Proceedings of Machine Learning Research vol 75:1–25, 2018 31st Annual Conference on Learning Theory

Approximate Nearest Neighbors in Limited Space

Piotr Indyk INDYK@MIT.EDU
CSAIL, MIT

Tal Wagner TALW@MIT.EDU
CSAIL, MIT

Editors: Sebastien Bubeck, Vianney Perchet and Philippe Rigollet

Abstract

We consider the $(1+\epsilon)$-approximate nearest neighbor search problem: given a set X of n points in a d-dimensional space, build a data structure that, given any query point y, finds a point $x \in X$ whose distance to y is at most $(1+\epsilon)\min_{x\in X}\|x-y\|$ for an accuracy parameter $\epsilon \in (0,1)$. Our main result is a data structure that occupies only $O(\epsilon^{-2} n \log(n) \log(1/\epsilon))$ bits of space, assuming all point coordinates are integers in the range $\{-n^{O(1)} \dots n^{O(1)}\}$, i.e., the coordinates have $O(\log n)$ bits of precision. This improves over the best previously known space bound of $O(\epsilon^{-2} n \log(n)^2)$, obtained via the randomized dimensionality reduction method of Johnson and Lindenstrauss (1984). We also consider the more general problem of estimating all distances from a collection of query points to all data points X, and provide almost tight upper and lower bounds for the space complexity of this problem.

Keywords: nearest neighbor, quantization, distance estimation, metric compression, distance sketches, dimension reduction

1. Introduction

The nearest neighbor search problem is defined as follows: given a set X of n points in a d-dimensional space, build a data structure that, given any query point y, returns the point in X closest to y. For efficiency reasons, the problem is often relaxed to approximate nearest neighbor search, where the goal is to find a point $x \in X$ whose distance to y is at most $c\min_{x\in X}\|x-y\|$ for some approximation factor $c > 1$. Both problems have found numerous applications in machine learning, computer vision, information retrieval and other areas. In machine learning in particular, nearest neighbor classifiers are popular baseline methods whose classification error often comes close to that of the best known techniques (Efros (2017)).

Developing fast approximate nearest neighbor search algorithms have been a subject of extensive research efforts over the last last two decades, see e.g., Shakhnarovich et al. (2006); Andoni and Indyk (2017) for an overview. More recently, there has been increased focus on designing nearest neighbor methods that use a limited amount of *space*. This is motivated by the need to fit the data set in the main memory (Johnson et al. (2017b,a)) or an Internet of Things device (Gupta et al. (2017)). Furthermore, even a simple linear scan over the data is more time-efficient if the data is compressed. The data set compression is most often achieved by developing compact representations of data that approximately preserve the distances between the points (see Wang et al. (2016) for a survey). Such representations are smaller than the original (uncompressed) representation of the data set, while approximately preserving the distances between points.

Most of the approaches in the literature are only validated empirically. The currently best known *theoretical* tradeoffs between the representation size and the approximation quality are summarized in Table 1, together with their functionalities and constraints.

	Bits per point	Comments
No compression	$d \log n$	
Johnson and Lindenstrauss (1984)	$\epsilon^{-2} \log^2(n)$	Estimates distances between any y and all $x \in X$
Kushilevitz et al. (2000)	$\epsilon^{-2} \log(n) \log(R)$	Estimates distances between any y and all $x \in X$, assuming $\|x - y\| \in [r, Rr]$
Indyk and Wagner (2017)	$\epsilon^{-2} \log(n) \log(1/\epsilon)$	Estimates distances between all $x, y \in X$, does not provably support out-of-sample queries
This paper	$\epsilon^{-2} \log(n) \log(1/\epsilon)$	Returns an approximate nearest neighbor of y in X

Table 1: Comparison of Euclidean metric sketches with distortion $1 \pm \epsilon$. We assume that all point coordinates are represented using $\log \Phi$ bits, or alternatively that each coordinate is an integer in the range $\{-\Phi \ldots \Phi\}$. For the sake of exposition, the results depicted in the table assume $\Phi = n^{O(1)}$. Furthermore, the compression algorithm can be randomized, and the compressed representation must enable approximating distances up to a factor of $1 \pm \epsilon$ with probability $1/n^{O(1)}$.

Unfortunately, in the context of approximate nearest neighbor search, the above representations lead to sub-optimal results. The result from the last row of the table (from Indyk and Wagner (2017)) cannot be used to obtain provable bounds for nearest neighbor search, because the distance preservation guarantees hold only for pairs of points in the pointset X.[1] The second-to-last result (from Kushilevitz et al. (2000)) only estimates distances in a certain range; extending this approach to all distances would multiply the storage by a factor of $\log \Phi$. Finally, the representations obtained via a direct application of randomized dimensionality reduction (Johnson and Lindenstrauss (1984)) are also larger than the bound from Indyk and Wagner (2017) by almost a factor of $\log \Phi$.

Our results In this paper we show that it is possible to overcome the limitations of the previous results and design a compact representation that supports $(1+\epsilon)$-approximate nearest neighbor search, with a space bound essentially matching that of Indyk and Wagner (2017). This constitutes the first reduction in the space complexity of approximate nearest neighbor below the "Johnson-Lindenstrauss bound". Specifically, we show the following. Suppose that we want the data structure to answer q approximate nearest neighbor queries in a d-dimensional dataset of size n, in which coordinates are represented by $\log \Phi$ bits each. All q queries must be answered correctly with probability $1 - \delta$. (See Section 2 for the formal problem definition).

Theorem 1.1 *For the all-nearest-neighbors problem, there is a sketch of size*

$$O\left(n\left(\frac{\log n \cdot \log(1/\epsilon)}{\epsilon^2} + \log\log\Phi + \log\left(\frac{q}{\delta}\right)\right) + d\log\Phi + \log\left(\frac{q}{\delta}\right)\log\left(\frac{\log(q/\delta)}{\epsilon}\right)\right) \textit{ bits.}$$

1. We note, however, that a simplified version of this method, described in Indyk et al. (2017), was shown to have good empirical performance for nearest neighbor search.

The proof is given in Section 4. We also give a lower bound of $\Omega(n \log(n)/\epsilon^2)$ for $q = 1$ and $\delta = 1/n^{O(1)}$ (Section B), which shows that the first term in the above theorem is almost tight.

Interestingly, the representation by itself does not return the (approximate) *distance* between the query point and the returned neighbor. Thus, we also consider the problem of estimating distances from a query point to all data points. In this setting, a result of Molinaro et al. (2013) shows that the Johnson-Lindenstrauss space bound is optimal when the number of queries is *equal* to the number of data points. However, in many settings, the number of queries is often substantially smaller than the dataset size. We give nearly tight upper and lower bounds (up to a factor of $\log(1/\epsilon)$) for this problem, showing it is possible to smoothly interpolate between Indyk and Wagner (2017), which does not support out-of-sample distance queries, and the Johnson-Lindenstrauss bound.

Specifically, we show the following. Suppose that we want the data structure to estimate all cross-distances between a set of q queries and all points in X, all of which must be estimated correctly with probability $1 - \delta$ (see Section 2 for the formal problem definition).

Theorem 1.2 *For the all-cross-distances problem, there is a sketch of size*

$$O\left(\frac{n}{\epsilon^2}\left(\log n \cdot \log(1/\epsilon) + \log(d\Phi)\log\left(\frac{q}{\delta}\right)\right) + \mathrm{poly}(d, \log\Phi, \log(q/\delta), \log(1/\epsilon))\right) \text{ bits.}$$

Note that the dependence per point on Φ is logarithmic, as opposed to doubly logarithmic in Theorem 1.1. We show this dependence is necessary, as per the following theorem.

Theorem 1.3 *Suppose that $d^{1-\rho} \geq \epsilon^{-2}\log(nq/\delta)$, $\Phi \geq 1/\epsilon$, and $1/n^{0.5-\rho'} \leq \epsilon \leq \epsilon_0$ for some constants $\rho, \rho' > 0$ and a sufficiently small constant ϵ_0. Then, for the all-cross-distances problem, any sketch must use at least*

$$\Omega\left(\frac{n}{\epsilon^2}\left(\log n + \log(d\Phi)\log\left(\frac{q}{\delta}\right)\right)\right) \text{ bits.}$$

The proofs are given in Section 5 and Appendix C, respectively.

Practical variant Indyk et al. (2017) presented a simplified version of Indyk and Wagner (2017), which has slightly weaker size guarantees, but on the other hand is practical to implement and was shown to work well empirically. However, it did not provably support out-of-sample queries. Our techniques in this paper can be adapted to their algorithm and endow it with such provable guarantees, while retaining its simplicity and practicality. We elaborate on this in Appendix D.

Our techniques The starting point of our representation is the compressed tree data structure from Indyk and Wagner (2017). The structure is obtained by constructing a hierarchical clustering of the data set, forming a tree of clusters. The position of each point corresponding to a node in the tree is then represented by storing a (quantized) displacement vector between the point and its "ancestor" in the tree. The resulting tree is further compressed by identifying and post-processing "long" paths in the tree. The intuition is that a subtree at the bottom of such a path corresponds to a cluster of points that is "sufficiently separated" from the rest of the points (see Figure 1). This means that the data structure does not need to know the exact position of this cluster in order to estimate the distances between the points in the cluster and the rest of the data set. Thus the data structure replaces each long path by a quantized displacement vector, where the quantization error

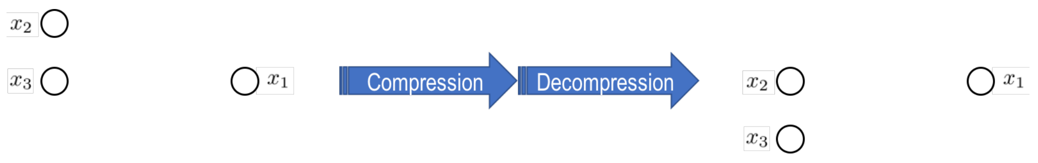

Figure 1: Compression and decompression of a two-dimensional dataset. The location of the well-separated cluster $\{x_2, x_3\}$ can be perturbed by the lossy compression algorithm, without significantly changing the distances to x_1.

does not depend on the length of the path. This ensures that the tree does not have long paths, which bounds its total size.

Unfortunately, this reasoning breaks down if one of the points is not known in advance, as it is the case for the approximate nearest neighbor problem. In particular, if the query point y lies in the vicinity of the separated cluster, then small perturbations to the cluster position can dramatically affect which points in the cluster are closest to y (see Figure 2 for an illustration).

In this paper we overcome this issue by maintaining extra information about the geometry of the point set. First, for each long path, we store not only the quantized displacement vector (which preserves the "global" position of the subtree with respect to the rest of the tree) but also the *suffix* of the path. Intuitively, this allows us to recover both the *most* significant bits and the *least* significant bits of points in the subtree corresponding to the "separated" clusters, which allows us to avoid cases as depicted in Figure 2. However, this intuition breaks down when the diameter of the cluster is much larger than the amount of "separation". Thus we also need to store extra information about the position of the subtree points. This is accomplished by storing a hashed representation of a representative point of the subtree (called "the center"). We note that this modification makes our data structure inherently randomized; in contrast, the data structure of Indyk and Wagner (2017) was deterministic.

