Proximity in the Age of Distraction: Robust Approximate Nearest Neighbor Search

Sariel Har-Peled* Sepideh Mahabadi†

November 24, 2015

Abstract

We introduce a new variant of the nearest neighbor search problem, which allows for some coordinates of the dataset to be arbitrarily corrupted or unknown. Formally, given a dataset of n points $P = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ in high-dimensions, and a parameter k, the goal is to preprocess the dataset, such that given a query point $\boldsymbol{q}$, one can compute quickly a point $\boldsymbol{x} \in P$, such that the distance of the query to the point $\boldsymbol{x}$ is minimized, when ignoring the "optimal" k coordinates. Note, that the coordinates being ignored are a function of both the query point and the point returned.

We present a general reduction from this problem to answering ANN queries, which is similar in spirit to LSH (locality sensitive hashing) [IM98]. Specifically, we give a sampling technique which achieves a bi-criterion approximation for this problem. If the distance to the nearest neighbor after ignoring k coordinates is r, the data-structure returns a point that is within a distance of $O(r)$ after ignoring $O(k)$ coordinates. We also present other applications and further extensions and refinements of the above result.

The new data-structures are simple and (arguably) elegant, and should be practical – specifically, all bounds are polynomial in all relevant parameters (including the dimension of the space, and the robustness parameter k).

1. Introduction

The *nearest neighbor* problem (NN) is a fundamental geometric problem which has major applications in many areas such as databases, computer vision, pattern recognition, information retrieval, and many others. Given a set P of n points in a d-dimensional space, the goal is to build a data-structure, such that given a query point $\boldsymbol{q}$, the algorithm can report the closest point in P to the query $\boldsymbol{q}$. A particularly interesting and well-studied instance is when the points live in a d-dimensional real vector space $\mathbb{R}^d$. Efficient exact and approximate algorithms are known for this problem. (In the *approximate nearest neighbor* (ANN) problem, the algorithm is allowed to report a point whose distance to the query is larger by at most a factor of $(1+\varepsilon)$, than the real distance to the nearest point.) See [AMN+98; Kle97; HIM12; KOR00; Har01; KL04; DIIM04; CR10; Pan06; AC06; AI08; AINR14; AR15], the surveys

*Department of Computer Science; University of Illinois; 201 N. Goodwin Avenue; Urbana, IL, 61801, USA; sariel@illinois.edu; http://sarielhp.org. Work on this paper was partially supported by a NSF AF awards CCF-0915984 and CCF-1217462.

†Department of EECS; MIT; 77 Massachusetts Avenue, Cambridge, MA 02139, USA; mahabadi@mit.edu. This work was supported in part by NSF grant IIS -1447476.

[Sam05], [Ind04] and [SDI06], and references therein (of course, this list is by no means exhaustive). One of the state of the art algorithms for ANN, based on Locality Sensitive Hashing, finds the $(1+\varepsilon)$-ANN with query time $\widetilde{O}(dn^c)$, and preprocessing/space $O(dn + n^{1+c})$ in L_1, where $c = O((1+\varepsilon)^{-1})$ [IM98], where $\widetilde{O}$ hides a constant that is polynomial in $\log n$. For the L_2 norm, this improves to $c = O((1+\varepsilon)^{-2})$ [AI08].

Despite the ubiquity of nearest neighbor methods, the vast majority of current algorithms suffer from significant limitations when applied to data sets with corrupt, noisy, irrelevant or incomplete data. This is unfortunate since in the real world, rarely one can acquire data without some noise embedded in it. This could be because the data is based on real world measurements, which are inherently noisy, or the data describe complicated entities and properties that might be irrelevant for the task at hand.

In this paper, we address this issue by formulating and solving a variant of the nearest neighbor problem that allows for some data coordinates to be arbitrarily corrupt. Given a parameter k, the ***k-Robust Nearest Neighbor*** for a query point $\boldsymbol{q}$, is a point $\boldsymbol{x} \in P$ whose distance to the query point is minimized ignoring "the optimal" set of k-coordinates (the term 'robust' ANN is used as an analogy to *Robust PCA* [CLMW11]). That is, the k coordinates are chosen so that deleting these coordinates, from both $\boldsymbol{x}$ and $\boldsymbol{q}$ minimizes the distance between them. In other words, the problem is to solve the ANN problem in a different space (which is definitely not a metric), where the distance between any two points is computed ignoring the worst k coordinates. To the best of our knowledge, this is the first paper considering this formulation of the Robust ANN problem.

This problem has natural applications in various fields such as computer vision, information retrieval, etc. In these applications, the value of some of the coordinates (either in the dataset points or the query point) might be either corrupted, unknown, or simply irrelevant. In computer vision, examples include image de-noising where some percent of the pixels are corrupted, or image retrieval under partial occlusion (e.g. see [HE07]), where some part of the query or the dataset image has been occluded. In these applications there exists a perfect match for the query after we ignore some dimensions. Also, in medical data and recommender systems, due to incomplete data [SWC+09; CFV+13; WCNK13], not all the features (coordinates) are known for all the people/recommendations (points), and moreover, the set of known values differ for each point. Hence, the goal is to find the perfect match for the query ignoring some of those features.

For the binary hypercube, under the Hamming distance, the k-robust nearest neighbor problem is equivalent to the *near neighbor* problem. The near neighbor problem is the decision version of the ANN search, where a radius r is given to the data structure in advance, and the goal is to report a point that is within distance r of the query point. Indeed, there exists a point $\boldsymbol{x}$ within distance r of the query point $\boldsymbol{q}$ if and only if r coordinates can be ignored such that the distance between $\boldsymbol{x}$ and $\boldsymbol{q}$ is zero.

Budgeted Version. We also consider the weighted generalization of the problem where the amount of uncertainty varies for each feature. In this model, each coordinate is assigned a weight $0 \leq w_i \leq 1$ in advance, which tries to capture the certainty level about the value of the coordinate ($w_i = 1$ indicates that the value of the coordinate is correct and $w_i = 0$ indicates that it can not be trusted). The goal is to ignore a set of coordinates B of total weight at most 1, and find a point $\boldsymbol{x} \in P$, such that the distance of the query to the point $\boldsymbol{x}$ is minimized ignoring the coordinates in B. Surprisingly, even computing the distance between two points under this measure is NP-Complete (it is almost an instance of Min Knapsack).

1.1. Results

We present the following new results:

(A) Reduction from Robust ANN to ANN. We present a general reduction from the robust ANN problem to the "standard" ANN problem. This results in a bi-criterion constant factor approximation, with sublinear query time, for the k-robust nearest neighbor problem.

(I) For L_1-norm the result can be stated as follows. If there exists a point $\boldsymbol{q}^*$ whose distance to the query point $\boldsymbol{q}$ is at most r by ignoring k coordinates, the new algorithm would report a point $\boldsymbol{x}$ whose distance to the query point is at most $O(r/\delta)$, ignoring $O(k/\delta)$ coordinates. The query algorithm performs $O(n^\delta)$ ANN queries in 2-ANN data structures, where $\delta \in (0,1)$ is a prespecified parameter.

(II) In Section 3, we present the above result in the somewhat more general settings of the L_ρ norm. The algorithm reports a point whose distance is within $O(r(c+\frac{1}{\delta})^{1/\rho})$ after ignoring $O(k(\frac{1}{\delta}+c))$ coordinates while performing n^δ of $c^{1/\rho}$-ANN queries.

