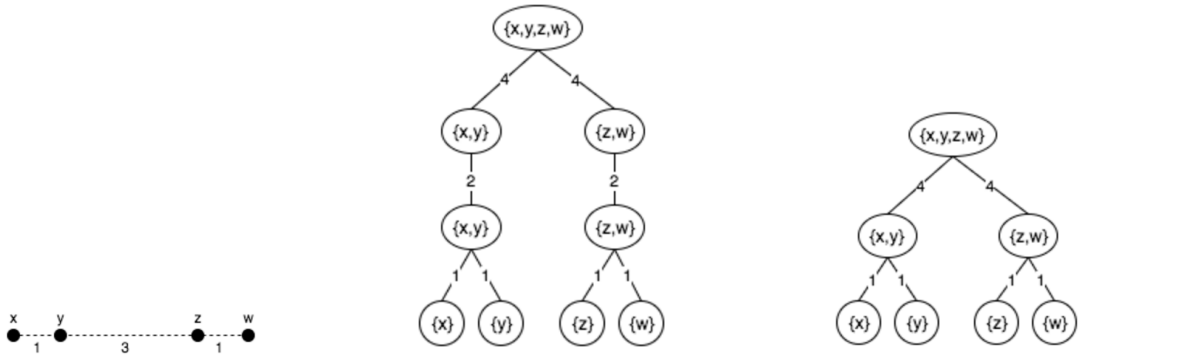
Lemma 3.3. *The tree after compression has at most $2n(2 + \log(1/\epsilon))$ nodes.*

Proof. We charge the degree-1 nodes on every 1-path to the bottom node of the path. The total number of nodes in the tree can then be written as $\sum_{v:\deg(v)\neq 1} k(v)$, where $k(v)$ is the length of the maximal 1-path whose bottom node is v. Due to the compression we have $k(v) \leq \log(\frac{\Delta(v_k)}{2^{\ell(v_k)}}) + \log(1/\epsilon)$. Since the tree has n leaves, it has at most $2n$ non-degree-1 nodes, so the total contribution of the second term is at most $2n \log(1/\epsilon)$. For the total contribution of the first term, we need to show

$$\sum_{v:\deg(v)\neq 1} \log\left(\frac{\Delta(v)}{2^{\ell(v)}}\right) \leq 4n. \tag{1}$$

To this end, consider the original tree T (before compression) and contract every edge whose top (parent) node has degree 1. Do this repeatedly, until there are no more degree-1 nodes, to obtain a tree T'. When contracting an edge $u \to v$, we identify the contracted node with the bottom node v of the original edge, and it keeps its $\Delta(v)$ and $\ell(v)$ that were set in T (note that $\ell(v)$ continues to denote the level of v in T, and not in T'). It is clearly sufficient to prove eq. (1) for T' instead of T, since we have only removed degree-1 nodes in the transition. Also note that T' has n leaves and no degree-1 nodes, and hence at most $2n$ nodes in total.

For every node $v \in T'$, $\Delta(v)$ is upper-bounded by the sum of the edge weights in the subtree of T' rooted at v, where the weight of an edge $u \to u'$ in T' is $2^{\ell(u)}$ (see Definition 3.1 and recall that $\ell(u)$ denotes the level of u in T). To see this, consider an edge $u \to v$. By construction of T, this means the cluster $C(v)$ has been merged into the larger cluster $C(u)$, when the distance from $C(v)$ to $C(u) \setminus C(v)$ was at most $2^{\ell(u)}$. Hence the edge weight, which is $2^{\ell(u)}$, bounds the contribution of that merging to the diameter of $C(u)$. However, if u is a degree-1 node, then $C(u) = C(v)$ and no merging has been performed, so there is no contribution to the diameter of $C(u)$ that needs to be accounted for. In sum, only those edges whose top nodes has degree different than 1 are needed in order to bound the cluster diameters, and these are exactly the edges in T'. See Figure 1 for illustration.



(a) The metric space (b) The tree T (c) The contracted tree T'

Figure 1: The metric space consists of 4 colinear points at distances as indicated in (a). In the contracted tree T', the diameter of every cluster is bounded by the sum of edge weights (written on the edges) in the corresponding subtree.

Consequently, it is sufficient to prove

$$\sum_{v \in T'} \log \left(\frac{\mathrm{wt}(v)}{2^{\ell(v)}} \right) \leq 4n,$$

where the *weight* $\mathrm{wt}(v)$ of a node v in T' is the sum of the edge weights in its subtree. We will prove the stronger bound,

$$\sum_{v \in T'} \frac{\mathrm{wt}(v)}{2^{\ell(v)}} \leq 4n, \tag{2}$$

by summing over edges. An edge in T' contributes 1 to the term $\frac{\mathrm{wt}(v)}{2^{\ell(v)}}$ of its parent v (recall that the edge weight is $2^{\ell(v)}$), 1/2 to the term $\frac{\mathrm{wt}(v')}{2^{\ell(v')}}$ of its grandparent v', 1/4 to the great-grandparent term, and so on. In total, each edge contributes at most 2 to the sum in eq. (2), and since T' has at most $2n$ edges, the sum is bounded by $4n$. □

From now on T will denote the tree after compression. We will often partition it into *subtrees* by removing the long edges.

3.3 Centers

With every node v in T we now associate a *center* $c(v) \in X$, which will be a representative point for the associated cluster $C(v)$. We choose the centers by the following bottom-up process on T:

- If v is a leaf in T, i.e. $v = \mathrm{leaf}(x)$ for some $x \in X$, then set $c(v) := x$.
- If v is the top node of a long edge, then recall it is the unique edge outgoing from v (since by construction top nodes of long edges have degree 1). Denote by u the bottom node of the long edge, and set $c(v) := c(u)$.
- Otherwise, v has children $v_1, \ldots, v_k$ connected to it by short edges. Recall that in the graph $G_{\ell(v)}$, $C(v)$ is a connected component that contains each of $C(v_1), \ldots, C(v_k)$. By contracting each of those clusters in $G_{\ell(v)}$ into a single node, we get a connected graph whose nodes correspond to $v_1, \ldots, v_k$. Fix an arbitrary rooted spanning tree of this graph, and denote it $\tau(v)$. Suppose w.l.o.g. that the root is v_1. Set $c(v) := c(v_1)$.

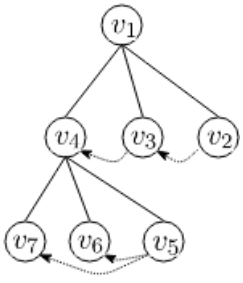


Figure 2: The solid arcs represent the tree T, and the dashed arrows represent the trees $\tau(v_1)$ and $\tau(v_4)$, defined on their children in T. The tree $\tau(v_1)$ is the path $v_2 \to v_3 \to v_4$. The tree $\tau(v_4)$ is the star with center v_5 pointing at v_6 and v_7. The *parent* of v_3 is v_1, while its τ-*predecessor* is v_2.