Given the above information, the approximate nearest neighbor search is performed top down, as follows. In each step, we recover and enumerate points in the current subtree, some of which could be centers of "separated" clusters as described above. The "correct" center, guaranteed to contain an approximate nearest neighbor of the query point, is identified by its hashed value (if no hash match is found, then any center is equally good). Note that our data structure does not allow us to compute all distances from the query point y to all points in X (in fact, as mentioned earlier, this task is not possible to achieve within the desired space bound). Instead, it stores just enough information to ensure that the procedure never selects a "wrong" subtree to iterate on.

Lastly, suppose we also wish to estimate all distances from y to X. To this end, we augment each subtree with the distance sketches due to Kushilevitz et al. (2000) and Johnson and Lindenstrauss (1984). The former allows us to identify the cluster of *all* approximate nearest neighbors of y (whereas the above algorithm was only guaranteed to return *one* approximate nearest neighbor). The latter stores the approximate distance from that cluster. These are the smallest distances from y to X, which are the most challenging to estimate; the remaining distances can be estimated based on the hierarchical partition into well-separated clusters, which is already present in the sketch.

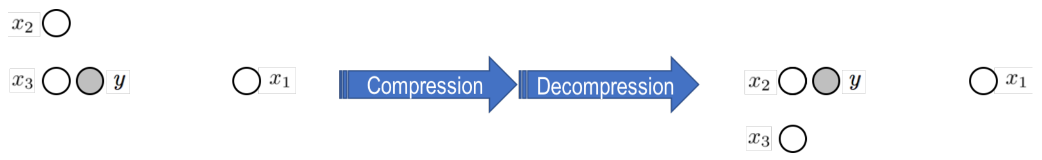

Figure 2: Compression and decompression of a dataset x_1, x_2, x_3 in the presence of a new query point y, which is unknown during compression. The same small perturbation in the location of $\{x_2, x_3\}$ as in Figure 1 fails to preserve x_3 as the nearest neighbor of y.

2. Formal Problem Statements

We formalize the problems in terms of one-way communication complexity. The setting is as follows. Alice has n data points, $X = \{x_1, \ldots, x_n\} \subset \{-\Phi \ldots \Phi\}^d$, while Bob has q query points, $Y = \{y_1, \ldots, y_q\} \subset \{-\Phi \ldots \Phi\}^d$, where $1 \le q \le n$. Distances are Euclidean, and we can assume w.l.o.g. that $d \le n$.[2] Let $\epsilon, \delta \in (0,1)$ be given parameters. In the one-way communication model, Alice computes a compact representation (called a *sketch*) of her data points and sends it to Bob, who then needs to report the output. We define two problems in this model (with private randomness), each parameterized by $n, q, d, \Phi, \epsilon, \delta$:[3]

Problem 1 – All-nearest-neighbors: Bob needs to report a $(1+\epsilon)$-approximate nearest neighbor in X for all his points simultaneously, with probability $1-\delta$. That is, for every $j \in [q]$, Bob reports an index $i_j \in [n]$ such that

$$\Pr\left[\forall j \in [q],\ \ \|y_j - x_{i_j}\| \le (1+\epsilon) \min_{i\in[n]} \|y_j - x_i\|\right] \ge 1-\delta.$$

Our upper bound for this problem is stated in Theorem 1.1.

Problem 2 – All-cross-distancess: Bob needs to estimate all distances $\|x_i - y_j\|$ up to distortion $(1 \pm \epsilon)$ simultaneously, with probability $1-\delta$. That is, for every $i \in [n]$ and $j \in [q]$, Bob reports an estimate $E(i,j)$ such that

$$\Pr\left[\forall i \in [n], j \in [q],\ \ (1-\epsilon)\|x_i - y_j\| \le E(i,j) \le (1+\epsilon)\|x_i - y_j\|\right] \ge 1-\delta.$$

Our upper and lower bounds for this problem are stated in Theorems 1.2 and 1.3.

3. Basic Sketch

In this section we describe the basic data structure (generated by Alice) used for all of our results. The data structure augments the representation from Indyk and Wagner (2017), which we will now reproduce. For the sake of readability, the notions from the latter paper (tree construction via hierarchical clustering, centers, ingresses and surrogates) are interleaved with the new ideas introduced

2. Any N-point Euclidean metric can be embedded into $N-1$ dimensions.

3. Throughout we use $[m]$ to denote $\{1, \ldots, m\}$, for an integer $m > 0$.

in this paper (top-out compression, grid quantization and surrogate hashing). Proofs in this section are deferred to Appendix A.

3.1. Hierarchical Clustering Tree

The sketch consists of an annotated hierarchical clustering tree, which we now describe with our modified "top-out compression" step.

Tree construction We construct the inter-link hierarchical clustering tree of X: In the bottom level (numbered 0) every point is a singleton cluster, and level $\ell > 0$ is formed from level $\ell - 1$ by recursively merging any two clusters whose distance is at most 2^ℓ, until no two such clusters are present. We repeat this until level $\lceil \log(2\sqrt{d}\Phi) \rceil$, even if all points in X are already joined in one cluster at a lower level. The following observation is immediate.

Lemma 3.1 *If $x, x' \in X$ are in different clusters at level ℓ, then $\|x - x'\| \geq 2^\ell$.*

Notation Let T^* denote the tree. For every tree node v, we denote its level by $\ell(v)$, its associated cluster by $C(v) \subset X$, and its cluster diameter by $\Delta(v)$. For a point $x_i \in X$, let $\mathrm{leaf}(x_i)$ denote the tree leaf whose associated cluster is $\{x_i\}$.

Top-out compression The *degree* of a node in T^* is its number of children. A 1*-path with k edges* in T^* is a downward path $u_0, u_1, \ldots, u_k$, such that (i) each of the nodes $u_0, \ldots, u_{k-1}$ has degree 1, (ii) u_k has degree either 0 or more than 1, (iii) if u_0 is not the root of T^*, then its ancestor has degree more than 1.

For every node v denote $\Lambda(v) := \log(\Delta(v)/(2^{\ell(v)}\epsilon))$. If v is the bottom of a 1-path with more than $\Lambda(v)$ edges, we replace all but the bottom $\Lambda(v)$ edges with a *long edge*, and annotate it by the length of the path it represents. More precisely, if the downward 1-path is $u_0, \ldots, u_k = v$ and $k > \Lambda(v)$, then we connect u_0 directly to $u_{k-\Lambda(v)}$ by the long edge, and the nodes $u_1, \ldots, u_{k-\Lambda(v)-1}$ are removed from the tree, and the long edge is annotated with length $k - \Lambda(v)$.

Lemma 3.2 *The compressed tree has $O(n \log(1/\epsilon))$ nodes.*

We henceforth refer only to the compressed tree, and denote it by T. However, for every node v in T, $\ell(v)$ continues to denote its level before compression (i.e., the level where the long edges are counted according to their lengths). We partition T into *subtrees* by removing the long edges. Let $\mathcal{F}(T)$ denote the set of subtrees.

Lemma 3.3 *Let v be the bottom node of a long edge, and $x, x' \in C(v)$. Then $\|x - x'\| \leq 2^{\ell(v)}\epsilon$.*

Lemma 3.4 *Let u be a leaf of a subtree in $\mathcal{F}(T)$, and $x, x' \in C(u)$. Then $\|x - x'\| \leq 2^{\ell(u)}\epsilon$.*

3.2. Surrogates

The purpose of annotating the tree is to be able to recover a list of *surrogates* for every point in X. A surrogate is a point whose location approximates x. Since we will need to compare x to a new query point, which is unknown during sketching, we define the surrogates to encompass a certain amount information about the absolute point location, by hashing a coarsened grid quantization of a representative point in each subtree.

Centers With every tree node v we associate an index $c(v) \in [n]$ such that $x_{c(v)} \in C(v)$, and we call $x_{c(v)}$ the *center* of $C(v)$. The centers are chosen bottom-up in T as follows. For a leaf v, $C(v)$ contains a single point $x_i \in X$, and we set $c(v) = i$. For a non-leaf v with children $u_1, \dots, u_k$, we set $c(v) = \min\{c(u_i) : i \in [k]\}$.

Ingresses Fix a subtree $T' \in \mathcal{F}(T)$. To every node u in T', except the root, we will now assign an *ingress* node, denoted $\mathrm{in(u)}$. Intuitively this is a node in the same subtree whose center is close to u, and the purpose is to store the location of u by its quantized displacement from that center (whose location will have been already stored, by induction).

We will now assign ingresses to all children of a given node v. (Doing this for every v in T' defines ingresses for all nodes in T' except its root.) Let $u_1, \dots, u_k$ be the children of v, and w.l.o.g. $c(v) = c(u_1)$. Consider the graph H_v whose nodes are $u_1, \dots, u_k$, and u_i, u_j are neighbors if there are points $x \in C(u_i)$ and $x' \in C(u_j)$ such that $\|x - x'\| \le 2^{\ell(v)}$. By the tree construction, H_v is connected. We fix an arbitrary spanning tree $\tau(v)$ of H_v which is rooted at u_1.

For u_1 we set $\mathrm{in}(u_1) := v$. For u_i with $i > 1$, let u_j be its (unique) direct ancestor in the tree $\tau(v)$. Let $x \in C(u_j)$ be the closest point to $C(u_i)$ in $C(u_j)$. Note that in T there is a downward path from u_j to $\mathrm{leaf}(x)$. Let u_x be the bottom node in that path that belongs to T'. (Equivalently, u_x is the bottom node on that downward path that is reachable from u without traversing a long edge.) We set $\mathrm{in}(u_i) := u_x$.

Grid net quantization Assume w.l.o.g. that Φ is a power of 2. We define a hierarchy of grids aligned with $\{-\Phi \dots \Phi\}^d$ as follows. We begin with the single hypercube whose corners are $(\pm\Phi, \dots, \pm\Phi)^d$. We generate the next grid by halving along each dimension, and so on. For every $\gamma > 0$, let $\mathcal{N}_\gamma$ be the coarsest grid generated, whose cell side is at most $\gamma/\sqrt{d}$. Note that every cell in $\mathcal{N}_\gamma$ has diameter at most γ. For a point $x \in \mathbb{R}^d$, we denote by $\mathcal{N}_\gamma[x]$ the closest corner of the grid cell containing it.

We will rely on the following fact about the intersection size of a grid and a ball; see, for example, Har-Peled et al. (2012).

Claim 3.5 *For every $\gamma > 0$, the number of points in $\mathcal{N}_\gamma$ at distance at most 2γ from any given point, is at most $O(1)^d$.*

Surrogates Fix a subtree $T' \in \mathcal{F}(T)$. With every node v in T' we will now associate a *surrogate* $s^*(v) \in \mathbb{R}^d$. Define the following for every node v in T':

$$\gamma(v) = \begin{cases} \left(5 + \lceil \frac{\Delta(v)}{2^{\ell(v)}} \rceil\right)^{-1} \cdot \epsilon & \text{if } v \text{ is a leaf in } T', \\ \left(5 + \lceil \frac{\Delta(v)}{2^{\ell(v)}} \rceil\right)^{-1} & \text{otherwise.} \end{cases}$$

The surrogates are defined by induction on the ingresses.