(B) $(1+\varepsilon)$-approximation. We modify the new algorithm to report a point whose distance to the query point is within $r(1+\varepsilon)$ by ignoring $O(\frac{k}{\delta\varepsilon})$ coordinates while performing $\widetilde{O}(n^\delta/\varepsilon)$ ANN queries (specifically, $(1+O(\varepsilon))$-ANN). For the sake of simplicity of exposition, we present this extension only in the L_1 norm. See Appendix A for details.

(C) Budgeted version. In Section 4, we generalize our algorithm for the weighted case of the problem. If there exists a point within distance r of the query point by ignoring a set of coordinates of weight at most 1, then our algorithm would report a point whose distance to the query is at most $O(r)$ by ignoring a set of coordinates of weight at most $O(1)$. Again, for the sake of simplicity of exposition, we present this extension only in the L_1 norm.

(D) Data sensitive LSH queries. It is a well known phenomenon in proximity search (e.g. see Andoni *et al.* [ADI+06, Section 4.5]), that many data-structures perform dramatically better than their theoretical analysis. Not only that, but also they find the real nearest neighbor early in the search process, and then spend quite a bit of time on proving that this point is indeed a good ANN. It is natural then to ask whether one can modify proximity search data-structures to take an advantage of such a behavior. That is, if the query is easy, the data-structure should answer quickly (and maybe even provide the exact nearest neighbor in such a case), but if the instance is hard, then the data-structure works significantly harder to answer the query.

As an application of our sampling approach, we show a data-sensitive algorithm for LSH for the binary hypercube case under the Hamming distance. The new algorithm solves the Approximate Near Neighbor problem, in time $O(d\exp(\Delta)\log n)$, where Δ is the smallest value with $\sum_{i=1}^{n}\exp\bigl(-\Delta\,\mathrm{dist}(\boldsymbol{q},\boldsymbol{x}_i)/r\bigr) \le 1$, where $\mathrm{dist}(\boldsymbol{q},\boldsymbol{x}_i)$ is the distance of the query $\boldsymbol{q}$ to the ith point $\boldsymbol{x}_i \in P$, and r is the distance being tested.

We also get that such LSH queries works quickly on low dimensional data, see Remark 5.3.2 for details.

The new algorithms are clean and should be practical. Moreover, our results for the k-robust ANN hold for a wide range of the parameter k, from $O(1)$ to $O(d)$.

1.2. Related Work

There has been a large body of research focused on adapting widely used methods for high-dimensional data processing to make them applicable to corrupt or irrelevant data. For example, Robust PCA [CLMW11] is an adaptation of the PCA algorithm that handles a limited amount of adversarial errors in the input matrix. Although similar in spirit, those approaches follow a different technical development than the one in this paper.

Also, similar approaches to robustness has been used in theoretical works. In the work of Indyk

on L_ρ sketching [Ind06], the distance between two points x and y, is defined to be the median of $\{|x_1 - y_1|, ..., |x_d - y_d|\}$. Thus, it is required to compute the L_∞ norm of their distance, but only over the smallest $d/2$ coordinates.

Finally, several generalizations of the ANN problem have been considered in the literature. Two related generalizations are the Nearest k-flat Search and the Approximate Nearest k-flat. In the former, the dataset is a set of k-flats (k-dimensional affine subspace) instead of simple points but the query is still a point (see [Mah15] for example). In the latter however, the dataset consists of a set of points but the query is now a k-flat (see for example [AIKN09; MNSS15]). We note that our problem cannot be solved using these variations (at least naively) since the set of coordinates that are being ignored in our problem are not specified in advance and varies for each query. This would mean that $\binom{d}{k}$ different subspaces are to be considered for each point. In our settings, both d and k can be quite large, and the new data-structures have polynomial dependency in both parameters.

Data sensitive LSH. The fast query time for low dimensional data was demonstrated before for an LSH scheme [DIIM04, Appendix A] (in our case, this is an easy consequence of our data-structure). Similarly, optimizing the parameters of the LSH construction to the hardness of the data/queries was done before [ADI+06, Section 4.3.1] – our result however does this on the fly for the query, depending on the query itself, instead of doing this fine tuning of the whole data-structure in advance for all queries.

1.3. Techniques

By definition of the problem, we cannot directly apply Johnson-Lindenstrauss lemma to reduce the dimensions (in the L_2 norm case). Intuitively, dimension reduction has the reverse effect of what we want – it spreads the mass of a coordinate "uniformly" in the projection's coordinates – thus contaminating all projected coordinates with noise from the "bad" coordinates.

The basic idea of our approach is to generate a set of random projections, such that all of these random projections map far points to far points (from the query point), and at least one of them projects a close point to a close point. Thus, doing ANN queries in each of these projected point sets, generates a set of candidate points, one of them is the desired robust ANN.

Our basic approach is based on a simple sampling scheme, similar to the Clarkson-Shor technique [CS89] and LSH [HIM12]. The projection matrices we use are *probing* matrices. Every row contains a single non-zero entry, thus every row copies one original coordinate, and potentially scales it up by some constant.

A sketch of the technique: Consider the case where we allow to drop k coordinates. For a given query point $\boldsymbol{q}$, it has a robust nearest neighbor $\boldsymbol{q}^* \in P$, such that there is a set B of k "bad" coordinates, such that the distance between $\boldsymbol{q}$ and $\boldsymbol{q}^*$ is minimum if we ignore the k coordinates of B (and this is minimum among all such choices).

We generate a projection matrix by picking the ith coordinate to be present with probability $1/(\alpha k)$, where α is some constant, for $i = 1, \ldots, d$. Clearly, the probability that such a projection matrix avoids picking the k bad coordinates is $(1 - \frac{1}{\alpha k})^k \approx e^{-1/\alpha}$. In particular, if we repeat this process $\beta \ln n$ times, where β is some constant, then the resulting projection avoids picking any bad coordinate with probability $\approx e^{-\beta \ln n/\alpha} = n^{-\beta/\alpha}$. On the other hand, imagine a "bad" point $\boldsymbol{x} \in P$, such that one has to remove, say, $(\alpha/\beta)k$ coordinates before the distance of the point to the query $\boldsymbol{q}$ is closer than the robust NN $\boldsymbol{q}^*$ (when ignoring only k coordinates). Furthermore, imagine the case where picking any of these coordinates is fatal – the value in each one of these bad coordinates is so large, that choosing any of these

bad coordinates results in this bad point being mapped to a far away point. Then, the probability that the projection fails to select any of these bad coordinates is going to be roughly $(1 - \frac{1}{\alpha k})^{\alpha k \ln n} \approx 1/n$. Namely, somewhat informally, with decent probability all bad points get mapped to faraway points, and the near point gets mapped to a nearby point. Thus, with probability roughly $\approx n^{-\beta/\alpha}$, doing a regular ANN query on the projected points, would return the desired ANN. As such, repeating this embedding $O(n^{\beta/\alpha} \log n)$ times, and returning the best ANN encountered would return the desired robust ANN with high probability.