For every node v in T we have just fixed a rooted tree $\tau(v)$ on its children $v_1, \ldots, v_k$. We will use those trees later in the construction. To make the text clearer, we will refer to the parent of v_i in $\tau(v)$ as τ-*predecessor* of v_i (which is another child of v in T). In contrast, the term *parent* of v_i will be reserved for its parent in T (which is v). See Figure 2 for illustration.

3.4 Ingresses

With every node v in T we will now associate an *ingress* node, $in(v) \in T$. Intuitively, the idea is to store the location of $c(v)$ as its displacement from $c(in(v))$, the center of the ingress node. Therefore we would like the ingress to be a tree node whose center is close to $c(v)$, and such that we have available an approximate location for it.

The ingresses are chosen separately within each subtree, where we recall that the *subtrees* are formed from T by removing the long edges. For the root of the subtree we do not set an ingress as we will not need one. Now suppose we have a node v with children $v_1, \ldots, v_k$ in the same subtree (i.e. connected to v with short edges). Recall we have a tree $\tau(v)$ on $v_1, \ldots, v_k$, rooted at v_1, where an edge in τ connecting v_i, v_j means that

$$2^{\ell(v)-1} \leq \operatorname{dist}(C(v_i), C(v_j)) < 2^{\ell(v)}, \tag{3}$$

where

$$\operatorname{dist}(C(v_i), C(v_j)) := \min\{\|f(x) - f(y)\| : x \in C(v_i), y \in C(v_j)\}.$$

For v_1, we set $in(v_1) = v$. For v_i with $i > 1$, let v_j be the τ-predecessor of v_i. Let $y_i \in C(v_j)$ be the closest point to $C(v_i)$ in $C(v_j)$. Note that in T, there is a downward path from v_j to $\operatorname{leaf}(y_i)$. We set $in(v_i)$ to be the lowest point on that path that does not go through any long edge.

The motivation for this choice is that ideally we would like $\operatorname{leaf}(y_i)$ to be the ingress of v_i, but $\operatorname{leaf}(y_i)$ might be outside the current subtree. As we will see next, we will have approximate locations relative to $c(v_i)$ only for nodes in the same subtree as v_i. Therefore we choose the ingress as the node in the current subtree whose center is closest to y_i.

The following lemma gives us a bound on the distance between the node center and its ingress center.

Lemma 3.4. *For every $u \in T$ which is not a root of a subtree, we have*

$$\|f(c(u)) - f(c(in(u)))\| \leq 3 \cdot 2^{\ell(u)} + \Delta(u). \tag{4}$$

Proof. We use the same notation as in the above choice of ingresses. Suppose we have a node v with children $v_1, \ldots, v_k$, and we wish to prove the bound for some $u = v_i$. For v_1 we have $in(v_1) = v$ and (by choice of centers) $c(v_1) = c(v)$, hence $c(v_1) = c(in(v_1))$ and eq. (4) holds for $u = v_1$ trivially.

Now suppose $i > 1$. By eq. (3), the point y_i satisfies

$$\|f(c(v_i)) - f(y_i)\| \leq 2^{\ell(v)} + \Delta(v_i).$$

Noting that $\ell(v_i) = \ell(v) - 1$, we have

$$\|f(c(v_i)) - f(y_i)\| \leq 2 \cdot 2^{\ell(v_i)} + \Delta(v_i). \tag{5}$$

We have set $in(v_i)$ to be the lowest node in the path from v_j to $\text{leaf}(y_i)$ that does not traverse a long edge. We consider two cases:

- The path has no long edges, which means $in(v_i) = \text{leaf}(y_i)$. Then $c(in(v_i)) = y_i$, and eq. (4) for $u = v_i$ follows from eq. (5).
- The path has long edges, which means $in(v_i)$ is the top node of a long edge. Let k be its original length and w its bottom node. Note that $c(in(v_i)) = c(w)$. Then

$$\|f(y_i) - f(c(in(v_i)))\| = \|f(y_i) - f(c(w))\| \leq \Delta(w)$$

$$= 2^{\ell(w)+\log(\frac{\Delta(w)}{2^{\ell(w)}})} < 2^{\ell(w)+k} = 2^{\ell(in(v_i))} \leq 2^{\ell(v_i)}.$$

  Combining this with eq. (5) yields eq. (4) for $u = v_i$.

□

We also state the following fact.

Claim 3.5. *For a node with an ingress, $\ell(in(v)) \leq \ell(v) + 1$.*

Proof. By construction, $in(v)$ is either the parent of v in T, or a descendant of the parent. □

3.5 Surrogates

We now associate a *surrogate* $s^*(v) \in \mathbb{R}^d$ with each tree node v, which will be an approximate location for its center $c(v)$. The goal is to choose the surrogates such that the distances between them can be recovered from the sketch, thus approximating the distances between the actual points in X.

For $\delta > 0$ and $B \subset \mathbb{R}^d$, recall that $N \subset \mathbb{R}^d$ is a *δ-net* for B if for every $q \in B$ there is $\bar{q} \in N$ such that $\|q - \bar{q}\| \leq \delta$. We use the following known result.

Lemma 3.6. *For every $\delta > 0$ there is a δ-net $\mathcal{N}_\delta$ for the unit ball in ℓ_p^d, of size $O(1/\delta)^d$.*

Let us first give an intuitive description of the choice of surrogates. Take a node v and put $q := f(c(v))$ for brevity; this is the node location in $\mathbb{R}^d$. We wish to approximately store q with a small number of bits. To this end we pick a point $\bar{q}$ close to q, i.e. such that $\sigma := q - \bar{q}$ has small norm. We then round σ to a vector $\tilde{\sigma}$ using a δ-net, and use $\bar{q} + \tilde{\sigma}$ as the surrogate.

The natural choice for $\bar{q}$ is the ingress of v. Lemma 3.4 then gives a bound on $\|\sigma\|$, which lets us pick δ that provides satisfactory approximation while keeping the storage cost of $\tilde{\sigma}$ small. However, in order to recover the surrogate we also need to store $\tilde{q}$, the location of the ingress, which is too costly. Instead, we choose $\bar{q}$ inductively as the surrogate of the ingress, $\bar{q} := s^*(in(v))$.

We proceed to the formal construction. The surrogates are defined independently in each subtree. Within a subtree T_{sub} of T, we wish to define $s^*(v)$ inductively from $s^*(in(v))$, so we need an ordering for the induction such that a node is always processed after its ingress. We can achieve this by traversing T_{sub} in a DFS order, with the order of traversing the children of each node v (with degree greater than 1) being top-down on $\tau(v)$. Put differently, when we traverse a node v we first process it, and then (recursively) traverse its children in a top-down order by $\tau(v)$. This means that whenever we process a node v, both its parent v' in T_{sub} and its τ-predecessor v_τ have already been traversed. Since it is a DFS scan, and v_τ is a sibling of v in T_{sub}, this means all descendants of v_τ in T_{sub} have already been processed. In particular, since $in(v)$ is by construction either v' or a descendant of v_τ in T_{sub}, it means $in(v)$ has already been processed. As we will refer to this ordering again later on, we call it for brevity the *τ-DFS* ordering of the nodes in a subtree of T.