Induction base: For the root v of T' we set $s^*(v) := \mathcal{N}_{2^{\ell(v)}}[x_{c(v)}]$.

Induction step: For a non-root v we denote the quantized displacement of $c(v)$ from its ingress by $\eta(v) = \mathcal{N}_{\gamma(v)}\left[\frac{\gamma(v)}{2^{\ell(v)}}(x_{c(v)} - s^*(in(v)))\right]$, and set $s^*(v) := s^*(in(v)) + \frac{2^{\ell(v)}}{\gamma(v)} \cdot \eta(v)$.

Lemma 3.6 *For every node v, $\|x_{c(v)} - s^*(v)\| \le 2^{\ell(v)}$. Furthermore if v is a leaf of a subtree in $\mathcal{F}(T)$, then $\|x_{c(v)} - s^*(v)\| \le 2^{\ell(v)}\epsilon$.*

Hash functions For every level ℓ in the tree, we pick a hash function $H_\ell : \mathcal{N}_{2^\ell} \to [m]$, from a universal family (Carter and Wegman (1979)), where $m = O(1)^d \cdot \log(2\sqrt{d}\Phi) \cdot q/\delta$. The $O(1)$ term is the same constant from Claim 3.5 above. For every subtree root v, we store its hashed surrogate $H_{\ell(v)}(\mathcal{N}_{2^{\ell(v)}}[x_{c(v)}])$. We also store the description of each hash function H_ℓ for every level ℓ.

3.3. Sketch Size

The sketch contains the tree T, with each node v annotated by its center $c(v)$, ingress $\mathrm{in(u)}$, precision $\gamma(v)$ and quantized displacement $\eta(v)$ (if applicable). For subtree roots we store their hashed surrogate, and for long edges we store their length. We also store the hash functions $\{H_\ell\}$.

Lemma 3.7 *The total sketch size is*

$$O\left(n\left((d+\log n)\log(1/\epsilon) + \log\log\Phi + \log\frac{q}{\delta}\right) + d\log\Phi\right) \quad \textit{bits.}$$

As a preprocessing step, Alice can reduce the dimension of her points to $O(\epsilon^{-2}\log(qn/\delta))$ by a Johnson-Lindenstrauss projection. She then augments the sketch with the projection, in order for Bob to be able to project his points as well. By Kane et al. (2011), the projection can be stored with $O(\log d + \log(q/\delta) \cdot \log\log((q/\delta)/\epsilon))$ bits. This yields the sketch size stated in Theorem 1.1.

Remark Both the hash functions and the projection map can be sampled using public randomness. If one is only interested in the communication complexity, one can use the general reduction from public to private randomness due to Newman (1991), which replaces the public coins by augmenting $O(\log(nd\Phi))$ bits to the sketch (since Alice's input has size $O(nd\Phi)$ bits). The bound in Theorem 1.1 then improves to $O\left(n\left(\frac{\log n \cdot \log(1/\epsilon)}{\epsilon^2} + \log\log\Phi + \log\left(\frac{q}{\delta}\right)\right) + \log\Phi\right)$ bits, and the bound in Theorem 1.2 improves to $O\left(\frac{n}{\epsilon^2}\left(\log n \cdot \log(1/\epsilon) + \log(d\Phi)\log\left(\frac{q}{\delta}\right)\right)\right)$ bits. However, that reduction is non-constructive; we state our bounds so as to describe explicit sketches.

4. Approximate Nearest Neighbor Search

We now describe our approximate nearest neighbor search query procedure, and prove Theorem 1.1. Suppose Bob wants to report a $(1+\epsilon)$-approximate nearest neighbor in X for a point $y \in Y$.

Algorithm Report Nearest Neighbor:

1. Start at the subtree $T' \in \mathcal{F}(T)$ that contains the root of T.

2. Recover all surrogates $\{s^*(v) : v \in T'\}$, by the subroutine below.

3. Let v be the leaf of T' that minimizes $\|y - s^*(v)\|$.

4. If v is the head of a long edge, recurse on the subtree under that long edge. Otherwise v is a leaf in T, and in that case return $c(v)$.

Subroutine Recover Surrogates: This is a subroutine that attempts to recover all surrogates $\{s^*(v) : v \in T'\}$ in a given subtree $T' \in \mathcal{F}(T)$, using both Alice's sketch and Bob's point y.

Observe that to this end, the only information missing from the sketch is the root surrogate $s^*(r)$, which served as the induction base for defining the rest of the surrogates. The induction steps are fully defined by $\ell(v)$, $\mathrm{in}(v)$, $\gamma(v)$, and $\eta(v)$, which are stored in the sketch for every node $v \neq r$ in the subtree. The missing root surrogate was defined as $s^*(r) = \mathcal{N}_{2^{\ell(r)}}[x_{c(r)}]$. Instead, the sketch stores its hashed value $H_{\ell(r)}(\mathcal{N}_{2^{\ell(r)}}[x_{c(r)}])$ and the hash function $H_{\ell(r)}$.[4]

The subroutine attempts to reverse the hash. It enumerates over all points $p \in \mathcal{N}_{2^{\ell(r)}}$ such that $\|p - y\| \leq 2 \cdot 2^{\ell(r)}$. For each p it computes $H_{\ell(r)}(p)$. If $H_{\ell(r)}(x_{c(r)}) = H_{\ell(r)}(p)$ then it sets $s^*(r) = p$ and recovers all surrogates accordingly. If either no p, or more than one p, satisfy $H_{\ell(r)}(x_{c(r)}) = H_{\ell(r)}(p)$, then it proceeds with $s^*(r)$ set to an arbitrary point (say, the origin in $\mathbb{R}^d$).

Analysis. Let $r_0, r_1, \ldots$ be the roots of the subtrees traversed on the algorithm. Note that they reside on a downward path in T.

Claim 4.1 $\|x_{c(r_0)} - y\| \leq 2^{\ell(r_0)}$.

Proof Since $X \cup Y \subset \{-\Phi \ldots \Phi\}^d$, we have $\|x_{c(r_0)} - y\| \leq 2\sqrt{d}\Phi \leq 2^{\lceil \log(2\sqrt{d}\Phi) \rceil} = 2^{\ell(r_0)}$. ■

Let t be the smallest such that r_t satisfies $\|x_{c(r_t)} - y\| > 2^{\ell(r_t)}$. (The algorithm does not identify t, but we will use it for the analysis.)

Lemma 4.2 *With probability $1 - \delta/q$, for every $i = 0, \ldots, t-1$ simultaneously, the subroutine recovers $s^*(r_i)$ correctly as $\mathcal{N}_{2^{\ell(r)}}[x_{c(r)}]$. (Consequently, all surrogates in the subtree rooted by r_i are also recovered correctly.)*

Proof Fix a subtree $T' \in \mathcal{F}(T)$ rooted in r, that satisfies $\|y - x_{c(r)}\| \leq 2^{\ell(r)}$. Since $\|x_{c(r)} - s^*(r)\| \leq 2^{\ell(r)}$ (by Lemma 3.6), we have $\|y - s^*(r)\| \leq 2 \cdot 2^{\ell(r)}$. Hence the surrogate recovery subroutine tries $s^*(r)$ as one of the hash pre-image candidates, and will identify that $H_{\ell(r)}(s^*(r))$ matches the hash stored in the sketch. Furthermore, by Claim 3.5, the number of candidates is at most $O(1)^d$. Since the range of $H_{\ell(r)}$ has size $m = O(1)^d \cdot \log(2\sqrt{d}\Phi) \cdot q/\delta$, then with probability $1 - \delta/(q \log(2\sqrt{d}\Phi))$ there are no collisions, and $s^*(r)$ is recovered correctly. The lemma follows by taking a union bound over the first t subtrees traversed by the algorithm, i.e. those rooted by r_i for $i = 0, 1, \ldots, t-1$. Noting that t is upper-bounded by the number of levels in the tree, $\log(2\sqrt{d}\Phi)$, we get that all the $s^*(r_i)$'s are recovered correctly simultaneously with probability $1 - \delta/q$. ■

From now on we assume that the event in Lemma 4.2 succeeds, meaning in steps $0, 1, \ldots, t-1$, the algorithm recovers all surrogates correctly. We henceforth prove that under this event, the algorithm returns a $(1+\epsilon)$-approximate nearest neighbor of y. In what follows, let $x^* \in X$ be a fixed true nearest neighbor of y in X.

Lemma 4.3 *Let $T' \in \mathcal{F}(T)$ be a subtree rooted in r, such that $x^* \in C(r)$. Let v a leaf of T' that minimizes $\|y - s^*(v)\|$. Then either $x^* \in C(v)$, or every $z \in C(v)$ is a $(1 + O(\epsilon))$-approximate nearest neighbor of y.*

4. Note that fully storing the root surrogates is prohibitive: $\mathcal{N}_{2^{\ell(r)}}$ has $\Theta(2\sqrt{d}\Phi/2^{\ell(r)})^d$ cells, hence storing a cell ID takes $\Omega(d \log d)$ bits, and since there can be $\Omega(n)$ subtree roots, this would bring the total sketch size to $\Omega(nd \log d)$.

Proof Suppose w.l.o.g. by scaling that $\epsilon < 1/6$. If $x^* \in C(v)$ then we are done. Assume now that $x^* \in C(u)$ for a leaf $u \neq v$ of T'. Let $\ell := \max\{\ell(v), \ell(u)\}$. We start by showing that $\|y - x^*\| > \frac{1}{4} \cdot 2^\ell$. Assume by contradiction this is not the case. Since u is a subtree leaf and $x^* \in C(u)$, we have $\|x^* - x_{c(u)}\| \leq 2^\ell\epsilon$ by Lemma 3.4. We also have $\|x_{c(u)} - s^*(u)\| \leq 2^\ell\epsilon$ by Lemma 3.6. Together, $\|y - s^*(u)\| \leq (\frac{1}{4} + 2\epsilon)2^\ell$. On the other hand, by the triangle inequality, $\|y - s^*(v)\| \geq \|x^* - x_{c(v)}\| - \|y - x^*\| - \|x_{c(v)} - s^*(v)\|$. Noting that $\|x^* - x_{c(v)}\| \geq 2^\ell$ (by Lemma 3.1, since x^* and $x_{c(v)}$ are separated at level ℓ), $\|y - x^*\| \leq \frac{1}{4} \cdot 2^\ell$ (by the contradiction hypothesis) and $\|x_{c(v)} - s^*(v)\| \leq 2^\ell\epsilon$ (by Lemma 3.6), we get $\|y - s^*(v)\| \geq (\frac{3}{4} - \epsilon)2^\ell > (\frac{1}{4} + 2\epsilon)2^\ell \geq \|y - s^*(u)\|$. This contradicts the choice of v.