The good, the bad, and the truncated. Ultimately, our technique works by probing the coordinates, trying to detect the "hidden" mass of the distance of a bad point from the query. The mass of such a distance might be concentrated in few coordinates (say, a point has $k+1$ coordinates with huge value in them, but all other coordinates are equal to the query point) – such a point is arguably still relatively good, since ignoring slightly more than the threshold k coordinates results in a point that is pretty close by.

On the other hand, a point where one has to ignore a large number of coordinates (say $2k$) before it becomes reasonably close to the query point is clearly bad in the sense of robustness. As such, our data-structure would classify points, where one has to ignore slightly more than k coordinates to get a small distance, as being close.

To capture this intuition, we want to bound the influence a single coordinate has on the overall distance between the query and a point. To this end, if the robust nearest neighbor distance, to $\boldsymbol{q}$ when ignoring k coordinates, is r, then we consider capping the contribution of every coordinate, in the distance computation, by a certain value, roughly, r/k. Under this *truncation*, our data-structure returns a point that is $O(r)$ away from the query point, where r is the distance to the k-robust NN point.

Thus, our algorithm can be viewed as a bicriterion approximation algorithm - it returns a point where one might have to ignore slightly more coordinates than k, but the resulting point is constant approximation to the nearest-neighbor when ignoring k points.

In particular, a point that is still bad after such an aggressive truncation, is amenable to the above random probing. By carefully analyzing the variance of the resulting projections for such points, we can prove that such points would be rejected by the ANN data-structure on the projected points.

Budgeted version. To solve the budgeted version of the problem we use a similar technique to importance sampling. If the weight of a coordinate is w_i, then in the projection matrix, we sample the ith coordinate with probability $\approx 1/w_i$ and scale it by a factor of $\approx w_i$. This would ensure that the set of "bad" coordinates are not sampled with probability $e^{-2/\alpha}$. Again we repeat it $\beta \ln n$ times to get the desired bounds.

Data sensitive LSH. The idea behind the data-sensitive LSH, is that LSH can be interpreted as an estimator of the local density of the point set. In particular, if the data set is sparse near the query point, not only the LSH data-structure would hit the nearest-neighbor point quickly (assuming we are working in the right resolution), but furthermore, the density estimation would tell us that this event happened. As such, we can do the regular exponential search – start with an insensitive LSH scheme (that is fast), and go on to use more sensitive LSHs, till the density estimation part tells us that we are done. Of course, if all fails, the last LSH data-structure used is essentially the old LSH scheme.

2. Preliminaries

2.1. The problem

Definition 2.1.1. For a point $\boldsymbol{x} \in \mathbb{R}^d$, let $\pi = \mathrm{sort}(\boldsymbol{x})$ be a permutation of $[\![d]\!] = \{1, \ldots, d\}$, such that $|x_{\pi(1)}| \geq |x_{\pi(2)}| \geq \cdots \geq |x_{\pi(d)}|$. For a parameter $1 \leq i \leq d$, the ***i*-tail** of $\boldsymbol{x}$ is the point $\boldsymbol{x}_{\setminus\setminus i} = \big(0, \ldots, 0, |x_{\pi(i+1)}|, |x_{\pi(i+2)}|, \ldots, |x_{\pi(d)}|\big)$. Note, that given $\boldsymbol{x} \in \mathbb{R}^d$ and i, computing $\boldsymbol{x}_{\setminus\setminus i}$ can be done in $O(d)$ time, by median selection.

Thus, given two points $\boldsymbol{x}, \boldsymbol{v} \in \mathbb{R}^d$, their distance (in the L_ρ-norm), ignoring the k worst coordinates (which we believe to be noise), is $\left\|(u-v)_{\setminus\setminus k}\right\|_\rho$. Here, we are interested in computing the nearest neighbor when one is allowed to ignore k coordinates. Formally, we have the following:

Definition 2.1.2. For parameters ρ, k, a set of points $P \subseteq \mathbb{R}^d$, and a query point $\boldsymbol{q} \in \mathbb{R}^d$, the ***k*-robust nearest-neighbor** to $\boldsymbol{q}$ in P is $\mathsf{nn}_{\setminus\setminus k}(P, \boldsymbol{q}) = \arg\min_{\boldsymbol{x} \in P} \left\|(\boldsymbol{x} - \boldsymbol{q})_{\setminus\setminus k}\right\|_\rho$, which can be computed, naively, in $O(d\,|P|)$ time.

Definition 2.1.3. For a point $\boldsymbol{x} \in \mathbb{R}^d$ and a set of coordinates $I \subseteq [\![d]\!]$, we define $\boldsymbol{x}_{\setminus I}$ to be a point in $\mathbb{R}^{d-|I|}$ which is obtained from $\boldsymbol{x}$ by deleting the coordinates that are in I.

2.1.1. Projecting a point set

Definition 2.1.4. Consider a sequence $\mathfrak{m}$ of ℓ, not necessarily distinct, integers $i_1, i_2, \ldots, i_\ell \in [\![d]\!]$, where $[\![d]\!] = \{1, \ldots, d\}$. For a point $\boldsymbol{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$, its ***projection*** by $\mathfrak{m}$, denoted by $\mathfrak{m}\boldsymbol{x}$, is the point $(x_{i_1}, \ldots, x_{i_\ell}) \in \mathbb{R}^\ell$. Similarly, the *projection* of a point set $P \subseteq \mathbb{R}^d$ by $\mathfrak{m}$ is the point set $\mathfrak{m}P = \{\mathfrak{m}\boldsymbol{x} \mid \boldsymbol{x} \in P\}$. Given a weight vector $\boldsymbol{w} = (w_1, \ldots, w_d)$ the ***weighted projection*** by a sequence $\mathfrak{m}$ of a point $\boldsymbol{x}$ is $\mathfrak{m}_{\boldsymbol{w}}(\boldsymbol{x}) = (w_{i_1} x_{i_1}, \ldots, w_{i_\ell} x_{i_\ell}) \in \mathbb{R}^\ell$.

Note, that can interpret $\mathfrak{m}$ as matrix with dimensions $\ell \times d$, where the jth row is zero everywhere except for having w_{i_j} at the i_jth coordinate (or 1 in the unweighted case), for $j = 1, \ldots, \ell$. This is a restricted type of a projection matrix.

Definition 2.1.5. Given a probability $\tau > 0$, a natural way to create a projection, as defined above, is to include the ith coordinate, for $i = 1, \ldots, d$, with probability τ. Let $\mathcal{D}_\tau$ denote the distribution of such projections.

Given two sequences $\mathfrak{m} = i_1, \ldots, i_\ell$ and $\mathfrak{n} = j_1, \ldots, j_{\ell'}$, let $\mathfrak{m}|\mathfrak{n}$ denote the *concatenated* sequence $\mathfrak{m}|\mathfrak{n} = i_1, \ldots, i_\ell, j_1, \ldots, j_{\ell'}$. Let $\mathcal{D}_\tau^t$ denote the distribution resulting from concatenating t such independent sequences sampled from $\mathcal{D}_\tau$. (I.e., we get a random projection matrix, which is the result of concatenating t independent projections.)

Observe that for a point $\boldsymbol{x} \in \mathbb{R}^d$ and a projection $\boldsymbol{M} \in \mathcal{D}_\tau^t$, the projected point $\boldsymbol{M}\boldsymbol{x}$ might be higher dimensional than the original point $\boldsymbol{x}$ as it might contain repeated coordinates of the original point.