We now define the induction steps. Denote

$$\delta(v) := \left(5 + \lceil \frac{\Delta(v)}{2^{\ell(v)}} \rceil\right)^{-1}.$$

Induction base: For the root v of the subtree, set $s^*(v) = f(c(v))$.

Inductive step: For a non-root v,

- Let $\text{disp}(v) := f(c(v)) - s^*(in(v))$ be the displacement from the ingress' surrogate.
- Let $\eta^*(v) := \frac{\delta(v)}{2^{\ell(v)}} \cdot \text{disp}(v)$ be the normalized displacement. (We will soon show $\|\eta^*(v)\| \leq 1$.)
- Let $\eta(v)$ be the closest point to $\eta^*(v)$ in the net $\mathcal{N}_{\delta(v)}$.
- Finally, the surrogate is $s^*(v) := s^*(in(v)) + \frac{2^{\ell(v)}}{\delta(v)} \cdot \eta(v)$.

Lemma 3.7. *For every $v \in T$, $\|f(c(v)) - s^*(v)\| \leq 2^{\ell(v)}$.*

Proof. By induction on the τ-DFS ordering within each subtree. In the base case, v is the root and then the claim is trivial since $s^*(v) = f(c(v))$. Now suppose v is not the root. By induction on the ingress we have

$$\|f(c(in(v))) - s^*(in(v))\| \leq 2^{\ell(in(v))},$$

and then by Claim 3.5,

$$\|f(c(in(v))) - s^*(in(v))\| \leq 2 \cdot 2^{\ell(v)}.$$

By Lemma 3.4,

$$\|f(c(v)) - f(c(in(v)))\| \leq 3 \cdot 2^{\ell(v)} + \Delta(v),$$

and together,

$$\|f(c(v)) - s^*(in(v))\| \leq 5 \cdot 2^{\ell(v)} + \Delta(v) \leq \frac{2^{\ell(v)}}{\delta(v)}.$$

This implies $\|\eta^*(v)\| \leq 1$, and since $\mathcal{N}_{\delta(v)}$ is a net for the unit ball, this ensures $\|\eta^*(v) - \eta(v)\| \leq \delta(v)$. Finally,

$$\begin{aligned}
\|f(c(v)) - s^*(v)\| &= \|f(c(v)) - s^*(in(v)) - \tfrac{2^{\ell(v)}}{\delta(v)} \cdot \eta(v)\| \\
&= \|f(c(v)) - s^*(in(v)) - \tfrac{2^{\ell(v)}}{\delta(v)} \cdot (\eta^*(v) - \eta^*(v) + \eta(v))\| \\
&= \|\tfrac{2^{\ell(v)}}{\delta(v)} \cdot (\eta(v) - \eta^*(v))\| \\
&\leq \tfrac{2^{\ell(v)}}{\delta(v)} \cdot \delta(v) \\
&= 2^{\ell(v)}.
\end{aligned}$$

□

For the leaves of each subtree we will actually use a better $\delta(v)$,

$$\delta'(v) := \delta(v) \cdot \epsilon.$$

Then the previous lemma yields

Corollary 3.8. *For $v \in T$ which is a leaf in its subtree, $\|f(c(v)) - s^*(v)\| \leq 2^{\ell(v)} \cdot \epsilon$.*

The corollary follows by simply executing the last round of induction in the proof of Lemma 3.7 with the improved $\delta(v)$.

3.6 The Sketch

In the sketch we store the following information:

- The tree T. For each edge we store whether it is short or long, and for the long edges we store their original lengths.
- For every tree node v we store the center label $c(v)$, the ingress label $in(v)$, the value $\delta(v)^{-1}$ (which is the integer $5 + \lceil \frac{\Delta(v)}{2^{\ell(v)}} \rceil$), and the approximate displacement $\eta(v)$, encoded as an element of $N_{\delta(v)}$ (or $N_{\delta'(v)}$, if v is a leaf in its subtree).

The purpose of storing the lengths of long edges is to compute the levels $\ell(v)$, which we recall are the levels in the uncompressed tree. They are needed in order to recover the surrogates (up to a shift), as will be discussed in Section 4.

We now bound the total size of the sketch. We start with the following observation.

Claim 3.9. (*i*) *The number of long edges in T is at most $2n$.*
(*ii*) *The number of nodes in T which are leaves in their subtree is at most $3n$.*

Proof. For part (*i*), recall that the bottom node of every long edge had degree different than 1 in the original tree (before compression). Since that tree had n leaves, it could only have $2n$ such nodes. Part (*ii*) follows from (*i*) by noting that each node in T which is a leaf in its subtree is either a leaf in the original (non-compressed) tree, or the top node of a long edge. □

Lemma 3.10. *The total sketch size is $O(n(d + \log n)\log(1/\epsilon) + n \log\log \Phi)$.*

Proof. We start by analyzing the space needed to store the tree structure. By Lemma 3.3 the compressed tree T has size $O(n\log(1/\epsilon))$, so its structure can be stored using $O(n\log(1/\epsilon))$ bits. The length of each long edge is bounded by the height of the original tree, which by construction is at most $\log\Phi+1$, so by Claim 3.9 the total storage cost of the lengths is at most $2n\log(\log\Phi+1)$ bits. Overall, the tree structure requires $O(n\log(1/\epsilon)+n\log\log\Phi)$ bits to store. We now analyze the cost of the information stored for each node.

- Centers: Each center is a label in X and hence takes $\log n$ bits to store. In total, $O(n\log(1/\epsilon)\cdot\log n)$ bits.
- Ingresses: $in(v)$ is a node in T, but we can further observe that $in(v)$ is either the parent of v or a node which is a leaf in its subtree. Therefore by Claim 3.9 the ingress is one of $O(n)$ possible nodes, and takes $O(\log n)$ bits to store. In total, $O(n\log(1/\epsilon)\cdot\log n)$ bits.
- Precisions: Their total storage cost is

$$\sum_{v\in T}\log\left(\frac{1}{\delta(v)}\right)=\sum_{v\in T}\log\left(5+\lceil\frac{\Delta(v)}{2^{\ell(v)}}\rceil\right)\le 3|T|+\sum_{v\in T}\log\left(\frac{\Delta(v)}{2^{\ell(v)}}\right).$$

  Since $\sum_{v\in T}\log\left(\frac{\Delta(v)}{2^{\ell(v)}}\right)=O(|T|)=O(n\log(1/\epsilon))$ (see the proof of Lemma 3.3), the total storage cost of the precisions is $O(n\log(1/\epsilon))$ bits.
- Displacements: By Lemma 3.6, $\eta(v)$ is a point in a set of size $O(1/\delta(v))^d$, hence storing $\eta(v)$ takes $d\log(1/\delta(v))$ bits. Summing over all $v\in T$ we get $O(dn\log(1/\epsilon))$ bits, as shown above for the precisions. For the leaves of every subtree we kept a displacement up to an improved approximation, $\delta'(v)=\delta(v)\cdot\epsilon$. This adds $d\log(1/\epsilon)$ bits per v, and since by Claim 3.9 there are $O(n)$ such nodes, in total this consumes additional $O(nd\log(1/\epsilon))$ bits.