The lemma now follows because for every $z \in C(v)$,

$$\|y - z\| \leq \|y - s^*(v)\| + \|s^*(v) - x_{c(v)}\| + \|x_{c(v)} - z\| \tag{1}$$

$$\leq \|y - s^*(u)\| + \|s^*(v) - x_{c(v)}\| + \|x_{c(v)} - z\| \tag{2}$$

$$\leq \|y - x^*\| + \|x^* - x_{c(u)}\| + \|x_{c(u)} - s^*(u)\| + \|s^*(v) - x_{c(v)}\| + \|x_{c(v)} - z\| \tag{3}$$

$$\leq \|y - x^*\| + 4 \cdot 2^\ell\epsilon \tag{4}$$

$$\leq (1 + 16\epsilon)\|y - x^*\|, \tag{5}$$

where (1) and (3) are by the triangle inequality, (2) is since $\|y - s^*(v)\| \leq \|y - s^*(u)\|$ by choice of v, (4) is by Lemmas 3.4 and 3.6, and (5) is since we have shown that $\|y - x^*\| > \frac{1}{4} \cdot 2^\ell$. Therefore z is a $(1 + 16\epsilon)$-approximate nearest neighbor of y. ■

Proof of Theorem 1.1. We may assume w.l.o.g. that ϵ is smaller than a sufficiently small constant. Suppose that the event in Lemma 4.2 holds, hence all surrogates in the subtrees rooted by $r_0, r_1, \ldots, r_{t-1}$ are recovered correctly. We consider two cases. In the first case, $x^* \notin C(r_t)$. Let $i \in \{1, \ldots, t\}$ be the smallest such that $x^* \notin C(r_i)$. By applying Lemma 4.3 on r_{i-1}, we have that every point in $C(r_i)$ is a $(1 + O(\epsilon))$-approximate nearest neighbor of y. After reaching r_i, the algorithm would return the center of some leaf reachable from r_i, and it would be a correct output.

In the second case, $x^* \in C(r_t)$. We will show that every point in $C(r_t)$ is a $(1 + O(\epsilon))$-approximate nearest neighbor of y, so once again, once the algorithm arrives at r_t it can return anything. By Lemma 3.3, every $x \in C(r_t)$ satisfies

$$\|x - x^*\| \leq 2^{\ell(r_t)}\epsilon. \tag{6}$$

In particular, $\|x_{c(r_t)} - x^*\| \leq 2^{\ell(r_t)}\epsilon$. By definition of t we have $\|x_{c(r_t)} - y\| > 2^{\ell(r_t)}$. Combining the two yields $\|y - x^*\| \geq \|y - x_{c(r_t)}\| - \|x_{c(r_t)} - x^*\| > (1 - \epsilon)2^{\ell(r_t)}$. Combining this with eq. (6), we find that every $x \in C(r_t)$ satisfies $\|x - x^*\| \leq \frac{\epsilon}{1-\epsilon}\|y - x^*\|$, and hence $\|y - x\| \leq (1 + 2\epsilon)\|y - x^*\|$ (for $\epsilon \leq 1/2$). Hence x is a $(1 + 2\epsilon)$-nearest neighbor of y.

The proof assumes the event in Lemma 4.2, which occurs with probability $1 - \delta/q$. By a union bound, the simultaneous success probability of the q query points of Bob is $1 - \delta$ as required. ■

5. Distance Estimation

We now prove theorem 1.2. To this end, we augment the basic sketch from Section 3 with additional information, relying on the following distance sketches due to Achlioptas (2001) (following Johnson and Lindenstrauss (1984)) and Kushilevitz et al. (2000).

Lemma 5.1 (**Achlioptas** (**2001**)) *Let $\epsilon, \delta' > 0$. Let $d' = c\epsilon^{-2}\log(1/\delta')$ for a sufficiently large constant $c > 0$. Let M be a random $d' \times d$ matrix in which every entry is chosen independently uniformly at random from $\{-1/\sqrt{d'}, 1/\sqrt{d'}\}$. Then for every $x, y \in \mathbb{R}^d$, with probability $1 - \delta'$, $\|Mx - My\| = (1 \pm \epsilon)\|x - y\|$.*

Lemma 5.2 (**Kushilevitz et al.** (**2000**)) *Let $R > 0$ be fixed and let $\epsilon, \delta' > 0$. There is a randomized map sk_R of vectors in $\mathbb{R}^d$ into $O(\epsilon^{-2}\log(1/\delta'))$ bits, with the following guarantee. For every $x, y \in \mathbb{R}^d$, given $\mathrm{sk}_R(x)$ and $\mathrm{sk}_R(y)$, one can output the following with probability $1 - \delta'$:*

- *If $R \leq \|x - y\| \leq 2R$, output a $(1 + \epsilon)$-estimate of $\|x - y\|$.*
- *If $\|x - y\| \leq (1 - \epsilon)R$, output "Small".*
- *If $\|x - y\| \geq (1 + \epsilon)R$, output "Large".*

We augment the basic sketch from Section 3 as follows. We sample a matrix M from Lemma 5.1, with $\delta' = \delta/q$. In addition, for every level ℓ in the tree T, we sample a map sk_{2^ℓ} from Lemma 5.2, with $\delta' = \delta/(q\log(2\sqrt{d}\Phi))$. For every subtree root r in T, we store $Mx_{c(r)}$ and $\mathrm{sk}_{2^{\ell(r)}}(x_{c(r)})$ in the sketch. Let us calculate the added size to the sketch:

- Since $x_{c(r)}$ has d coordinates of magnitude $O(\Phi)$ each, $Mx_{c(r)}$ has d' coordinates of magnitude $O(d\Phi)$ each. Since there are $O(n)$ subtree roots (cf. Lemma 3.7), storing $Mx_{c(r)}$ for every r adds $O(nd'd\Phi) = O(\epsilon^{-2}n\log(q/\delta)\log(d\Phi))$ bits to the sketch. In addition we store the matrix M, which takes $O(d'd)$ bits to store, which is dominated by the previous term.
- By Lemma 5.2, each $\mathrm{sk}_{2^{\ell(r)}}(x_{c(r)})$ adds $O(\epsilon^{-2}\log(q\log(2\sqrt{d}\Phi)/\delta))$ bits to the sketch, and as above there are $O(n)$ of these. In addition we store the map $\mathrm{sk}_{2^{\ell(r)}}$ for every ℓ. Each map takes $\mathrm{poly}(d, \log\Phi, \log(q/\delta), 1/\epsilon)$ bits to store.

In total, we get the sketch size stated in Theorem 1.2. Next we show how to compute all distances from a new query point y.

Query algorithm. Given the sketch, an index $k \in [n]$ of a point in X, and a new query point y, the algorithm needs to estimate $\|y - x_k\|$ up to $1 \pm O(\epsilon)$ distortion. It proceeds as follows.

1. Perform the approximate nearest neighbor query algorithm from Section 4. Let $r_0, r_1, \ldots$ be the downward sequence of subtree roots traversed by it.
2. For each r_j, estimate from the sketch whether $\|y - x_{c(r_j)}\| \leq 2^{\ell(r_j)}$. This can be done by Lemma 5.2, since the sketch stores $\mathrm{sk}_{2^{\ell(r_j)}}(x_{c(r_j)})$ and also the map $\mathrm{sk}_{2^{\ell(r_j)}}$, with which we can compute $\mathrm{sk}_{2^{\ell(r_j)}}(y)$.
3. Let t be the smallest j that satisfies $\|y - x_{c(r_j)}\| > 2^{\ell(r_j)}$ according the estimates of Lemma 5.2. (This attempts to recover from the sketch the same t as defined in the analysis in Section 4.)
4. Let $t_k \in \{0, \ldots, t\}$ be the maximal such that $x_k \in C(r_{t_k})$.
   (In words, r_{t_k} is the root of the subtree in which x_k and y "part ways".)
5. If $t_k = t$, return $\|My - Mx_{c(r_t)}\|$. Note that M and $Mx_{c(r_t)}$ are stored in the sketch.
6. If $t_k < t$, let v_k be the bottom node on the downward path from r_{t_k} to $\mathrm{leaf}(x_k)$ that does not traverse a long edge. Return $\|y - s^*(v_k)\|$.

Analysis. Fix a query point y. Define the "good event" $\mathcal{A}(y)$ as the intersection of the following:

1. For every subtree root r_j traversed by the query algorithm above, the invocation of Lemma 5.2 on $\mathrm{sk}_{2^{\ell(r_j)}}(x_{c(r_j)})$ and $\mathrm{sk}_{2^{\ell(r_j)}}(y)$ succeeds in deciding whether $\|y-x_{c(r_j)}\| \le 2^{\ell(r_j)}$. Specifically, this ensures that $\|y-x_{c(r_j)}\| \le 2^{\ell(r_j)}$ for every $j<t$, and $\|y-x_{c(r_t)}\| \ge (1-\epsilon)2^{\ell(r_t)}$. Recalling that we invoked the lemma with $\delta' = \delta/(q\log(2\sqrt{d}\Phi))$, we can take a union bound and succeed in all levels simultaneously with probability $1-\delta/q$.
2. $\|My - Mx_{c(r_t)}\| = (1\pm\epsilon)\|y-x_{c(r_t)}\|$. By Lemma 5.1 this holds with probability $1-\delta/q$.

Altogether, $\mathcal{A}(y)$ occurs with probability $1-O(\delta/q)$.

Lemma 5.3 *Conditioned on $\mathcal{A}(y)$ occuring, with probabiliy $1-\delta/q$, Lemma 4.2 holds. Namely, the query algorithm correctly recovers all surrogates in the subtrees rooted by r_j for $j=0,1,\dots,t-1$.*

Proof The proof of Lemma 4.2 in Section 4 relied on having $\|y-x_{c(r_j)}\| \le 2^{\ell(r_j)}$ for every $j<t$. Conditioning on $\mathcal{A}(y)$ ensures this holds. ■

Proof of Theorem 1.2. Let $\mathcal{A}^*(y)$ denote the event in which both $\mathcal{A}(y)$ occurs and the conclusion of Lemma 4.2 occurs. By the above lemma, $\mathcal{A}^*(y)$ happens with probability $1-O(\delta/q)$. From now on we will assume that $\mathcal{A}^*(y)$ occurs, and conditioned on this, we will show that the distance from y to any data point can be deterministically estimated correctly. To this end, fix $k\in[n]$ and suppose our goal is to estimate $\|y-x_k\|$. Let t_k and v_k be as defined by the distance query algorithm above. We handle the two cases of the algorithm separately.

Case I: $t_k = t$. This means $x_k \in C(r_t)$. By Lemma 3.3 we have $\|x_k - x_{c(r_t)}\| \le 2^{\ell(r_t)}\epsilon$. By the occurance of $\mathcal{A}^*(y)$ we have $\|y-x_{c(r_t)}\| > (1-\epsilon)2^{\ell(r_t)}$. Together, $\|y-x_k\| = \|y-x_{c(r_t)}\| \pm \|x_k - x_{c(r_t)}\| = (1\pm 2\epsilon)\|y-x_{c(r_t)}\|$. This means that $\|y-x_{c(r_t)}\|$ is a good estimate for $\|y-x_k\|$. Since $\mathcal{A}^*(y)$ occurs, it holds that $\|My-Mx_{c(r_t)}\| = (1\pm\epsilon)\|y-x_{c(r_t)}\|$, hence $\|My-Mx_{c(r_t)}\|$ is also a good estimate for $\|y-x_k\|$, and this is what the algorithm returns.