Remark 2.1.6 (Compressing the projections). Consider a projection $\boldsymbol{M} \in \mathcal{D}_\tau^t$ that was generated by the above process (for either the weighted or unweighted case). Note, that since we do not care about the order of the projected coordinates, one can encode $\boldsymbol{M}$ by counting for each coordinate $i \in [\![d]\!]$, how many time it is being projected. As such, even if the range dimension of $\boldsymbol{M}$ is larger than d, one can compute the projection of a point in $O(d)$ time. One can also compute the distance between two such projected points in $O(d)$ times.

3. The k-robust ANN under the L_ρ-norm

In this section, we present an algorithm for approximating the k-robust nearest neighbor under the L_ρ-norm, where ρ is some prespecified fixed constant (say 1 or 2). As usual in such applications, we approximate the ρth power of the L_ρ-norm, which is a sum of ρth powers of the coordinates.

3.1. The preprocessing and query algorithms

Input. The input is a set P of n points in $\mathbb{R}^d$, and a parameter k. Furthermore, we assume we have access to a data-structure that answer (regular) $c^{1/\rho}$-ANN queries efficiently, where c is a quality parameter associated with these data-structures.

Preprocessing. Let α, β, δ be three constants to be specified shortly, such that $\delta \in (0,1)$. We set $\tau = 1/(\alpha k)$, $t = \beta \ln n$, and $L = O(n^\delta \log n)$. We randomly and independently pick L sequences $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_L \in \mathcal{D}_\tau^t$. Next, the algorithm computes the point sets $P_i = \boldsymbol{M}_i P$, for $i = 1, \ldots, L$, and preprocesses each one of them for $c^{1/\rho}$-approximate nearest-neighbor queries for the L_ρ-norm ($c \geq 1$), using a standard data-structure for ANN that supports this. Let D_i denote the resulting ANN data-structure for P_i, for $i = 1, \ldots, L$ (for example we can use the data structure of Indyk and Motwani [IM98; HIM12] for the L_1/L_2 cases).

Query. Given a query point $\boldsymbol{q} \in \mathbb{R}^d$, for $i = 1, \ldots, L$, the algorithm computes the point $\boldsymbol{q}_i = \boldsymbol{M}_i \boldsymbol{q}$, and its ANN $\boldsymbol{v}_i$ in P_i using the data-structure D_i. Each computed point $\boldsymbol{v}_i \in P_i$ corresponds to an original point $\boldsymbol{x}_i \in P$. The algorithm returns the k-robust nearest neighbor to $\boldsymbol{q}$ (under the L_ρ-norm) among $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_L$ via direct calculation.

3.2. Analysis

3.2.1. Points: Truncated, light and heavy

Definition 3.2.1. For a point $\boldsymbol{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$, and a threshold $\psi > 0$, let $\boldsymbol{x}_{\leq \psi} = (x'_1, \ldots, x'_d)$ be the ***ψ-truncated*** point, where $x'_i = \min(|x_i|, \psi)$, for $i = 1, \ldots, d$. In words, we max out every coordinate of $\boldsymbol{x}$ by the threshold ψ. As such, the *ψ-truncated L_ρ-norm* of $\boldsymbol{x} = (x_1, \cdots, x_d)$ is $\|\boldsymbol{x}_{\leq \psi}\|_\rho = \left(\sum_{i=1}^d \left(\min(|x_i|, \psi)\right)^\rho\right)^{1/\rho}$.

Definition 3.2.2. For parameters ψ and r, a point $\boldsymbol{x}$ is ***(ψ, r)-light*** if $\|\boldsymbol{x}_{\leq \psi}\|_\rho \leq r$. Similarly, for a parameter $R > 0$, a point is ***(ψ, R)-heavy*** if $\|\boldsymbol{x}_{\leq \psi}\|_\rho \geq R$.

Intuitively a light point can have only a few large coordinates. The following lemma shows that being light implies a small tail.

Lemma 3.2.3. *For a number r, if a point $\boldsymbol{x} \in \mathbb{R}^d$ is $(r/k^{1/\rho}, r)$-light, then $\|\boldsymbol{x}_{\setminus k}\|_\rho \leq r$.*

Proof: Let $\psi = r/k^{1/\rho}$, and let y be the number of truncated coordinates in $\boldsymbol{x}_{\leq \psi}$. If $y > k$ then $\|\boldsymbol{x}_{\leq \psi}\|_\rho \geq \sqrt[\rho]{y\psi^\rho} > \sqrt[\rho]{k\psi^\rho} = k^{1/\rho}\psi = r$, which is a contradiction. As such, all the non-zero coordinates of $\boldsymbol{x}_{\setminus k}$ are present in $\boldsymbol{x}_{\leq \psi}$, and we have that $\|\boldsymbol{x}_{\setminus k}\|_\rho \leq \|\boldsymbol{x}_{\leq \psi}\|_\rho \leq r$. ∎

3.2.2. On the probability of a heavy point to be sampled as light, and vice versa

Lemma 3.2.4. *Let $\boldsymbol{x} = (x_1, \ldots, x_d)$ be a point in $\mathbb{R}^d$, and consider a random $\boldsymbol{m} \in \mathcal{D}_\tau$, see Definition 2.1.5. We have that $\mathbf{E}\Big[\|\boldsymbol{m}\boldsymbol{x}\|_\rho^\rho\Big] = \tau \|\boldsymbol{x}\|_\rho^\rho$, and $\mathbf{V}\Big[\|\boldsymbol{m}\boldsymbol{x}\|_\rho^\rho\Big] = \tau(1-\tau)\|\boldsymbol{x}\|_{2\rho}^{2\rho}$.*

Proof: Let X_i be a random variable that is $|x_i|^\rho$ with probability τ and 0 otherwise. For $Z = \|\boldsymbol{m}\boldsymbol{x}\|_\rho^\rho$, we have that $\mathbf{E}[Z] = \sum_{i=1}^d \mathbf{E}[X_i] = \sum_{i=1}^d \tau |x_i|^\rho = \tau \|\boldsymbol{x}\|_\rho^\rho$. As for the variance, we have $\mathbf{V}[X_i] = \mathbf{E}[X_i^2] - (\mathbf{E}[X_i])^2 = \tau |x_i|^{2\rho} - \tau^2 |x_i|^{2\rho} = \tau(1-\tau)|x_i|^{2\rho}$. As such, we have $\mathbf{V}[Z] = \sum_{i=1}^d \mathbf{V}[X_i] = \tau(1-\tau)\sum_{i=1}^d |x_i|^{2\rho} = t\tau(1-\tau)\|\boldsymbol{x}\|_{2\rho}^{2\rho}$. ∎

Lemma 3.2.5. *Let $\boldsymbol{x}$ be a point in $\mathbb{R}^d$, and let $\psi > 0$ be a number. We have that $\|\boldsymbol{x}_{\le\psi}\|_{2\rho}^{2\rho} \le \psi^\rho \|\boldsymbol{x}\|_\rho^\rho$.*

Proof: Consider the ψ-truncated point $\boldsymbol{v} = \boldsymbol{x}_{\le\psi}$, see Definition 3.2.1. Each coordinate of $\boldsymbol{v}$ is smaller than ψ, and thus $\|\boldsymbol{v}\|_{2\rho}^{2\rho} = \sum_{i=1}^d |v_i|^{2\rho} \le \sum_{i=1}^d \psi^\rho |v_i|^\rho \le \psi^\rho \|\boldsymbol{v}\|_\rho^\rho$. ∎