In total, $O(n(d+\log n)\log(1/\epsilon)+n\log\log\Phi)$ bits. □

4 Estimation Algorithm

We now show how to use the sketch to produce a $(1\pm\epsilon)$-approximation for the distance between any two points in X. The key point is that within each subtree, we can recover the surrogates up to a fixed (unknown) shift from the sketch. Formally, for every $v\in T$ we define the *shifted surrogate* $s(v)\in\mathbb{R}^d$:

- If v is the root of its subtree, set $s(v):=\mathbf{0}$ (the origin in $\mathbb{R}^d$).
- Otherwise, set $s(v):=s(in(v))+\frac{2^{\ell(v)}}{\delta(v)}\cdot\eta(v)$.

Observe that we can indeed compute the shifted surrogate from the sketch: For every v we have stored explicitly $in(v)$, $\ell(v)$ (inherent in storing the tree structure), $\delta(v)^{-1}$, and an encoding of $\eta(v)$ as an element in $\mathcal{N}_{\delta(v)}$ that can now be decoded. With those at hand, we can compute the shifted surrogates inductively in the τ-DFS order on the subtree.

Furthermore, by comparing this construction to that of Section 3.5, it is straightforward to see that for every node $v\in T$ we have $s(v)=s^*(v)-s^*(r)$, where r is the root of the subtree in which v resides. This means that the shifted surrogates within every subtree are the same as the original surrogates up to a fixed shift $s^*(r)$ (which cannot be recovered from the sketch, since it equals $f(c(r))$ and we never stored the true embedding of any point in the sketch). Hence,

Claim 4.1. *For every $v, v' \in T$ which are in the same subtree, $\|s(v) - s(v')\| = \|s^*(v) - s^*(v')\|$.*

Now given $x, y \in X$, we show to how to compute from the sketch a $(1 \pm \epsilon)$-estimate for $\|f(x) - f(y)\|$. Let u be the lowest common ancestor of leaf(x) and leaf(y). Let v_x be the lowest node on the path from u down to leaf(x) that does not traverse a long edge. Similarly define v_y for y. Note that u, v_x, v_y are all in the same subtree, and v_x, v_y are leaves in that subtree. See Figure 3 for illustration. The estimate we return is $\|s(v_x) - s(v_y)\|$. By Claim 4.1 it equals $\|s^*(v_x) - s^*(v_y)\|$, so our goal is to prove

$$\|s^*(v_x) - s^*(v_y)\| = (1 \pm O(\epsilon)) \cdot \|f(x) - f(y)\|. \tag{6}$$

By the triangle inequality we have

$$\|s^*(v_x) - s^*(v_y)\| = \|f(x) - f(y)\| \pm (\|f(x) - s^*(v_x)\| + \|f(y) - s^*(v_y)\|)\,. \tag{7}$$

Now consider two cases for v_x:

- If $v_x = \text{leaf}(x)$ then $c(v_x) = x$, and hence by Corollary 3.8, $\|f(x) - s^*(v_x)\| \le 2^{\ell(v_x)}\epsilon$.
- Otherwise, v_x is the top node of a long edge. Let k be its original length and w_x its bottom node. Recall that by the construction, $k > \log(\Delta(w_x)/2^{\ell(w_x)}) + \log(1/\epsilon)$. Also note that $c(v_x) = c(w_x)$. Then

$$\|f(x) - f(c(v_x))\| = \|f(x) - f(c(w_x))\| \le \Delta(w_x) = 2^{\ell(w_x) + \log(\frac{\Delta(w_x)}{2^{\ell(w_x)}})} < 2^{\ell(w_x) + k - \log(1/\epsilon)} = 2^{\ell(v_x)}\epsilon.$$

Combining this with Corollary 3.8, we get by the triangle inequality that $\|f(x) - s^*(v_x)\| \le 2 \cdot 2^{\ell(v_x)}\epsilon$. This bound holds in both the the above cases. Similarly one shows $\|f(y) - s^*(v_y)\| \le 2 \cdot 2^{\ell(v_y)}\epsilon$. Since $\ell(v_x) \le \ell(u) - 1$ and $\ell(v_y) \le \ell(u) - 1$, we can add these and obtain

$$\|f(x) - s^*(v_x)\| + \|f(y) - s^*(v_y)\| \le 2 \cdot 2^{\ell(u)}\epsilon.$$

By the construction of T, the fact that u is the lowest common ancestor of leaf(x) and leaf(y) implies $\|f(x) - f(y)\| \ge 2^{\ell(u)-1}$ (see Observation 3.2). Plugging this into the equation above yields

$$\|f(x) - s^*(v_x)\| + \|f(y) - s^*(v_y)\| \le \|f(x) - f(y)\| \cdot 4\epsilon,$$

and plugging this into eq. (7) proves eq. (6), which proves Theorem 2.1.

5 Running Times

To analyze running times, we need an efficient version of Lemma 3.6. We prove the following lemma in Appendix B.

Lemma 5.1. *For every $\delta > 0$, the ℓ_p unit ball in $\mathbb{R}^d$ has a δ-net $\mathcal{N}_\delta$ such that*

1. *Given $\eta^* \in \mathbb{R}^d$ with $\|\eta^*\|_p \le 1$, one can find a δ-close vector $\eta \in \mathcal{N}_\delta$ in time $O(d)$.*
2. *A vector $\eta \in \mathcal{N}_\delta$ can be encoded as a bitstring of length $O(d \log(1/\delta))$, in time $O(\frac{1}{\delta} d^{1+1/p})$.*
3. *Given the bitstring encoding as above, the coordinates of η in $\mathbb{R}^d$ can be recovered in time $O(\frac{1}{\delta} d^{1+1/p})$.*

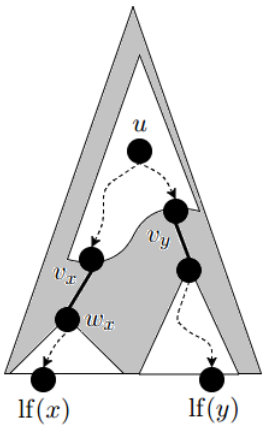


Figure 3: The estimate for $\|f(x) - f(y)\|$ is $\|s(v_x) - s(v_y)\|$. The external shaded triangle is the tree T. The white regions are subtrees. The dashed arrows are downward paths in T. The thick arcs are long edges.