Case II: $t_k < t$. Let T_{t_k} be the subtree rooted by r_{t_k}. By the occurance of $\mathcal{A}^*(y)$, all surrogates in T_{t_k} are recovered correctly, and in particular $s^*(v_k)$ is recovered correctly. By Lemma 3.6 we have $\|x_{c(v_k)} - s^*(v_k)\| \le 2^{\ell(v_k)}\epsilon$, and by Lemma 3.4 (noting that $x_k \in C(v_k)$ by choice of v_k) we have $\|x_k - x_{c(v_k)}\| \le 2^{\ell(v_k)}\epsilon$. Together, $\|x_k - s^*(v_k)\| \le 2\cdot 2^{\ell(v_k)}\epsilon$.

Let v be the leaf in T_{t_k} that minimizes $\|y-s^*(v)\|$ (over all leaves of T_{t_k}). Equivalently, v is the top node of the long edge whose bottom node is r_{t_k+1}. Let $\ell := \max\{\ell(v),\ell(v_k)\}$. By choice of t_k we have $v \ne v_k$, hence the centers of these two leaves are separated already at level ℓ, hence $\|x_{c(v_k)} - x_{c(v)}\| \ge 2^\ell$ by Lemma 3.1. By two applications of Lemma 3.6 we have $\|x_{c(v_k)} - s^*(v_k)\| \le 2^\ell\epsilon$ and $\|x_{c(v)} - s^*(v)\| \le 2^\ell\epsilon$. Together, $\|s^*(v_k) - s^*(v)\| \ge (1-2\epsilon)\cdot 2^\ell$. Since y is closer to $s^*(v)$ than to $s^*(v_k)$ (by choice of v), we have $\|y-s^*(v_k)\| \ge \frac{1}{2}\cdot\|s^*(v_k)-s^*(v)\| \ge \left(\frac{1}{2}-\epsilon\right)\cdot 2^\ell$. Combining this with $\|x_k - s^*(v_k)\| \le 2\cdot 2^{\ell(v_k)}\epsilon$, which was shown above, yields $\|x_k - s^*(v_k)\| \le \epsilon\cdot\frac{1}{1/2-\epsilon}\cdot\|y-s^*(v_k)\| = O(\epsilon)\cdot\|y-s^*(v_k)\|$. Therefore, $\|y-x_k\| = \|y-s^*(v_k)\| \pm \|x_k - s^*(v_k)\| = (1\pm O(\epsilon))\cdot\|y-s^*(v_k)\|$, which means $\|y-s^*(v_k)\|$ is a good estimate for $\|y-x_k\|$, and this is what the algorithm returns.

Conclusion: Combining both cases, we have shown that for any query point y, all distances from y to X can be estimated correctly with probability $1-O(\delta/q)$. Taking a union bound over q queries, and scaling δ and ϵ appropriately by a constant, yields the theorem. ■

Acknowledgments

This work was supported by grants from the MITEI-Shell program, Amazon Research Award and Simons Investigator Award.

References

Dimitris Achlioptas. Database-friendly random projections. In *Proceedings of the twentieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 274–281. ACM, 2001.

Alexandr Andoni and Piotr Indyk. Nearest neighbors in high-dimensional spaces. *CRC Handbook of Discrete and Computational Geometry*, 2017.

J Lawrence Carter and Mark N Wegman. Universal classes of hash functions. *Journal of computer and system sciences*, 18(2):143–154, 1979.

A. Efros. How to stop worrying and learn to love nearest neighbors. *https://nn2017.mit.edu/wp-content/uploads/sites/5/2017/12/Efros-NIPS-NN-17.pdf*, 2017.

Chirag Gupta, Arun Sai Suggala, Ankit Goyal, Harsha Vardhan Simhadri, Bhargavi Paranjape, Ashish Kumar, Saurabh Goyal, Raghavendra Udupa, Manik Varma, and Prateek Jain. Protonn: Compressed and accurate knn for resource-scarce devices. In *International Conference on Machine Learning*, pages 1331–1340, 2017.

Sariel Har-Peled, Piotr Indyk, and Rajeev Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory of computing*, 8(1):321–350, 2012.

Piotr Indyk and Tal Wagner. Near-optimal (euclidean) metric compression. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 710–723. SIAM, 2017.

Piotr Indyk, Ilya Razenshteyn, and Tal Wagner. Practical data-dependent metric compression with provable guarantees. In *Advances in Neural Information Processing Systems*, pages 2614–2623, 2017.

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(3): 26, 2013.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Faiss: A library for efficient similarity search. *https://code.facebook.com/posts/1373769912645926/faiss-a-library-for-efficient-similarity-search/*, 2017a.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *arXiv preprint arXiv:1702.08734*, 2017b.

William B Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary mathematics*, 26(189-206):1, 1984.

Daniel Kane, Raghu Meka, and Jelani Nelson. Almost optimal explicit johnson-lindenstrauss families. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 628–639. Springer, 2011.

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.

Marco Molinaro, David P Woodruff, and Grigory Yaroslavtsev. Beating the direct sum theorem in communication complexity with implications for sketching. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1738–1756. Society for Industrial and Applied Mathematics, 2013.

Ilan Newman. Private vs. common random bits in communication complexity. *Information processing letters*, 39(2):67–71, 1991.

Gregory Shakhnarovich, Trevor Darrell, and Piotr Indyk. *Nearest-neighbor methods in learning and vision: theory and practice (neural information processing)*. The MIT press, 2006.

Jingdong Wang, Ting Zhang, Nicu Sebe, Heng Tao Shen, et al. A survey on learning to hash. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):769–790, 2018.

Jun Wang, Wei Liu, Sanjiv Kumar, and Shih-Fu Chang. Learning to hash for indexing big data: a survey. *Proceedings of the IEEE*, 104(1):34–57, 2016.

Appendix A. Deferred Proofs from Section 3

Proof [Proof of Lemma 3.2] Charging the degree-1 nodes along every maximal 1-path to its bottom (non-degree-1) node, the total number of nodes after top-out compression is bounded by

$$\sum_{v:\deg(v)\neq 1} \Lambda(v).$$

Indyk and Wagner (2017) show this is at most $O(n\log(1/\epsilon))$. The difference is that their compression replaces summands larger than $\Lambda(v)$ by zero, while our (top-out) compression trims them to $\Lambda(v)$. ■

Proof [Proof of Lemma 3.3] By top-out compression, v is the top of a downward 1-path of length $\Lambda(v')$ whose bottom node is v'. Since no clusters are joined along a 1-path, we have $C(v') = C(v)$, hence $x, x' \in C(v')$ and hence $\|x - x'\| \leq \Delta(v')$. Noting that $\ell(v) = \ell(v') + \Lambda(v') = \ell(v') + \log(\Delta(v')/(2^{\ell(v')}\epsilon)) = \log(\Delta(v')/\epsilon)$ and rearranging, we find $\Delta(v') = 2^{\ell(v)}\epsilon$, which yields the claim. ■

Proof [Proof of Lemma 3.4] If u is a leaf in T then $C(u)$ is a singleton cluster, hence $x = x'$. Otherwise u is the top node of a long edge, and the claim follows by Lemma 3.3 on the bottom node of that long edge. ■

Proof [Proof of Lemma 3.6] The first part of the lemma (where v is any node, not necessarily a subtree leaf) is proved by induction on the ingresses. In the base case we use that $\|x_{c(v)} - s^*(v)\| \leq 2^{\ell(v)}$ by the choice of grid net. The induction step is identical to Indyk and Wagner (2017). The "furthermore" part of the lemma then follows as a corollary due to the refined definition of $\gamma(v)$ for a subtree leaf v, in the induction step leading to it. ■

Proof [Proof of Lemma 3.7] The sketch of Indyk and Wagner (2017) stores the compressed tree T', with each node annotated by its center $c(v)$, ingress $\text{in}(v)$, precision $\gamma(v)$ and quantized displacement $\eta(v)$. Every long edge is annotated by its length. They show this takes

$$O\left(n\left((d+\log n)\log(1/\epsilon) + \log\log\Phi\right)\right)$$

bits; note that by Lemma 3.2, top-out compression did not effect this bound.

We additionally store the hashed surrogates of subtree roots. There are $O(n)$ subtrees,[5] and each hash takes $\log m$ bits to store, which adds $O(n(d + \log\log\Phi + \log(q/\delta)))$ bits to the above. Finally, we store the hash functions H_ℓ for every ℓ. The domain of each H_ℓ is N_{2^ℓ}, which is a subset of $\{-\Phi\ldots\Phi\}^d$, and hence H_ℓ can be specified by $O(\log(\Phi^d))$ random bits (Carter and Wegman (1979)). Since we do not require independence between hash functions of different levels, we can use the same random bits for all hash functions, adding a total of $O(d\log\Phi)$ bits to the sketch. ■

5. By construction, the tree of subtrees in T has no degree-1 nodes. Since T has n leaves, there are at most $2n-1$ subtrees.

Appendix B. Approximate Nearest Neighbor Sketching Lower Bound

Theorem B.1 *Suppose that $d \geq \Omega(\epsilon^{-2} \log n)$, $\Phi \geq 1/\epsilon$, and $1/n^{0.5-\beta} \leq \epsilon \leq \epsilon_0$ for a constant $\beta > 0$ and a sufficiently small constant ϵ_0. Suppose also that $\delta < 1/n^2$. Then, for the all-nearest-neighbors problem, Alice must use a sketch of at least $\Omega(\beta\epsilon^{-2} n \log n)$ bits.*

Proof We start with dimension $d = n + 1 + \log n$; it can then be reduced by standard dimension reduction. Fix $k = 1/\epsilon^2$ and assume w.l.o.g. that k is a square integer (by taking ϵ to be appropriately small). Note that since $\epsilon > 1/\sqrt{n}$ we have $k \leq n$, and that since $\Phi \geq 1/\epsilon$ we have $\sqrt{k} \leq \Phi$.

The data set will consist of $2n$ points, $x_1, \dots, x_n$ and $z_1, \dots, z_n$. Let $i \in [n]$. We choose the first n coordinates of x_i to be an arbitrary k-sparse vector, in which each nonzero coordinate equals $1/\sqrt{k}$. Note that the norm of this part is 1. The $(n+1)$th coordinate of x_i is set to 0. The remaining $\log n$ coordinates encode the binary encoding of i, with each coordinate multiplied by 10.

Next we define z_i. The first n coordinates are 0. The $(n+1)$th coordinate equals $\sqrt{1-\epsilon}$. The remaining $\log n$ coordinates encode i similarly to x_i.

The number of different choices for $\{x_1, \dots, x_n\}$ is $\binom{n}{k}^n$. Therefore if we show that one can fully recover $x_1, \dots, x_n$ from a given all-nearest-neighbor sketch of the dataset, we would get the desired lower bound

$$\log\left(\binom{n}{k}^n\right) \geq nk\log(n/k) = \epsilon^{-2} n \log(\epsilon^2 n) = \epsilon^{-2} n \log(n^{2\beta}) = 2\beta\epsilon^{-2} n \log n.$$

Suppose we have such a sketch. For given $i, j \in [n]$ we now show how to recover the jth coordinate of x_i, denoted $x_i(j)$, with a single approximate nearest neighbor query. Let y_{ij} be the following vector in $\mathbb{R}^d$: The first $n+1$ coordinates are all zeros, except for the jth coordinate which is set to 1. The last $\log n$ coordinates encode i similary to x_i and z_i.