Lemma 3.2.6. *Consider a sequence $\boldsymbol{m} \in \mathcal{D}_\tau$. If $\boldsymbol{x}$ is a $(r/k^{1/\rho}, R)$-heavy point and $R \ge (8\alpha)^{1/\rho} r$, then $\mathbf{Pr}\Big[\|\boldsymbol{m}\boldsymbol{x}\|_\rho^\rho \ge \frac{1}{2}\mathbf{E}\big[\|\boldsymbol{m}\boldsymbol{x}_{\le\psi}\|_\rho^\rho\big]\Big] \ge 1/2$, where $\psi = r/k^{1/\rho}$.*

Proof: Consider the ψ-truncated point $\boldsymbol{v} = \boldsymbol{x}_{\le\psi}$. Since $\boldsymbol{x}$ is $(r/k^{1/\rho}, R)$-heavy, we have that $\|\boldsymbol{x}\|_\rho \ge \|\boldsymbol{v}\|_\rho \ge R$. Now, setting $Z = \|\boldsymbol{m}\boldsymbol{v}\|_\rho^\rho$, and using Lemma 3.2.4, we have $\mu = \mathbf{E}[Z] = \tau\|\boldsymbol{v}\|_\rho^\rho$ and $\sigma^2 = \mathbf{V}[Z] = \tau(1-\tau)\|\boldsymbol{v}\|_{2\rho}^{2\rho} \le \tau(1-\tau)\psi^\rho \|\boldsymbol{v}\|_\rho^\rho$, by Lemma 3.2.5. Now, we have that $\mathbf{Pr}\big[\|\boldsymbol{m}\boldsymbol{x}\|_\rho^\rho \le \mu/2\big] \le \mathbf{Pr}\big[\|\boldsymbol{m}\boldsymbol{v}\|_\rho^\rho \le \mu/2\big]$. As such, by Chebyshev's inequality, and since $\tau = 1/(\alpha k)$, if $R \ge (8\alpha)^{1/\rho} r$, we have

$$\mathbf{Pr}\Big[Z \le \frac{\mu}{2}\Big] \le \mathbf{Pr}\Big[|Z-\mu| \ge \frac{\mu/2}{\sigma}\sigma\Big] \le \left(\frac{\sigma}{\mu/2}\right)^2 \le \frac{\tau(1-\tau)\psi^\rho\|\boldsymbol{v}\|_\rho^\rho}{\tau^2\|\boldsymbol{v}\|_\rho^{2\rho}/4} \le \frac{4\psi^\rho}{\tau\|\boldsymbol{v}\|_\rho^\rho} \le 4\frac{\alpha k r^\rho}{kR^\rho} \le \frac{1}{2}.$$ ∎

Lemma 3.2.7. *Let $\boldsymbol{x}$ be a prespecified point. The probability that a sequence $\boldsymbol{M}$ sampled from $\mathcal{D}_\tau^t$ does not sample any of the k heaviest coordinates of $\boldsymbol{x}$ is $\ge n^{-\beta/\alpha - 2\beta/(\alpha^2 k)} \approx 1/n^{\beta/\alpha}$ (for the simplicity of exposition, in the following, we use this rougher estimate).*

Proof: Let S be the set of k indices of the coordinates of $\boldsymbol{x}$ that are largest (in absolute value). The probability that $\boldsymbol{m}_i$ does not contain any of these coordinates is $(1-\tau)^k$, and overall this probability is $\nu = \big(1 - \frac{1}{\alpha k}\big)^{kt} = \big(1 - \frac{1}{\alpha k}\big)^{\alpha k(\beta/\alpha)\ln n}$. Now, we have $\nu \ge \exp\big(-\frac{\beta \ln n}{\alpha}\big)\big(1 - \frac{1}{\alpha k}\big)^{(\beta/\alpha)\ln n} \ge n^{-\beta/\alpha}\exp\big(-\frac{2\beta\ln n}{\alpha^2 k}\big) = n^{-\beta/\alpha - 2\beta/(\alpha^2 k)}$, since, for any integer m, we have $(1-1/m)^{m-1} \ge 1/e \ge (1-1/m)^m$, and $e^{-2x} \le 1 - x \le e^x$, for $x \in (0, 1/2)$. ∎

Lemma 3.2.8. *Consider a point $\boldsymbol{x}$ such that $\|\boldsymbol{x}_{\setminus k}\|_\rho \le r$ (see Definition 2.1.1). Conditioned on the event of Lemma 3.2.7, we have that $\mathbf{Pr}\Big[\|\boldsymbol{M}\boldsymbol{x}\|_\rho^\rho \ge 2t\tau r^\rho\Big] \le 1/2$, where $\boldsymbol{M} \in \mathcal{D}_\tau^t$.*

Proof: By Lemma 3.2.4 for $Z = \|\boldsymbol{M}\boldsymbol{x}_{\setminus k}\|_\rho^\rho$, we have $\mu = \mathbf{E}[Z] = t\tau\|\boldsymbol{x}_{\setminus k}\|_\rho^\rho \le t\tau r^\rho$. The desired probability is $\mathbf{Pr}[Z \ge 2\mu] \le 1/2$, which holds by Markov's inequality. ∎

Lemma 3.2.9. *Let $R \ge (8\alpha)^{1/\rho} r$. If $\boldsymbol{x}$ is a $(r/k^{1/\rho}, R)$-heavy point, then $\mathbf{Pr}\Big[\|\boldsymbol{M}\boldsymbol{x}\|_\rho^\rho \ge t\tau R^\rho/8\Big] \ge 1 - 2/n^{\beta/8}$.*

Proof: Let $\psi = r/k^{1/\rho}$ and $\boldsymbol{v} = \boldsymbol{x}_{\leq\psi}$, and for all i, let $Y_i = \|\boldsymbol{m}_i\boldsymbol{v}\|_\rho^\rho$. By Lemma 3.2.6, with probability at least half, we have that $Y_i \geq \mathbf{E}[Y_i]/2 \geq \tau\|\boldsymbol{v}\|_\rho^\rho/2 \geq \tau R^\rho/2$. In particular, let $Z_i = \min(Y_i, \tau R^\rho/2)$, and observe that $\mathbf{E}[Z_i] \geq (\tau R^\rho/2)\,\mathbf{Pr}[Y_i \geq \tau R^\rho/2] \geq \tau R^\rho/4$. Thus, we have that $\mu = \mathbf{E}[Z] = \mathbf{E}\big[\sum_{i=1}^t Z_i\big] \geq t\tau R^\rho/4$. Now set $U = t\tau R^\rho/8$ and note that $\mu \geq 2U$. Now, by Hoeffding's inequality, we have that

$$\begin{aligned}\mathbf{Pr}\Big[\|\boldsymbol{M}\boldsymbol{x}\|_\rho^\rho \leq U\Big] &\leq \mathbf{Pr}\Big[Z \leq U\Big] \leq \mathbf{Pr}\Big[|Z-\mu| \geq \mu - U\Big] \leq 2\exp\left(-\frac{2(\mu-U)^2}{t(\tau R^\rho/2)^2}\right)\\ &\leq 2\exp\left(-\frac{8(\mu/2)^2}{t\tau^2R^{2\rho}}\right) \leq 2\exp\left(-\frac{8(t\tau R^\rho/8)^2}{t\tau^2R^{2\rho}}\right) = 2\exp\left(-\frac{\beta\ln n}{8}\right) \leq \frac{2}{n^{\beta/8}}.\end{aligned}$$