Summary time. We spend $O(n^2 \log \Phi)$ time setting up the distances graph and building and compressing the tree. Then, the processing time for every node $v \in T$ is dominated by encoding the $\delta(v)$-net vectors, which by Lemma 5.1 takes time $O(d^{1+1/p}/\delta(v))$. Summing over the nodes, and recalling that $\delta(v) \geq \epsilon \cdot \left(5 + \lceil \frac{\Delta(v)}{2^{\ell(v)}} \rceil\right)^{-1} \geq \epsilon \cdot \left(6 + \frac{\Delta(v)}{2^{\ell(v)}}\right)^{-1}$, we get

$$\sum_{v \in T} \frac{d^{1+1/p}}{\delta(v)} \leq \frac{d^{1+1/p}}{\epsilon} \left(6|T| + \sum_{v \in T} \frac{\Delta(v)}{2^{\ell(v)}}\right) = \frac{d^{1+1/p}}{\epsilon} \cdot O\left(n \log\left(\frac{1}{\epsilon}\right)\right). \tag{8}$$

(See the proof of Lemma 3.3 for the latter bound.) The total summary time is $O(n^2 \log \Phi + nd^{1+1/p}\epsilon^{-1} \log(1/\epsilon))$.

Observation 5.2. *Note that in the Euclidean case, the $n^2 \log \Phi$ term in the running time bound can be reduced to $O(n^{1+\alpha} \log \Phi)$ for any constant $\alpha > 0$, at the cost of increasing the sketch size by a multiplicative factor of α^{-1}. (The Johnson-Lindenstrauss transform, which we also use as a preceding step, can be executed in time $O(\epsilon^{-2} n \log n)$ [AC09].) To this end, set $c := \alpha^{-1/2}$. In constructing the tree, we use the algorithm of [HPIM12] to compute c-approximate connected components in each level. Their algorithm is based on Locality-Sensitive Hashing (LSH), which in Euclidean spaces can be implemented in time $O(n^{1+1/c^2})$ [AI06]. Using c-approximate connected components means that clusters in level ℓ of the tree can be merged if the distance between them is at most $c \cdot 2^\ell$ (rather than just 2^ℓ), and to account for this constant loss, we need to scale ϵ down to ϵ/c. Since the dependence of the sketch size on ϵ is $\log(1/\epsilon)/\epsilon^2$, the multiplicative loss in the sketch size is $c^2 = \alpha^{-1}$.*

Estimation Time. Since the height of the tree is at most $\log \Phi + 2$, we spend $O(\log \Phi)$ time finding the lowest common ancestor of leaf(x), leaf(y) and finding v_x, v_y. Then we need to compute the shifted surrogates $s(v_x), s(v_y)$. Due to the inductive definition of the $s(v_x)$, in order to compute $s(v_x)$ we need to traverse τ-predecessors backwards until we reach the root of the subtree, whose shifted surrogate is known to be $\mathbf{0}$. In the worst case we might traverse all nodes in T. For each node v we need to decode the $\delta(v)$-net vector $\eta(v)$, which by Lemma 5.1 takes time $O(d^{1+1/p}/\delta(v))$. Applying eq. (8) again, we see that the total estimation time is $O(\log \Phi + nd^{1+1/p}\epsilon^{-1} \log(1/\epsilon))$.

In practical settings such query time is often considered prohibitive. We now describe a modification to our scheme that yields a different trade-off between the sketch size and the estimation time. In particular, letting

$$K := \lceil \log(2 \cdot \Phi \cdot \epsilon^{-1} \cdot d^{1/p}) \rceil,$$

we show how to achieve estimation time of $O(\log \Phi + dK)$ in the expense of increasing the sketch size by a factor of $\log d$. To demonstrate why this is beneficial, consider a typical Euclidean setting in which $d = O(\epsilon^{-2} \log n)$ (by Johnson-Lindenstrauss dimension reduction) and $\Phi = \text{poly}(n)$. Theorem 2.2 gives a sketch size of $O(\epsilon^{-2} \log(1/\epsilon) \cdot n \log n)$ bits with $\tilde{O}(\epsilon^{-4} n)$ estimation time.[1] The modification increases the sketch size by a factor of $O(\log \log n + \log(1/\epsilon))$, and improves the estimation time to $\tilde{O}(\epsilon^{-2} \log^2 n)$.

The first estimation bottleneck is decoding the net vectors, and we resolve this by replacing the δ-net from Lemma 5.1 with the uniform grid $(\frac{\delta}{d^{1/p}} \mathbb{Z})^d$. In contrast, Lemma 5.1 uses the intersection of this grid with the unit ball. We can store a point in this grid using $O(d \log(1/\delta) + d \log d)$ bits without any encoding, which adds $O(d \log d)$ bits per point over Lemma 5.1. In total, the sketch size increases by a factor of $O(\log d)$, and the processing time of a node v decreases to $O(d)$.

The second estimation bottleneck is computing the shifted surrogates by induction on the τ-predecessors all the way back to the subtree root. We resolve this by storing some shifted surrogates explicitly in the sketch. This is done separately in each subtree T', as follows.

1. Construct the tree T'_τ on the nodes of T', by attaching each node as a child of its τ-predecessor.
2. Pick $\lceil |T'_\tau|/K \rceil$ nodes in T'_τ, called *landmark nodes*, such that for every $v \in T'_\tau$, we can reach a landmark node from v by going upward in T'_τ at most K steps. This can be done as follows: Start with a lowest node $v \in T'_\tau$; climb upward K steps (or less if the root is reached), to a node $\hat{v}$; declare $\hat{v}$ a landmark node, remove it from T'_τ with all its decendants, and iterate. Since every iteration but the last removes at least K nodes from T'_τ, we finish with at most $\lceil |T'_\tau|/K \rceil$ landmark nodes.

For every landmark node $\hat{v}$ we explicitly store in the sketch the shifted surrogate $s(\hat{v})$. Now, in order to compute $s(v)$ of any given node v, we need to trace the τ-predecessors backward at most K times until we reach a landmark nodes whose shifted surrogate is known. The computation time per node is $O(d)$, so in total, the resulting estimation time is $O(\log \Phi + dK)$.