Consider the distances from y_{ij} to all data points. We start with x_i. It is identical to y_i in the last $\log n + 1$ coordinates, so we will restrict both to the first n coordinates and denote the restricted vectors by $x_i^{:n}$ and $y_{ij}^{:n}$. $x_i^{:n}$ is a k-sparse vector with nonzero entries equal to $1/\sqrt{k}$, hence $\|x_i^{:n}\| = 1$. $y_{ij}^{:n}$ is just the standard basis vector e_j in $\mathbb{R}^n$. Hence,

$$\|x_i - y_{ij}\|^2 = \|x_i^{:n} - y_{ij}^{:n}\|^2 = \|x_i^{:n}\|^2 + \|y_{ij}^{:n}\|^2 - 2(x_i^{:n})^\top y_{ij}^{:n} = 2 - 2x_i(j).$$

This equals 2 if $x_i(j) = 0$ and $2 - 2/\sqrt{k} = 2 - 2\epsilon$ if $x_i(j) = 1/\sqrt{k}$.

Next consider z_i. It is identical to y_{ij} in all except the jth coordinate, which is 0 in z_i and 1 in y_{ij}, and the $(n+1)$th coordinate, which is 0 for y_{ij} and $\sqrt{1-\epsilon}$ for z_i. Therefore, $\|z_i - y_{ij}\|^2 = 2 - \epsilon$.

Finally, for every $i' \neq i$, both $x_{i'}$ and $z_{i'}$ are at distance at least 10 from y_{ij} due to the encoding of i (as binary multiplied by 10) in the last $\log n$ coordinates.

In summation we have established the following:

- If $x_i(j) \neq 0$, then the closest point to y_{ij} in the dataset is x_i at distance $\sqrt{2-2\epsilon}$, and the next closest point is z_i at distance $\sqrt{2-\epsilon}$.
- If $x_i(j) = 0$, then the closest point to y_{ij} in the dataset is z_i at distance $\sqrt{2-\epsilon}$, and the next closest point is x_i at distance 2.

Therefore, if the sketch supports $(1 + \frac{1}{8}\epsilon)$-approximate nearest neighbors, we can recover the true nearest neighbor of y_{ij} and thus recover $x_i(j)$. By hypothesis, the query succeeds with probability

$\delta < 1/n^2$. By a union bound over all $i, j \in [n]$ we can recover all of $x_1, \dots, x_n$ simultaneously, and the theorem follows. ■

Appendix C. Lower Bound for Distance Estimation

In this section we prove Theorem 1.3. We handle the two terms in the lower bound separately.

C.1. First Lower Bound Term

Lemma C.1 *Suppose that $d \geq \Omega(\epsilon^{-2} \log n)$; $\Phi \geq 1/\epsilon$; ϵ is at most a sufficiently small constant; and $\epsilon \geq 1/n^{0.5-\rho'}$ for a constant $\rho' > 0$. Then, for the all-cross-distances problem, Alice must use a sketch of at least $\Omega(\rho' \epsilon^{-2} n \log n)$ bits.*

Proof Consider the following problem: Given a dataset $X \subset \{-\Phi \dots \Phi\}^d$ consisting of n points, we need to produce a sketch, from which we can recover (deterministically) all distances $\{\|x - x'\| : x, x' \in X\}$ up to distortion $1 \pm \epsilon$. This is the problem considered in Indyk and Wagner (2017), and they show a lower bound of $\Omega(\rho' \epsilon^{-2} n \log n)$ under the asssumptions of the current lemma. We will obtain the same lower bound by reducing this problem to all-cross-distances.

To this end, suppose we have a given sketching procedure for the all-cross-distances problem that uses $s = s(n, d, \Phi, 1, \epsilon, \delta)$ amortized bits per point. We invoke it on X and denote the resulting sketch by S_0. For every point $y \in \{-\Phi \dots \Phi\}^d$, with probabiliy $1 - \delta$, all distances $\{\|x - y\| : x \in X\}$ can be recovered from S_0. In particular, this holds in expectation for $(1 - \delta)n$ of the points in X. By Markov's inequality, this holds for $\frac{1}{2}(1 - \delta)n > \frac{1}{4}n$ of the points in X with probability at least $1/2$. We proceed by recursion on the remaining $\frac{3}{4}n$ points in X. The sketch produced in the ith step of the recursion is denoted by S_i and has total size $(\frac{3}{4})^i ns$ bits. After $t = O(\log n)$ steps, with nonzero probability $1/n^{O(1)}$, we have produced a sequence of sketches $S_0, \dots, S_t$ from which every distance in $\{\|x - x'\| : x, x' \in X\}$ can be recovered, with a total size of $O(\sum_{i=0}^{t} (\frac{3}{4})^i ns) = O(ns)$ bits. This yields the desired lower bound. ■

C.2. Second Lower Bound Term

Lemma C.2 *Suppose that $d^{1-\rho} \geq \epsilon^{-2} \log(q/\delta)$ for a constant $\rho > 0$, and ϵ is at most a sufficiently small constant. Then, for the all-cross-distances problem, Alice must use a sketch of at least $\Omega(\epsilon^{-2} n \log(d\Phi) \log(q/\delta))$ bits.*

The proof is by adapting the framework of Molinaro et al. (2013), who proved (among other results) this statement for the case $q = n$. We describe the adaption and refer to Molinaro et al. (2013) for missing details that remain similar.

C.2.1. Preliminaries

We follow the approach of Jayram and Woodruff (2013), of proving one-way communication lower bounds by reduction to variants of the augmented indexing problem, defined next.

Definition C.3 (Augmented Indexing) *In the Augmented Indexing problem $AugInd(k, \delta)$, Alice gets a vector A with k entries, whose elements are entries of a universe of size $20/\delta$. Bob gets an index $i \in [k]$, an element e, and the elements $A(i')$ for every $i' < i$. Bob needs to decide whether $e = A(i)$, and succeed with probability $1 - \delta$.*

Jayram and Woodruff (2013) give a one-way communication lower bound of $\Omega(k \log(1/\delta))$ for this problem. The main component in Molinaro et al. (2013) is a modified one-way communication model, in which the protocol is allowed to abort with a substantially larger (constant) probability than it is allowed to err. We will refer to it simply as the *abortion model* and refer to Molinaro et al. (2013) for the exact definition (which we will not require). They prove the same lower bound for Augmented Indexing.

Lemma C.4 (informal) *In the abortion model, the one-way communication complexity of $AugInd(k, \delta)$ is $\Omega(k \log(1/\delta))$.*

C.2.2. Variants of Augmented Indexing

We start by defining a variant of augmented indexing that will be suitable for out purpose.

Definition C.5 *In the Matrix Augmented Indexing problem $MatAugInd(k, m, \delta)$, Alice gets a matrix A of order $k \times m$, whose entries are elements of a universe of size $1/\delta$. Bob gets indices $i \in [k]$ and $j \in [m]$, an element e, and the elements $A(i, j')$ for every $j < j'$. Bob needs to decide whether $e = A(i, j)$, and succeed with probability $1 - \delta$.*

This problem is clearly at least as difficult as $AugInd(km, \delta)$ from Definition C.3, since in the latter Bob gets more information (namely, if we arrange the vector A in $AugInd(km, \delta)$ as a $k \times m$ matrix, then Bob gets all entries of A which lexicographically precede $A(i, j)$). We get the following immediate corollary from Lemma C.4.

Corollary C.6 *In the abortion model, the one-way communication complexity of $MatAugInd(k, m, \delta)$ is $\Omega(km \log(1/\delta))$.*

Molinaro et al. (2013) reformulate Augmented Indexing so that Alice's input is a set instead of vector. Similary, we reformulate Matrix Augmented Indexing as follows.

Definition C.7 *Let $m > 0$ and $k > 0$ be integers, and $\delta \in (0, 1)$. Partition the interval $[m/\delta]$ into m intervals $I_1, \ldots, I_m$ of size $1/\delta$ each.*

In the Augmented Set List problem $AugSetList(k, m, \delta)$, Alice gets a list of subsets $S_1, \ldots, S_k \subset [m/\delta]$, such that each S_i has size exactly m and contains exactly one element from each interval $I_1, \ldots, I_m$. Bob gets an index $i \in [k]$, an element $e \in [m/\delta]$ and a subset T of S_i that contains exactly the elements of S_i that are smaller than e. Bob needs to decide whether $e \in S_i$, and succeed with probability at least $1 - \delta$.

The equivalence to Matrix Augmented Indexing is not hard to show; the details are similar to Molinaro et al. (2013) and we omit them here. By the equivalence, we get the following corollary from Corollary C.6.

Corollary C.8 *In the abortion model, the one-way communication complexity of $AugSetList(k, m, \delta)$ is $\Omega(km\log(1/\delta))$.*

Next we define the q-fold version of the same problem.

Definition C.9 *In the problem q-$AugSetList(k, m, \delta)$, Alice and Bob get q instances of $AugSetList(k, m, \delta/q)$, and Bob needs to answer correctly on all of them simoultaneously with probability at least $1 - \delta$.*

The main tehcnical result of Molinaro et al. (2013) is, loosely speaking, a direct-sum theorem which lifts a lower bound in the abortion model to a q-fold lower bound in the usual model. Applying their theorem to Corollary C.8, we obtain the following.

Corollary C.10 *The one-way communication complexity of q-$AugSetList(k, m, \delta)$ is $\Omega(qkm\log(q/\delta))$.*

Finally, we construct a "generalized augmented indexing" problem over r copies of the above problem.

Definition C.11 *In the problem r-$Ind(q$-$AugSetList(k, m, \delta))$, Alice gets r instances $A_1, \dots, A_r$ of q-$AugSetList(k, m, \delta)$. Bob gets an index $j \in [r]$, his part B_j of instance j, and Alice's instances $A_1, \dots, A_{j-1}$. Bob needs to solve instance j with success probability at least $1 - \delta$.*

By standard direct sum results in communication complexity (reproduced in Molinaro et al. (2013)) we obtain from Corollary C.10 the final lower bound we need.

Proposition C.12 *The one-way communication complexity of r-$Ind(q$-$AugSetList(k, m, \delta))$ is $\Omega(rqkm\log(q/\delta))$.*

C.2.3. Reductions to All-Cross-Distances

We now prove Lemma C.2, by reducing r-$Ind(q$-$AugSetList(k, m, \delta))$ to the all-cross-distances problem. We will use two reductions, to get a lower bound once in terms of d and once in terms of Φ. Specifically, in the first reduction we will set $m = 1/\epsilon^2$, $k = n/q$ and $r = \rho\log d$ (where ρ is the constant from the statement of Lemma C.2). Then the lower bound we would get by Proposition C.12 is $\Omega\left(\epsilon^{-2}n\log d\log(q/\delta)\right)$. In the second reduction we will set $r = \rho\log\Phi$, yielding the lower bound $\Omega\left(\epsilon^{-2}n\log\Phi\log(q/\delta)\right)$. Together they lead to Lemma C.2.