■

3.2.3. Putting everything together

Lemma 3.2.10. *Let $\delta \in (0,1)$ be a parameter. One can build the data-structure described in Section 3.1 with the following guarantees. For a query point $\boldsymbol{q} \in \mathbb{R}^d$, let $\boldsymbol{q}^* \in P$ be its k-robust nearest neighbor in P under the L_ρ norm, and let $r = \|(\boldsymbol{q}^* - \boldsymbol{q})_{\setminus k}\|_\rho$. Then, with high probability, the query algorithm returns a point $\boldsymbol{v} \in P$, such that $\boldsymbol{q}-\boldsymbol{v}$ is a $\Big(r/k^{1/\rho},\, O(r(c+1/\delta)^{1/\rho})\Big)$-light. The data-structure performs $O\big(n^\delta \log n\big)$ of $c^{1/\rho}$-ANN queries under L_ρ-norm.*

Proof: We start with the painful tedium of binding the parameters. For the bad probability, bounded by Lemma 3.2.9, to be smaller than $1/n$, we set $\beta = 16$. For the good probability $1/n^{\beta/\alpha}$ of Lemma 3.2.7 to be larger than $1/n^\delta$, implies $n^\delta \geq n^{\beta/\alpha}$, thus requiring $\alpha \geq \beta/\delta$. Namely, we set $\alpha = \beta/\delta$. Finally, Lemma 3.2.9 requires $R \geq (8\alpha)^{1/\rho}r = (128/\delta)^{1/\rho}r$. Let $\lambda = \max(128/\delta,\, 16c)$ and let $R = \lambda^{1/\rho}r$.

For a query point $\boldsymbol{q}$, let $\boldsymbol{q}^*$ be its k-robust NN, and let S be the set of k largest coordinates in $\boldsymbol{z} = \boldsymbol{q} - \boldsymbol{q}^*$. Let $\mathcal{E}$ denote the event of sampling a projection $\boldsymbol{M}_i \in \mathcal{D}_\tau^t$ that does not contain any of the coordinates of S. By Lemma 3.2.7, with probability $p \approx 1/n^{\beta/\alpha} = 1/n^\delta$, the event $\mathcal{E}$ happens for the data-structure D_i, for any i.

As such, since the number of such data-structures built is $L = \Theta(n^\delta \log n) = O\big((\log n)/p\big)$, we have that, by Chernoff inequality, with high probability, that there are at least $m = \Omega(\log n)$ such data structures, say $D_1, \ldots, D_m$.

Consider such a data-structure D_i. The idea is now to ignore the coordinates of S all together, and in particular, for a point $\boldsymbol{x} \in \mathbb{R}^d$, let $\boldsymbol{x}_{\setminus S} \in \mathbb{R}^{d-k}$ be the point where the k coordinates of S are removed (as defined in Definition 2.1.3). Since by assumption $\|\boldsymbol{z}_{\setminus S}\|_\rho = \|\boldsymbol{z}_{\setminus k}\|_\rho \leq r$, by Lemma 3.2.8, with probability at least half, the distance of $\boldsymbol{M}_i\boldsymbol{q}^*$ from $\boldsymbol{M}_i\boldsymbol{q}$ is at most $\ell = (2t\tau)^{1/\rho}r$. Since there are $\Omega(\log n)$ such data-structures, we know that, with high probability, in one of them, say D_1, this holds. By Lemma 3.2.9, any point $\boldsymbol{x}_{\setminus S}$ (of P), that is $(r/k^{1/\rho}, R)$-heavy, would be in distance at least $\ell' = (t\tau/8)^{1/\rho}R \geq (2ct\tau)^{1/\rho}r = c^{1/\rho}\cdot\ell$ in the projection $\boldsymbol{M}_1$ from the projected $\boldsymbol{q}$. Since D_1 is a $c^{1/\rho}$-ANN data-structure under the L_ρ norm, we conclude that no such point can be returned, because the distance from $\boldsymbol{q}$ to $\boldsymbol{q}^*$ in this data-structure is smaller than $c^{1/\rho}\ell \leq \ell'$. Note that since for the reported point $\boldsymbol{v}$, the point $\boldsymbol{v}_{\setminus S}$ cannot be $(r/k^{1/\rho}, \lambda^{1/\rho}r)$-heavy, and that the coordinates in S can contribute at most $k\,(r/k^{1/\rho})^\rho = r^\rho$. We conclude that the point $\boldsymbol{v}$ cannot be $(r/k^{1/\rho}, (\lambda+1)^{1/\rho}r)$-heavy. Thus, the data-structure returns the desired point with high probability.

As for the query performance, the data-structure performs L queries in $c^{1/\rho}$-ANN data-structures. ■

This lemma would translate to the following theorem using Lemma 3.2.3.

3.3. The result

Theorem 3.3.1. *Let $P \subseteq \mathbb{R}^d$ be a set of n points with the underlying distance being the L_ρ metric, and $k > 0$, $\delta \in (0,1)$, and $c \geq 1$ be parameters. One can build a data-structure for answering the k-robust ANN queries on P, with the following guarantees:*

(A) Preprocessing time/space is equal to the space/time needed to store $M = O\big(n^\delta \log n\big)$ data-structures for performing $c^{1/\rho}$-ANN queries under the L_ρ metric, for a set of n points in $O((d/k)\log n)$ dimensions.

(B) The query time is dominated by the time it takes to perform $c^{1/\rho}$-ANN queries in the M ANN data-structures.

(C) For a query point $\boldsymbol{q}$, the data-structure returns, with high probability, a point $\boldsymbol{v} \in P$, such that if one ignores $O(k(1/\delta + c))$ coordinates, then the L_ρ distance between $\boldsymbol{q}$ and $\boldsymbol{v}$ is at most $O(r(1/\delta + c)^{1/\rho})$, where r is the distance of the nearest neighbor to $\boldsymbol{q}$ when ignoring k coordinates. (Formally, $\boldsymbol{q} - \boldsymbol{v}$ is $\big(r/k^{1/\rho}, O(r(c + 1/\delta)^{1/\rho})\big)$-light.)

Corollary 3.3.2. *Setting $c = 2$, the algorithm would report a point $\boldsymbol{v}$ using $2^{1/\rho}$-ANN data-structures, such that if one ignores $O(k/\delta)$ coordinates, the L_ρ distance between $\boldsymbol{q}$ and $\boldsymbol{v}$ is at most $O(r/\delta^{1/\rho})$. Formally, $\boldsymbol{q} - \boldsymbol{v}$ is $\big(r/k^{1/\rho}, O(r/\delta^{1/\rho})\big)$-light.*

4. Budgeted version

4.1. Definition of the problem

In this section, we consider the budgeted version of the problem for L_1-norm. Here, a coordinate i has a cost $\xi_i \geq 0$ of ignoring it, and we have a budget of 1, of picking the coordinates to ignore (note that since we can safely remove all coordinates of cost $\xi_i = 0$, we can assume that $\xi_i > 0$). Formally, we have a vector of ***costs*** $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_d)$, where the ith coordinate, ξ_i, is the cost of ignoring this coordinate. Intuitively, the cost of a coordinate shows how much we are certain that the value of the coordinate is correct.