It remains to verify that storing the shifted surrogates for the landmark nodes does not asymptotically increase the sketch size. To this end, fix a landmark node $\hat{v}$. Recall that the shifted surrogates are defined recursively, starting at $\mathbf{0}$ for the subtree root, and then in each step adding a vector of the form $\delta(v)^{-1} 2^{\ell(v)} \eta(v)$ (see Section 4). Since $\eta(v)$ is a point on a grid with side either $\delta'(v)/d^{1/p} = \delta(v)\epsilon/d^{1/p}$ (if v is a leaf in its subtree) or $\delta(v)/d^{1/p}$ (otherwise), we see that each step adds an integer multiple of either $\epsilon/d^{1/p}$ or $1/d^{1/p}$ to each coordinate of $s(\hat{v})$. On the other hand, since $\mathbf{0}$ is also a shifted surrogate (of the center of the root of the subtree in which $\hat{v}$ is present), we must have $\|s(\hat{v}) - \mathbf{0}\| \leq (1+\epsilon)\Phi$, and in particular each coordinate of $s(\hat{v})$ is bounded by 2Φ. Together, we see that each coordinate of $s(\hat{v})$ can be represented with $\lceil \log(2\Phi \cdot \epsilon^{-1} \cdot d^{1/p}) \rceil = K$ bits. Multiplying by d coordinates, we find that $O(dK)$ bits suffice to fully store any shifted surrogate. Since we are storing them for $O(|T|/K)$ landmark nodes, we spend additional $O(d|T|) = O(nd \log(1/\epsilon))$ bits, which does not asymptotically change the sketch size.

[1] We use $\tilde{O}(f)$ to denote $O(f \cdot \text{polylog}(f))$.

6 Lower Bounds

Theorem 6.1 (Euclidean metrics)**.** *Fix $\gamma > 0$. If $\epsilon \geq 1/n^{0.5-\gamma}$, then*

$$b_{1+\epsilon}(\mathcal{D}_2(n, d, \Phi)) = \Omega(\gamma \cdot \epsilon^{-2} n \log n + n \log\log \Phi).$$

Proof. Denote $k := 1/\epsilon^2$. Note that since $\epsilon > 1/\sqrt{n}$, we may assume w.l.o.g. that k is an integer. Let B be the set of standard basis vectors in $\mathbb{R}^n$, and let $a_1, \ldots, a_n$ be an arbitrary sequence of k-sparse vectors in $\{0,1\}^n$ (note that we allow repetitions). Denote $A := \{a_1, \ldots, a_n\}$. We sketch the Euclidean metric on the set $(\frac{1}{\sqrt{k}}A) \cup B$ up to distortion $1 \pm \frac{1}{2}\epsilon$. We can also keep track of repeating elements in $a_1, \ldots, a_n$ using $n \log n$ bits (details omitted).

For every a_j and $i \in \{1, \ldots, n\}$ we have $a_j^T e_i = a_j(i)$ and hence

$$\|\frac{1}{\sqrt{k}} a_j - e_i\|_2^2 = 2 - \frac{2}{\sqrt{k}} a_j(i) = 2 - 2\epsilon a_j(i).$$

Since the sketch allows us to recover distances up to distortion $1 \pm \frac{1}{2}\epsilon$, we can recover each entry $a_j(i)$ of each a_j, and hence the entire sequence $a_1, \ldots, a_n$. The number of choices for this sequence is $\binom{n}{k}^n$, so the lower bound we get on the sketch size in bits is

$$\log\left(\binom{n}{k}^n\right) \geq nk \log\left(\frac{n}{k}\right) = \frac{n}{\epsilon^2} \cdot \log(n\epsilon^2) = \Omega(\gamma \cdot \epsilon^{-2} n \log n),$$

where the final bound is since $\log(n\epsilon^2) \geq \log(n^{2\gamma}) = 2\gamma \log n$.

Next we prove the lower bound $\Omega(n \log\log \Phi)$. Suppose w.l.o.g. that $\log \Phi$ is an integer. Consider the point set $X = \{1, \ldots, n\}$. Define a map $f : X \to \mathbb{R}$ by setting $g(1) := 0$, and for every $x \in X \setminus \{1\}$ setting $g(x) := 2^{\phi(x)}$ for an arbitrary $\phi(x) \in \{1, \ldots, \log \Phi\}$. This defines a set of $(\log \Phi)^{n-1}$ one-dimensional Euclidean embeddings of X, each of which induces a metric contained in $\mathcal{D}_2(n, 1, \Phi)$. We can fully recover a map g from this family given a sketch with distortion better than 2, since $D(1, x) = g(x)$ for every $x \in X$. Therefore, sketching those metrics requires at least $\log\left((\log \Phi)^{n-1}\right) = \Omega(n \log\log \Phi)$ bits.

To get the final lower bound $\Omega(\gamma \cdot \epsilon^{-2} n \log n + n \log\log \Phi)$, we augment the two metric famiilies constructed above into one. We constructed a family $\mathcal{F}_1$ of metrics embedded in $\mathbb{R}^n$, of size $|\mathcal{F}_1| \geq 2^{\Omega(\gamma \cdot \epsilon^{-2} n \log n)}$, and a family $\mathcal{F}_2$ of metrics embedded in $\mathbb{R}^1$, of size $|\mathcal{F}_2| \geq 2^{\Omega(n \log\log \Phi)}$. For every $D' \in \mathcal{F}_1$ and $D" \in \mathcal{F}_2$, we can naturally define a metric $D' \oplus D"$ embedded in $\mathbb{R}^{n+1}$ by embedding D' in the first n dimensions and $D"$ in the remaining dimension. This defines a family $\mathcal{F} := \{D' \oplus D" : D' \in \mathcal{F}_1, D" \in \mathcal{F}_2\}$ contained in $\mathcal{D}_2(2n, n+1, \Phi)$ of size $|\mathcal{F}_1| \cdot |\mathcal{F}_2|$, such that a b-bit sketching scheme with distortion $1 + \epsilon$ can recover a metric from $\mathcal{F}$, and the lower bound $b = \Omega(\gamma \cdot \epsilon^{-2} n \log n + n \log\log \Phi)$ follows. □

Theorem 6.2 (general metrics)**.**

$$b_{1+\epsilon}(\mathcal{D}_{all}(n, \Phi)) = \Omega(n^2 \log(1/\epsilon) + n \log\log \Phi).$$

Proof. Let $\epsilon > 0$ and suppose w.l.o.g. $1/\epsilon$ is an integer. Recall we use the convention $X = \{1, \ldots, n\}$. For every $x, y \in X$, $x < y$ set $d(x, y) = 1 + k(x, y) \cdot \epsilon$ for an arbitrary $k(x, y) \in \{0, \ldots, 1/\epsilon\}$. This actually defines a metric regardless of the choice of k's: we only need to verify the triangle inequality, and it holds trivially since all pairwise distances are lower-bounded by 1 and upper-bounded by 2. Hence we have defined a family of $(1/\epsilon)^{\binom{n}{2}}$ metrics. Next observe that a sketch with distortion $(1 \pm \frac{1}{2}\epsilon)$ is sufficient to fully recover a metric from this family, which proves a lower bound of $\log\left((1/\epsilon)^{\binom{n}{2}}\right) = \Omega(n^2 \log(1/\epsilon))$ on the sketch size in bits. The other lower bound $\Omega(n \log\log \Phi)$ is by the same proof as Theorem 6.1. □

Acknowledgments. We thank Arturs Backurs, Sepideh Mahabadi and Ilya Razenshteyn for helpful feedback on this manuscript. This work was supported in part by the NSF, MADALGO and the Simons Foundation.