In both settings, recall we are reducing to the following problem: For dimension $d = \Omega(\epsilon^{-2}\log(q/\delta))$ and aspect ratio Φ, Alice gets n points, Bob gets q points, and Bob needs to estimate all cross-distances up to distortion $1 \pm \epsilon$.

Consider an instance of r-$Ind(q$-$AugSetList(k, m, \delta))$. It can be visualized as follows: Alice gets a matrix S with $n = qk$ rows and r columns, where each entry contains a set of size m. Bob gets an index $j \in [r]$, indices $i_1, \dots, i_q \in [k]$, elements $e_1, \dots, e_q$, subsets $T_1 \subset S(i_1, j), \dots, T_q \subset S(i_q, j)$, and the first $j - 1$ columns of the matrix S.

We now use the encoding scheme of Jayram and Woodruff (2013), in the set formulation which was given in Molinaro et al. (2013). We restate the result.

Lemma C.13 (Jayram and Woodruff (2013)) *Let $m = 1/\epsilon^2$ and $0 < \eta < 1$. Suppose we have the following setting:*

- *Alice has subsets $S_1, \dots, S_r$ of $[m/\eta]$.*

- *Bob has an index $j \in [r]$, an element $e \in [m/\eta]$, the subset $T \subset S_j$ of elements smaller than e, and the sets $S_1, \ldots, S_{j-1}$.*

There is a shared-randomness mapping of their inputs into points v_A, v_B and a scale $\Psi > 0$ (the scale is known to both), such that

1. *$v_A, v_B \in \{0,1\}^D$ for $D = O(\epsilon^{-2}\log(\frac{1}{\eta})\exp(r))$.*
2. *If $e \in S_j$ (YES instance) then w.p. $1-\eta$, $\|v_A - v_B\|^2 \le (1-2\epsilon)\Psi$.*
3. *If $e \notin S_j$ (NO instance) then w.p. $1-\eta$, $\|v_A - v_B\|^2 \ge (1-\epsilon)\Psi$*

C.2.4. Lower Bound in terms of d

We start with the first reduction that yields a lower bound in terms of d.

Lemma C.14 *Under the assumptions of Lemma C.2, for the all-cross-distances problem, Alice must use a sketch of at least $\Omega(\epsilon^{-2}n\log(d)\log(q/\delta))$ bits.*

Proof We invoke Lemma C.13 with $r = \rho\log d$ and $\eta = \delta/q$. Note that the latter is the desired success probability in each instance of $q\text{-}AugSetList(k,m,\delta)$ (cf. Definition C.9). Alice encodes each row of the matrix, $(S(i,1),\ldots,S(i,r))$, into a point x_i, thus n points $x_1,\ldots,x_n$. Bob encodes $(S(i,1),\ldots,S(i_z,j-1),T_i,j,e_z)$ for each $z \in [q]$ into a point y_z, thus q points $y_1,\ldots,y_z$. For every $z \in [q]$, the problem represented by row i_z in the matrix S is reduced by Lemma C.13 to estimating the distance $\|x_{i_z} - y_z\|$. By Item 1 of Lemma C.13, the points $\{x_i\}_{i\in[n]}$, $\{y_z\}_{z\in[q]}$ have binary coordinates and dimension $D = O(\epsilon^{-2}\log(q/\delta)d^{\rho})$. By the hypothesis $d^{1-\rho} \ge \epsilon^{-2}\log(q/\delta)$ of Lemma C.2, $D = O(d)$. Therefore Alice and Bob can now feed them into a given black-box solution of the all-cross-distances problem, which estimates all the required distances and solves $r\text{-}Ind(q\text{-}AugSetList(k,m,\delta))$.

Let us establish the success probability of the reduction. Since we set $\eta = \delta/q$ in Lemma C.13, it preserves each distance $\|x_{i_z} - y_z\|$ for $z \in [q]$ with probability $1-\delta/q$. By a union bound, it preserves all of them simultaneously with probability $1-\delta$. The success probability of the all-cross-distances problem, simultaneously on all query points $\{\tilde{y}_z : z \in [q]\}$, is again $1-\delta$. Altogether, the reduction succeeds with probability $1-O(\delta)$. As a result, the all-cross-distances problem solves the given instance of $r\text{-}Ind(q\text{-}AugSetList(k,m,\delta))$, and Lemma C.14 follows. ■

C.2.5. Lower Bound in terms of Φ

We proceed to the second reduction that would yield a lower bound in terms of Φ.

Lemma C.15 *Under the assumptions of Lemma C.2, for the all-cross-distances problem, Alice must use a sketch of at least $\Omega(\epsilon^{-2}n\log(\Phi)\log(q/\delta))$ bits.*

Proof We may assume that $\Phi \ge d$ since otherwise Lemma C.15 already follows from Lemma C.14. Therefore $\Phi^{1-\rho} \ge \epsilon^{-2}\log(q/\delta)$.

The reduction is very similar to the one in Lemma C.14. Again we evoke Lemma C.13 with $\eta = \delta/q$, but this time we set $r = \rho\log\Phi$. Again we denote Alice's encoded points by $x_1,\ldots,x_n$, and Bob's by $y_1,\ldots,y_q$. By Item 1 of Lemma C.13, the points have binary coordinates and dimension

$D = O(\epsilon^{-2} \log(q/\delta)\Phi^{\rho})$. The difference from Lemma C.14 is that since it is possible that $\Phi \gg d$, the dimension D is too large for the given black-box solution of the all-cross-distances problem (which is limited to dimension $O(d)$).

To solve this, Alice and Bob project their points into dimension $D' = O(\epsilon^{-2} \log(q/\delta))$ by a Johnson-Lindenstrauss transform, using shared randomness. Let $\tilde{x}_1, \ldots, \tilde{x}_n$ and $\tilde{y}_1, \ldots, \tilde{y}_z$ denote the projected points. After the projection each coordinate has magnitude at most $O(\epsilon^{-2} \log(q/\delta)\Phi^{\rho})$. By our assumption $\Phi^{1-\rho} \geq \Omega(\epsilon^{-2} \log(q/\delta))$, this is at most $O(\Phi)$. Since the dimension D' is $O(d)$, Alice and Bob can now feed $\tilde{x}_1, \ldots, \tilde{x}_n$ and $\tilde{y}_1, \ldots, \tilde{y}_z$ into a given black-box solution of the all-cross-distances problem with dimension $O(d)$ and aspect ratio $O(\Phi)$.

Let us establish the success probability of the reduction. As before, Lemma C.13 preserves all the required distances, $\|x_{i_z} - y_z\|$ for $z \in [q]$, with probability $1 - \delta$. The Johnson-Lindenstrauss transform into dimension D' preserves each distance as $\|\tilde{x}_{i_z} - \tilde{y}_z\|$ with probability at least $1 - \delta$, since we picked the dimension to be $D' = O(\epsilon^{-2} \log(\frac{q}{\delta}))$. The success probability of the all-cross-distances problem simultaneously is again $1 - \delta$. Altogether, the reduction succeeds with probability $1 - O(\delta)$. As a result, the all-cross-distances problem solves the given instance of $r\text{-}Ind(q\text{-}AugSetList(k, m, \delta))$, and Lemma C.15 follows. ■

C.2.6. Conclusion

Lemmas C.14 and C.15 together imply Lemma C.2. The latter, together with Lemma C.1, implies Theorem 1.3. ■

Appendix D. Practical Variant

Indyk et al. (2017) presented a simplified version of the sketch of Indyk and Wagner (2017), which is lossier by a factor $O(\log\log n)$ in the size bound (more precisely it uses $O(\epsilon^{-2} \log(n)(\log\log(n) + \log(1/\epsilon)) + \log\log\Phi)$ bits per point; compare this to Table 1), but on the other hand is practical to implement and was shown to work well empirically. Both variants do not provably support out-of-sample queries.

In the main part of this work, we showed how to adapt the framework of Indyk and Wagner (2017) to support out-of-sample queries with nearly optimal size bounds. The goal of this section is to show that our techniques can also be applied in a simplified way to Indyk et al. (2017) in order to obtain a *practical* algorithm. Specifically, focusing on the all-nearest-neighbors problem, we will show that a slight modification to Indyk et al. (2017) yields provable support in out-of-sample approximate nearest neighbor queries, with a size bound that is the same as in Theorem 1.1 plus an additive $O(\epsilon^{-2} \log(n) \log\log(n))$ term.

Technique: Middle-out compression In Indyk et al. (2017), every 1-path is pruned (i.e. replaced by a long edge) except for its top Λ nodes, where Λ is an integer parameter. Combining this "bottom-out" compression with the "top-out" compression which was introduced in Section 3, we obtain *middle-out compression*: every long 1-path longer than 2Λ is replaced by a long edge, except for its top and bottom Λ nodes. As we will show in the remainder of this section, applying this pruning rule to the quadtree of Indyk et al. (2017) (instead of their "bottom-out" rule) is sufficient to obtain a sketch that provably supports out-of-sample approximate nearest neighbor queries. Thus, the sketching algorithm is nearly unchanged.

We remark that in Section 3 we introduced two additional modifications: *grid-net quantization* and *surrogate hashing*. These were required in order to prove Theorems 1.1 and 1.2, but in the framework of Indyk et al. (2017) they turn out to be unnecessary: grid-net quantization is already organically built into the quadtree approach of Indyk et al. (2017), and surrogate hashing only served to avoid a $O(\log\log n)$ factor in the sketch size (see footnote 4), but in Indyk et al. (2017) this factor is tolerated anyway.

D.1. Sketching Algorithm Recap

For completeness, let us briefly describe the sketching algorithm of Indyk et al. (2017) (the reader is referred to that paper for more formal details), with our modification. To this end, set

$$\Lambda = \lceil \log\left(\frac{16d^{1.5}\log\Phi}{\epsilon\delta}\right)\rceil.$$

Suppose w.l.o.g. that Φ is a power of 2. The sketching algorithm proceeds in three steps:

1. *Random shifted grids:* Impose a randomly shifted enclosing hypercube on the data points X. More precisely, choose a uniformly random shift $\sigma \in \{-\Phi,\dots,\Phi\}^d$, and set the enclosing hypercube to be $H = [-2\Phi+\sigma_1, 2\Phi+\sigma_1]\times[-2\Phi+\sigma_2, 2\Phi+\sigma_2]\times\ldots\times[-2\Phi+\sigma_d, 2\Phi+\sigma_d]$. Since $X \subset \{-\Phi,\dots,\Phi\}^d$, it is indeed enclosed by H. We then half H along every dimension to create a finer grid with 2^d cells, and proceed so (recursively halving every cell along every dimension) to create a hierarchy of nested grids, with $\log(4\Phi)+\Lambda$ hierarchy levels. The top level is numbered $\Phi+2$, which is the log the side length of H, and the next levels are decrementing, so that the grid cells in level ℓ have side length 2^ℓ.

2. *Quadtree construction:* Construct the quadtree which is naturally associated with the nested grids: the root corresponds to H, its children correspond to the non-empty cells of the next grid in the hierarchy (a cell is non-empty if it contains a point in X), and so on. Each tree edge is annotated by a bitstring of length d, that marks whether the child cell coincides with the bottom half (bit 0) or the top half (bit 1) of the parent cell in each dimension.