The set of ***admissible projections***, is

$$\mathcal{G} = \mathcal{G}(\boldsymbol{\xi}) = \left\{ I \;\middle|\; I \subseteq [\![d]\!] \text{ and } \textstyle\sum_{i \in [\![d]\!] \setminus I} \xi_i \leq 1 \right\}. \tag{4.1}$$

Given two points $\boldsymbol{x}, \boldsymbol{v}$, their ***admissible distance*** is

$$\mathsf{d}_{\boldsymbol{\xi}}(\boldsymbol{x}, \boldsymbol{v}) = \min_{I \in \mathcal{G}} \left\| I(\boldsymbol{x} - \boldsymbol{v}) \right\|_1, \tag{4.2}$$

where we interpret I as a projection (see Definition 2.1.4).

The problem is to find for a query point $\boldsymbol{q}$ and a set of points P, both in $\mathbb{R}^d$, the *robust nearest-neighbor distance* to $\boldsymbol{q}$; that is,

$$\mathsf{d}_{\boldsymbol{\xi}}(\boldsymbol{q}, P) = \min_{\boldsymbol{x} \in P} \mathsf{d}_{\boldsymbol{\xi}}(\boldsymbol{x}, \boldsymbol{v}). \tag{4.3}$$

The point in P realizing this distance is the ***robust nearest-neighbor*** to $\boldsymbol{q}$, denoted by $\boldsymbol{q}^* = \mathsf{nn}(\boldsymbol{q}) = \mathsf{nn}_{\boldsymbol{\xi}}(\boldsymbol{q}, P)$. The unweighted version can be interpreted as solving the problem for the case where all the coordinates have uniform cost $1/k$.

Definition 4.1.1. If $\boldsymbol{q}^*$ is the nearest-neighbor to $\boldsymbol{q}$ under the above measure, then the set of *good* coordinates is $G(\boldsymbol{q}) = \arg\min_{I\in\mathcal{G}} \|I(\boldsymbol{q}^* - \boldsymbol{q})\|_1 \subseteq [\![d]\!]$, and the set of ***bad*** coordinates is $B(\boldsymbol{q}) = [\![d]\!] \setminus G(\boldsymbol{q})$.

In what follows, we modify the algorithm for the unweighted case and analyze its performance for the budgeted case. Interestingly, the problem is significantly harder.

4.1.1. Hardness and approximation of robust distance for two points

For two points, computing their distance is a special instance of Min-Knapsack. The problem is NP-Hard (which is well known), as testified by the following lemma.

Lemma 4.1.2. *Given two points $\boldsymbol{x}, \boldsymbol{v} \in \mathbb{R}^d$, and a cost vector $\boldsymbol{\xi}$, computing $\min_{I\in\mathcal{G}} \|I(\boldsymbol{x} - \boldsymbol{q})\|_1$ is* NP-Complete*, where $\mathcal{G}$ is the set of admissible projections for $\boldsymbol{\xi}$ (see Eq. (4.1)).*

Proof: This is well known, and we provide the proof for the sake of completeness.

Consider an instance of Partition with integer numbers $b_1, \ldots, b_d$. Let $\alpha = \sum_{i=1}^d b_i$, and consider the point $\boldsymbol{x} = (2b_1/\alpha, \ldots, 2b_d/\alpha)$, and set the cost vector to be $\boldsymbol{\xi} = \boldsymbol{x}$. Observe that $\|\boldsymbol{x}\|_1 = \|\boldsymbol{\xi}\|_1 = 2$. In particular, there is a point in robust distance at most 1 from the origin, with the total cost of the omitted coordinates being 1 $\iff$ the given instance of Partition has a solution.

Indeed, consider the set of coordinates I realizing $\ell = \min_{I\in\mathcal{G}} \|I(\boldsymbol{x} - 0)\|_1$. Let $B = [\![d]\!] \setminus I$, and observe that the cost of the omitted coordinates $\ell' = \|B\boldsymbol{\xi}\|_1 = \|B\boldsymbol{x}\|_1$ is at most 1 (by the definition of the admissible set $\mathcal{G}$). In particular, we have $\ell + \ell' = \|\boldsymbol{x}\|_1 = 2$ and $\ell' \leq 1$. As such, the minimum possible value of ℓ is 1, and if it is 1, then $\ell = \ell' = 1$, and I and B realize the desired partition. ∎

Adapting the standard PTAS for subset-sum for this problem, readily gives the following.

Lemma 4.1.3 ([Isl09]). *Given points $\boldsymbol{x}, \boldsymbol{v} \in \mathbb{R}^d$, and a cost vector $\boldsymbol{\xi} \in [0,1]^d$, one can compute a set $I \in \mathcal{G}(\boldsymbol{\xi})$, such that $\mathsf{d}_{\boldsymbol{\xi}}(\boldsymbol{x}, \boldsymbol{v}) \leq \|I(\boldsymbol{x} - \boldsymbol{v})\|_1 \leq (1+\varepsilon)\mathsf{d}_{\boldsymbol{\xi}}(\boldsymbol{x}, \boldsymbol{v})$. The running time of this algorithm is $O(d^4/\varepsilon)$.*

4.1.2. Embedding with scaling

Given a vector $\boldsymbol{\xi} \in [0,1]^d$, consider generating a sequence $\boldsymbol{m}$ of integers $i_1 < \cdots < i_k$, by picking the number $i \in [\![d]\!]$, into the sequence, with probability ξ_i. We interpret this sequence, as in Definition 2.1.4, as a projection, except that we further scale the jth coordinate, by a factor of $1/\boldsymbol{\xi}_{i_j}$, for $j = 1, \ldots, k$. Namely, we project the ith coordinate with probability $\boldsymbol{\xi}_i$, and if so, we scale it up by a factor of $1/\boldsymbol{\xi}_i$, for $i = 1, \ldots, d$ (naturally, coordinates with $\xi_i = 0$ would never be picked, and thus would never be scaled). Let $\mathcal{B}_{\boldsymbol{\xi}}$ denote this distribution of weighted sequences (maybe a more natural interpolation is that this is a distribution of projection matrices).

Observation 4.1.4. *Let $\boldsymbol{\xi} \in (0,1]^d$ be a cost vector with non zero entries. For any point $\boldsymbol{x} \in \mathbb{R}^d$, and a random $\boldsymbol{m} \in \mathcal{B}_{\boldsymbol{\xi}}$, we have that $\mathbf{E}\left[\|\boldsymbol{m}\boldsymbol{x}\|_1\right] = \|\boldsymbol{x}\|_1$.*

4.2. Algorithm

The input is a point set P of n points in $\mathbb{R}^d$. Let $\boldsymbol{\xi} = (\xi_1, \cdots, \xi_d)$ be the vector of costs, and α, β, δ be three constants to be specified shortly, such that $\delta \in (0,1)$. Let $t = \beta \ln n$, and $L = O(n^\delta \log n)$.