References

[AC09] Nir Ailon and Bernard Chazelle, *The fast johnson–lindenstrauss transform and approximate nearest neighbors*, SIAM J. Comput. **39** (2009), no. 1, 302–322.

[Ach03] Dimitris Achlioptas, *Database-friendly random projections: Johnson-lindenstrauss with binary coins*, J. Comput. Syst. Sci. **66** (2003), no. 4, 671–687.

[AI06] Alexandr Andoni and Piotr Indyk, *Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions*, 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Berkeley, California, USA, Proceedings, 2006, pp. 459–468.

[AK16] Noga Alon and Bo'az Klartag, *Optimal compression of approximate euclidean distances*, arXiv preprint arXiv:1610.00239 (2016).

[Alo03] Noga Alon, *Problems and results in extremal combinatoricsi*, Discrete Mathematics **273** (2003), no. 13, 31 – 53, EuroComb'01.

[Bar96] Yair Bartal, *Probabilistic approximation of metric spaces and its algorithmic applications*, Foundations of Computer Science, 1996. Proceedings., 37th Annual Symposium on, IEEE, 1996, pp. 184–193.

[dBÁGB+13] Guillermo de Bernardo, Sandra Álvarez-García, Nieves R. Brisaboa, Gonzalo Navarro, and Oscar Pedreira, *Compact querieable representations of raster data*, pp. 96–108, Springer International Publishing, Cham, 2013.

[GGNL+15] Travis Gagie, Javier I. González-Nova, Susana Ladra, Gonzalo Navarro, and Diego Seco, *Faster compressed quadtrees*, Proceedings of the 2015 Data Compression Conference (Washington, DC, USA), DCC '15, IEEE Computer Society, 2015, pp. 93–102.

[HPIM12] Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani, *Approximate nearest neighbor: Towards removing the curse of dimensionality*, Theory of Computing **8** (2012), no. 14, 321–350.

[Hud09] Benoît Hudson, *Succinct representation of well-spaced point clouds*, arXiv preprint arXiv:0909.3137 (2009).

[JL84] William B Johnson and Joram Lindenstrauss, *Extensions of lipschitz mappings into a hilbert space*, Contemporary mathematics **26** (1984), no. 189-206, 1–1.

[JW13] Thathachar S Jayram and David P Woodruff, *Optimal bounds for johnson-lindenstrauss transforms and streaming problems with subconstant error*, ACM Transactions on Algorithms (TALG) **9** (2013), no. 3, 26.

[KOR98] Eyal Kushilevitz, Rafail Ostrovsky, and Yuval Rabani, *Efficient search for approximate nearest neighbor in high dimensional spaces*, Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '98, ACM, 1998, pp. 614–623.

[LN16] Kasper Green Larsen and Jelani Nelson, *Optimality of the johnson-lindenstrauss lemma*, arXiv preprint arXiv:1609.02094 (2016).

[MWY13] Marco Molinaro, David P Woodruff, and Grigory Yaroslavtsev, *Beating the direct sum theorem in communication complexity with implications for sketching*, Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, 2013, pp. 1738–1756.

[PS89] David Peleg and Alejandro A Schäffer, *Graph spanners*, Journal of graph theory **13** (1989), no. 1, 99–116.

[Sam88] Hanan Samet, *An overview of quadtrees, octrees, and related hierarchical data structures*, Theoretical Foundations of Computer Graphics and CAD, Springer, 1988, pp. 51–68.

[TZ05] Mikkel Thorup and Uri Zwick, *Approximate distance oracles*, Journal of the ACM (JACM) **52** (2005), no. 1, 1–24.

[VM14] Prayaag Venkat and David M. Mount, *A succinct, dynamic data structure for proximity queries on point sets*, Proceedings of the 26th Canadian Conference on Computational Geometry, CCCG 2014, Halifax, Nova Scotia, Canada, 2014, 2014.

A ℓ_1 Metrics

In this section we point out that in our setting, both upper and lower bounds for Euclidean metrics apply to ℓ_1 metrics as well. In particular,

Corollary A.1.

$$b_{1+\epsilon}(\mathcal{D}_1(n,d,\Phi)) = O(\epsilon^{-2}\log(1/\epsilon)\cdot n\log n + n\log\log\Phi),$$

and

$$b_{1+\epsilon}(\mathcal{D}_1(n,d,\Phi)) = \Omega(\epsilon^{-2} n\log n + n\log\log\Phi).$$

Proof. The upper bound follows from Theorem 2.2 since every ℓ_1-metric is of negative type, meaning it embeds isometrically into ℓ_2^2. Then it is enough to sketch the underlying Euclidean metric. The lower bound follows from Theorem 6.1 since ℓ_2 metrics embed isometrically into ℓ_1. □

B Grid Nets

In this section we prove Lemma 5.1. For $x \in \mathbb{R}^d$ and $r > 0$, denote by $\mathcal{B}^d(x,r)$ the radius-r ball centered at x in the ℓ_p norm. Let $\mathcal{G}_\delta^d$ be the uniform grid with side $\delta d^{-1/p}$ in $\mathbb{R}^d$. The δ-net for the ball would be its intersection with the grid,

$$\mathcal{N}_\delta^d(x,r) := \mathcal{B}^d(x,r) \cap \mathcal{G}_\delta^d.$$

Clearly, given a point in the ball, we can find a δ-close point in the net in time $O(d)$, by rounding each coordinate either up or down to an integer multiple of the grid side $\delta d^{-1/p}$. It remains to show how to encode and decode points in the net to bitstrings. For clarity, we present the proof for $p = 2$; the analysis for any $1 \le p \le \infty$ goes through with only a change of constants. Denote

$$M_\delta^d(r) := \lceil \left(\frac{4\sqrt{\pi} \cdot r}{\delta} \right)^d \rceil.$$

We now show that $M_\delta^d(r)$ is an upper bound on the size of the net $\mathcal{N}_\delta^d(x, r)$.

Fact B.1. $\sum_{i=1}^m (m^2 - i^2)^k \le \sqrt{\frac{\pi}{2k}} \cdot m^{2k+1}$.