3. *Middle-out compression:* For every path of degree-1 tree nodes whose length is more than 2Λ, we keep its top Λ and bottom Λ, and replace its remaining middle portion by a long edge. This removes the edge annotations of the middle section (this achieving compression). We label each long edge with the length of the path it replaces.

In the remainder of this section we prove the following.

Theorem D.1 *The above algorithm, with the above setting of Λ, runs in time $\tilde{O}(nd(\log\Phi+\Lambda))$ and produces a sketch for the all-nearest-neighbors problem, whose size in bits is*

$$O\left(n\left(\frac{\log n\cdot(\log\log n+\log(1/\epsilon))}{\epsilon^2}+\log\log\Phi+\log\left(\frac{q}{\delta}\right)\right)+d\log\Phi+\log\left(\frac{q}{\delta}\right)\log\left(\frac{\log(q/\delta)}{\epsilon}\right)\right).$$

The sketch size is the same as in Indyk et al. (2017), except that we keep at most 2Λ instead of Λ nodes per 1-path, which increases the sketch size by only a factor of 2.

D.2. Basic lemmas

We start with some useful properties of the above sketch, which are analogous to lemmas from in Section 3. In the notation below, for a node v in the quadtree, $C(v)$ denotes the subset of points in X that are contained in the grid cell associated with v. As in Section 3, the quadtree is partitioned into a set $\mathcal{F}(T)$ of *subtrees* by removing the long edges.

Lemma D.2 (analog of Lemma 3.1) *For every point $x \in X$, with probability $1-\delta$, the following holds. If $z \in \mathbb{R}^d$ is any point outside the grid cell that contains x in level ℓ of the quadtree, then $\|x-x'\| \geq 8\epsilon^{-1} \cdot 2^{\ell-\Lambda}\sqrt{d}$.*

Proof The setting of Λ is such that with probability $1-\delta$, in every level ℓ of the quadtree, the grid cell that contains x also contains the ball at radius $8\epsilon^{-1} \cdot 2^{\ell-\Lambda}\sqrt{d}$ around x. (This property is known as "padding".) The lemma is just a restatement of this property. See Lemma 1 and Equation (1) in Indyk et al. (2017) for details. ■

Lemma D.3 (analog of Lemmas 3.3, 3.4, 3.6) *Let v be a node in the quadtree, and $x, x' \in \mathbb{R}^d$ points contained in the grid cell associated with v. Then $\|x-x'\| \leq 2^{\ell(v)}\sqrt{d}$.*

Proof The grid cell associated with v is a hypercube with side $2^{\ell(v)}$ and diameter $2^{\ell(v)}\sqrt{d}$. ■

Before proceeding let us make the following point about the quadtree.

Claim D.4 *For every leaf v of the quadtree, $C(v)$ contains a single point of X, and v is the bottom of a 1-path of length at least Λ.*

Proof Refining the quadtree grid hierarchy for $\log(4\Phi)$ levels ensures that each grid cell contains at most one point from X, and refining for Λ additional levels ensures that each leaf is the bottom of a 1-path of length at least Λ. ■

Claim D.5 *Every subtree leaf in the quadtree is the bottom of a 1-path of length at least Λ.*

Proof If v is a leaf of the quadtree, this follows from Claim D.4. Otherwise this follows from middle-out compression. ■

Next we define centers and surrogates. Centers $c(v)$ are chosen similarly to Section 3. The surrogate $s^*(v)$ of every tree node v is simply defined to be the "bottom-left" (i.e. minimal in all dimensions) corner of the grid cell associated with v.

D.3. Approximate Nearest Neighbor Search

Finally, we can describe the query algorithm and complete its analysis. Let y be a query point for which we need to report an approximate nearest neighbor from the sketch. The query algorithm is the same as in Section 4: starting with the subtree that contains the quadtree root, it recovers the surrogates in the current subtree and chooses the subtree v whose surrogate is the closest to y. If v is a quadtree leaf, its center is returned as the approximate nearest neighbor. Otherwise, the algorithm proceeds by recursion on the subtree under v.

Surrogate recovery The difference is in the way we recover the surrogates of a given subtree. In Section 4 this was done using the surrogate hashes. Here we will use a simpler, deterministic surrogate recovery subroutine. Let $s^*(H) \in \mathbb{R}^d$ the surrogate of the quadtree root. (We store this point explicitly in the sketch, and it will be convenient to think of it w.l.o.g. as the the origin in $\mathbb{R}^d$.) As observed in Indyk et al. (2017), for every tree node v, if we concatenate the bits annotating the edges on the path from the root to v, we get the binary expansion of the point $s^*(H) + s^*(v)$. Therefore, we can recover $s^*(v)$ from the sketch, as long as the path from the root to v does not traverse a long edge.

If the path to v contains long edges (and thus missing bits in the binary expansion of $s^*(v)$), the algorithm completes these bits from the binary expansion of y. Let $r_0, r_1, \dots$ be the subtree roots traversed by the algorithm, and let $T_0, T_1, \dots$ be the corresponding subtrees. Let t be the smallest such that the algorithm does not recover the surrogates in T_t correctly (because the bits missing on the long edge connecting T_{t-1} to T_t are not truly equal to those of y). As in Section 4, the query algorithm does not know t (it simply always assumes that the bits of y are the correct missing ones), but we will use it for analysis. Note that by definition of t, all surrogates in the subtrees rooted at $r_0, \dots, r_{t-1}$ are recovered correctly. Thus, the event from Lemma 4.2 holds deterministically.

Proof of Theorem D.1 Let $x^* \in X$ be a fixed true nearest neighbor of y in X (chosen arbitrarily if there is more than one). We shall assume that the event in Lemma D.2 occurs for x^*.

Lemma D.6 (analog of Lemma 4.3) *Let $T' \in \mathcal{F}(T)$ be a subtree rooted in r, such that $x^* \in C(r)$. Let v a leaf of T' that minimizes $\|y - s^*(v)\|$. Then either $x^* \in C(v)$, or every $z \in C(v)$ is a $(1 + O(\epsilon))$-approximate nearest neighbor of y.*

Proof If $x^* \in C(v)$ then we are done. Assume now that $x^* \in C(u)$ for a leaf $u \neq v$ of T'. Let $\ell := \max\{\ell(v), \ell(u)\}$. We start by showing that $\|y - x^*\| > \epsilon^{-1} 2^\ell \sqrt{d}$. Assume by contradiction this is not the case. Since $x^* \in C(u)$ we have $\|x^* - x_{c(u)}\| \le 2^\ell \sqrt{d}$ by Lemma D.3, and similarly $\|x_{c(u)} - s^*(u)\| \le 2^\ell \sqrt{d}$. Together, $\|y - s^*(u)\| \le (\epsilon^{-1} + 2) 2^\ell \sqrt{d}$. On the other hand, by the triangle inequality, $\|y - s^*(v)\| \ge \|x^* - x_{c(v)}\| - \|y - x^*\| - \|x_{c(v)} - s^*(v)\|$. By Claim D.5, both v and u are the bottom of 1-paths of length at least Λ, This means that x^* and $x_{c(v)}$ are separated already at level $\ell + \Lambda$, and by Lemma D.2 this implies $\|x^* - x_{c(v)}\| \ge 8\epsilon^{-1} \cdot 2^\ell \sqrt{d}$. By the contradiction hypothesis we have $\|y - x^*\| \le \epsilon^{-1} 2^\ell \sqrt{d}$, and by Lemma D.3, $\|x_{c(v)} - s^*(v)\| \le 2^\ell \sqrt{d}$. Putting these together yields $\|y - s^*(v)\| \ge 8\epsilon^{-1} \cdot 2^\ell \sqrt{d} - \epsilon^{-1} 2^\ell \sqrt{d} - 2^\ell \sqrt{d} > (\epsilon^{-1} + 2) 2^\ell \sqrt{d} \ge \|x_{c(u)} - s^*(u)\|$. This contradicts the choice of v.

The lemma now follows because for every $z \in C(v)$,

$$\begin{aligned}
\|y - z\| &\le \|y - s^*(v)\| + \|s^*(v) - x_{c(v)}\| + \|x_{c(v)} - z\| && (7)\\
&\le \|y - s^*(u)\| + \|s^*(v) - x_{c(v)}\| + \|x_{c(v)} - z\| && (8)\\
&\le \|y - x^*\| + \|x^* - x_{c(u)}\| + \|x_{c(u)} - s^*(u)\| + \|s^*(v) - x_{c(v)}\| + \|x_{c(v)} - z\| && (9)\\
&\le \|y - x^*\| + 4 \cdot 2^\ell \sqrt{d} && (10)\\
&\le (1 + 4\epsilon)\|y - x^*\|, && (11)
\end{aligned}$$

where (7) and (9) are by the triangle inequality, (8) is since $\|y - s^*(v)\| \le \|y - s^*(u)\|$ by choice of v, (10) is by applying Lemma D.3 to each of the last four summands, and (11) is since we have shown that $\|y - x^*\| > \epsilon^{-1} 2^\ell \sqrt{d}$. Therefore z is a $(1 + 4\epsilon)$-approximate nearest neighbor of y. ■

Now we prove that the query algorithm returns an approximate nearest neighbor for y. We may assume w.l.o.g. that ϵ is smaller than a sufficiently small constant. We consider two cases. In the first case, $x^* \notin C(r_t)$. Let $i \in \{1, \ldots, t\}$ be the smallest such that $x^* \notin C(r_i)$. By applying Lemma D.6 on r_{i-1}, we have that every point in $C(r_i)$ is a $(1+O(\epsilon))$-approximate nearest neighbor of y. After reaching r_i, the algorithm would return the center of some leaf reachable from r_i, and it would be a correct output.

In the second case, $x^* \in C(r_t)$ contains a true nearest neighbor of y. We will show that every point in $C(r_t)$ is a $(1+O(\epsilon))$-approximate nearest neighbor of y, so once again, once the algorithm arrives at r_t it can return anything. By definition of t, we know that y does not reside in the grid cell associated with r_t. Since y does reside in that cell, we have $\|y - x^*\| \geq 8\epsilon^{-1}2^{\ell(r_t)-\Lambda}\sqrt{d}$ by Lemma D.2. On the other hand, by Claim D.5, r_t is the bottom of a 1-path of length at least Λ, and therefore any two points in $C(r_t)$ are contained in the same grid cell at level $\ell(r_t) - \Lambda$, whose diameter is $2^{\ell(r_t)-\Lambda}\sqrt{d}$. In particular, for every $x \in C(r_t)$ we have $\|x - x^*\| \leq 2^{\ell(r_t)-\Lambda}\sqrt{d} \leq \frac{1}{8}\epsilon\|y - x^*\|$. Altogether we get $\|y - x\| \leq \|y - x^*\| + \|x^* - x\| \leq (1 + \frac{1}{8}\epsilon)\|y - x^*\|$, so every $x \in C(r_t)$ is a $(1+\epsilon)$-approximate nearest neighbor of y in X.

The proof assumes the event in Lemma D.2 holds for x^*, which happens with probability $1-\delta$. To handle q queries, we can scale δ down to δ/q and take a union bound over the q nearest neighbors of the q query points ■