Proof.

$$\frac{1}{m^{2k+1}} \sum_{i=1}^m (m^2 - i^2)^k = \sum_{i=1}^m \left(1 - \left(\frac{i}{m}\right)^2\right)^k \frac{1}{m} \le \int_0^1 (1 - x^2)^k dx = \frac{\sqrt{\pi}}{2} \cdot \frac{\Gamma(k+1)}{\Gamma(k+1.5)} \le \sqrt{\frac{\pi}{2k}}.$$

□

Claim B.2.

$$M_\delta^d(r) \ge \sum_{i=-\lfloor r\sqrt{d}/\delta \rfloor}^{\lfloor r\sqrt{d}/\delta \rfloor} M_\delta^{d-1}\left(\sqrt{r^2 - (\tfrac{\delta}{\sqrt{d}} i)^2}\right).$$

Proof.

$$\begin{aligned}
&\sum_{i=1}^{\lfloor r\sqrt{d}/\delta \rfloor} M_\delta^{d-1}\left(\sqrt{r^2 - (\tfrac{\delta}{\sqrt{d}} i)^2}\right) \\
&\le \lfloor \frac{r\sqrt{d}}{\delta} \rfloor + \sum_{i=1}^{\lfloor r\sqrt{d}/\delta \rfloor} \left(\frac{4\sqrt{\pi} \cdot \sqrt{r^2 - (\tfrac{\delta}{\sqrt{d}} i)^2}}{\delta} \right)^{d-1} \\
&= \lfloor \frac{r\sqrt{d}}{\delta} \rfloor + \left(\frac{4\sqrt{\pi}}{\sqrt{d}}\right)^{d-1} \sum_{i=1}^{\lfloor r\sqrt{d}/\delta \rfloor} \left(\left(\frac{r\sqrt{d}}{\delta}\right)^2 - i^2 \right)^{\frac{d-1}{2}} \\
&\le \lfloor \frac{r\sqrt{d}}{\delta} \rfloor + \left(\frac{4\sqrt{\pi}}{\sqrt{d}}\right)^{d-1} \sum_{i=1}^{\lceil r\sqrt{d}/\delta \rceil} \left(\left(\lceil \frac{r\sqrt{d}}{\delta} \rceil\right)^2 - i^2 \right)^{\frac{d-1}{2}} \\
&\le \lfloor \frac{r\sqrt{d}}{\delta} \rfloor + \left(\frac{4\sqrt{\pi}}{\sqrt{d}}\right)^{d-1} \cdot \sqrt{\frac{\pi}{d-1}} \cdot \left(\lceil \frac{r\sqrt{d}}{\delta} \rceil\right)^d \qquad \text{by Fact B.1} \\
&\le \frac{1}{3} \left(\frac{4\sqrt{\pi} \cdot r}{\delta} \right)^d \le \frac{1}{3} M_\delta^d(r).
\end{aligned}$$

Therefore, letting $r_i := \sqrt{r^2 - (\frac{\delta}{\sqrt{d}} i)^2}$,

$$\sum_{i=-\lfloor r\sqrt{d}/\delta \rfloor}^{\lfloor r\sqrt{d}/\delta \rfloor} M_\delta^{d-1}(r_i) = \sum_{i=-\lfloor r\sqrt{d}/\delta \rfloor}^{-1} M_\delta^{d-1}(r_i) + M_\delta^{d-1}(r) + \sum_{i=1}^{\lfloor r\sqrt{d}/\delta \rfloor} M_\delta^{d-1}(r_i) \le M_\delta^d(r).$$

□

Corollary B.3. $\left|\mathcal{N}_\delta^d(x,r)\right| \le M_\delta^d(r) = O(r/\delta)^d$.

Proof. By induction on d. In the base case $d=1$, clearly $\mathcal{N}_\delta^1(x,r) \le \lceil r/\delta \rceil$ and the bound holds. For $d>1$, assume w.l.o.g. $x_1 = 0$, let x_{-1} denote the projection of x on its $d-1$ last coordinates. It is a simple observation that for any $\alpha \in [-r,r]$, the points $y \in \mathcal{B}_\delta^d(x,r)$ with $y_1 = \alpha$ form a $(d-1)$-dimensional ball of radius $\sqrt{r^2-\alpha^2}$. Therefore, by grouping the points in $\mathcal{N}_\delta^d(x,r)$ by their first coordinate value, we can write the grid net as a disjoint union of grid nets in $d-1$ dimensions. More precisely, denoting $r_i := \sqrt{r^2 - (\frac{\delta}{\sqrt{d}}i)^2}$,

$$\mathcal{N}_\delta^d(x,r) = \bigcup_{i=-\lfloor r\sqrt{d}/\delta \rfloor}^{\lfloor r\sqrt{d}/\delta \rfloor} \{(\tfrac{\delta}{\sqrt{d}}i, y) : y \in \mathcal{N}_\delta^{d-1}(x_{-1}, r_i)\}.$$

Then by induction,

$$\left|\mathcal{N}_\delta^d(x,r)\right| \le \sum_{i=-\lfloor r\sqrt{d}/\delta \rfloor}^{\lfloor r\sqrt{d}/\delta \rfloor} \left|\mathcal{N}_\delta^{d-1}(x_{-1}, r_i)\right| \le \sum_{i=-\lfloor r\sqrt{d}/\delta \rfloor}^{\lfloor r\sqrt{d}/\delta \rfloor} M_\delta^{d-1}(r_i),$$

and the bound follows from Claim B.2. □

Encoding and decoding. We map vectors in $\mathcal{N}_\delta^d(x,r)$ to integers in the range $1,\dots,M_\delta^d(r)$, or equivalently bitstring of length $\log(M_\delta^d(r))$, as follows. For $i = -\lfloor \frac{r\sqrt{d}}{\delta} \rfloor, ..., \lfloor \frac{r\sqrt{d}}{\delta} \rfloor$, we partition the range to segments of lengths $M_\delta^{d-1}(r_i)$; Claim B.2 ensures the sum of segments does not exceed $M_\delta^d(r)$. We group the vectors in the net by their first coordinate value, setting $N_i = \{\eta \in \mathcal{N}_\delta^d(x,r) : \eta_1 = \frac{\delta}{\sqrt{d}}i\}$, and we map N_i to the segment of length $M_\delta^{d-1}(r_i)$; Corollary B.3 ensures the segment is large enough. Within each segment, the mapping is defined recursively, by recalling that N_i projected on the last $d-1$ coordinates is the net $\mathcal{N}_\delta^{d-1}(x_{-1}, r_i)$.

In order to encode a given vector, we need to compute the $\frac{2r\sqrt{d}}{\delta}$ segment sizes $M_\delta^{d-1}(r_i)$, pick a segment according to the first coordinate, and recurse on the remaining coordinates. The total encoding time is hence $O(d \cdot \frac{r\sqrt{d}}{\delta})$. In order to decode a given encoding, we again need to compute the segment sizes in order to the determine the first coordinate, and then recurse on offset within the current segment. The decoding time is again $O(d^{1.5}r/\delta)$.