Preprocessing. We use the same algorithm as before. We sample L sequences $\boldsymbol{M}_1, \ldots, \boldsymbol{M}_L \in \mathcal{B}^t_{\boldsymbol{\xi}/\alpha}$ Then we embed the point set using these projections, setting $P_i = \boldsymbol{M}_i P$, for $i = 1, \ldots, L$. Next, we preprocess the point set P_i for 2-ANN queries under the L_1-norm, and let D_i be the resulting ANN data-structure, for $i = 1, \ldots, L$.

Answering a query. Given a query point $\boldsymbol{q}$, the algorithm performs a 2-ANN query for $\boldsymbol{M}_i\boldsymbol{q}$ in D_i, this ANN corresponds to some original point $\boldsymbol{x}_i \in P$, for $i = 1, \ldots, L$. The algorithm then $(1+\varepsilon)$-approximate the distance $\ell_i = \mathsf{d}_{\boldsymbol{\xi}}(\boldsymbol{q}, \boldsymbol{x}_i)$, for $i = 1, \ldots, L$, using the algorithm of Lemma 4.1.3, and returns the point realizing the minimum distance as the desired ANN.

4.3. The analysis

Lemma 4.3.1. *For a query point $\boldsymbol{q}$, let $B = B(\boldsymbol{q})$ be the bad coordinates of $\boldsymbol{q}$ (thus $\|B\boldsymbol{\xi}\|_1 \le 1$). Then the probability that $\boldsymbol{M} \in \mathcal{B}^t_{\boldsymbol{\xi}/\alpha}$ misses all the coordinates of B is at least $n^{-2\beta/\alpha}$.*

Proof: Let $j_1, \ldots, j_w$ be the bad coordinates in B. We have that $\sum_{i=1}^w \xi_{j_i} \le 1$. As such, the probability that $\boldsymbol{m} \in \mathcal{B}_{\boldsymbol{\xi}/\alpha}$ fails to sample a coordinate in B is $\prod_{i=1}^w (1 - \xi_{j_i}/\alpha) \ge \prod_{i=1}^w \exp(-2\xi_{j_i}/\alpha) \ge \exp(\sum_{i=1}^w -2\xi_{j_i}/\alpha) \ge \exp(-2/\alpha)$, as $\xi_{j_i}/\alpha \le 1/2$, and $1 - x \ge \exp(-2x)$, for $x \in (0, 1/2)$. As such, the probability that a sequence $\boldsymbol{M} = (\boldsymbol{m}_1, \ldots, \boldsymbol{m}_t) \in \mathcal{B}^t_{\boldsymbol{\xi}/\alpha}$ avoids B is at least $\exp(-2t/\alpha) = \exp(-2(\beta/\alpha)\ln n) = n^{-2\beta/\alpha}$. ∎

Let $G = G(\boldsymbol{q})$ and let $r = \|G(\boldsymbol{q} - \boldsymbol{q}^*)\|_1$ be the distance from $\boldsymbol{q}$ to its nearest neighbor only considering the coordinates in G.

Lemma 4.3.2. *For a point $\boldsymbol{x} \in \mathbb{R}^d$, and a random $\boldsymbol{M} \in \mathcal{B}^t_{\boldsymbol{\xi}/\alpha}$, we have that $\mathbf{E}[\|\boldsymbol{M}\boldsymbol{x}\|_1] = t\|\boldsymbol{x}\|_1$, and $\mathbf{V}\left[\|\boldsymbol{M}\boldsymbol{x}\|_1\right] = t\sum_{i=1}^d \boldsymbol{x}_i^2(\alpha/\xi_i - 1)$,*

Proof: The claim on the expectation follows readily from Observation 4.1.4. As for the variance, let i be a coordinate that has non-zero cost, and let X_i be a random variable that is α/ξ_i with probability ξ_i/α, and 0 otherwise. We have that $\mathbf{V}[X_i] = \mathbf{E}[X_i^2] - (\mathbf{E}[X_i])^2 = \left(\frac{\xi_i}{\alpha}(\alpha/\xi_i)^2 + (1 - \xi_i/\alpha)0\right) - 1^2 = \alpha/\xi_i - 1$. As such for $\boldsymbol{m} \in \mathcal{B}^t_{\boldsymbol{\xi}/\alpha}$, we have that $\mathbf{V}\left[\|\boldsymbol{m}\boldsymbol{x}\|_1\right] = \mathbf{V}\left[\sum_{i=1}^d \boldsymbol{x}_i X_i\right] = \sum_{i=1}^d \mathbf{V}[\boldsymbol{x}_i X_i] = \sum_{i=1}^d \boldsymbol{x}_i^2 \mathbf{V}[X_i] = \sum_{i=1}^d \boldsymbol{x}_i^2(\alpha/\xi_i - 1)$ and thus the lemma follows. ∎

As before, we want to avoid giving too much weight to a single coordinate which might have a huge (noisy) value in it. As such, we truncate coordinates that are too large. Here, things become somewhat more subtle, as we have to take into account the probability of a coordinate to be picked.

Definition 4.3.3. For a cost vector $\boldsymbol{\xi} \in \mathbb{R}^d$, a positive number $r > 0$, and a point $\boldsymbol{x} \in \mathbb{R}^d$, let $\mathrm{trc}_{r,\boldsymbol{\xi}}(\boldsymbol{x}) = (x'_1, \ldots, x'_d)$ be the ***truncated*** point, where

$$x'_i = \min\left(|x_i|, \frac{r}{\alpha/\xi_i - 1}\right), \tag{4.4}$$

for $i = 1, \ldots, d$.

The truncation seems a bit strange on the first look, but note that a coordinate i that has a cost ξ_i approaching 0, is going to be truncated to zero by the above. Furthermore, it ensures that a "heavy" point would have a relatively small variance in the norm under the projections we use, as testified by the following easy lemma.

Lemma 4.3.4. *Consider a point $\boldsymbol{x} \in \mathbb{R}^d$, and a random $\boldsymbol{m} \in \mathcal{B}_{\boldsymbol{\xi}/\alpha}$. For, $\boldsymbol{x}' = \mathrm{trc}_{r,\boldsymbol{\xi}}(\boldsymbol{x}) = (x'_1, \ldots, x'_d)$, consider the random variable $X = \|\boldsymbol{m}\boldsymbol{x}'\|_1$. We have that $\mathbf{E}\big[X\big] = \|\boldsymbol{x}'\|_1$, $\mathbf{V}\big[X\big] \le r\,\|\boldsymbol{x}'\|_1$, and $\mathbf{Pr}\big[X \ge \mathbf{E}[X]/2\big] \ge 1 - 4r/\|\boldsymbol{x}'\|_1$.*

Proof: The bound on the expectation is by definition. As for the variance, by Lemma 4.3.2 and Eq. (4.4), we have $\mathbf{V}\Big[\|\boldsymbol{m}\boldsymbol{x}'\|_1\Big] = \sum_{i=1}^{d} (x'_i)^2(\alpha/\xi_i - 1) \le \sum_{i=1}^{d} x'_i \dfrac{r}{\alpha/\xi_i - 1}(\alpha/\xi_i - 1) = r\,\|\boldsymbol{x}'\|_1$.

Let $\mu = \mathbf{E}[X]$, and $\sigma^2 = \mathbf{V}[X]$. By Chebyshev's inequality, we have that

$$\mathbf{Pr}\Big[X \ge \frac{\mu}{2}\Big] \ge 1 - \mathbf{Pr}\left[\Big|\|\boldsymbol{m}\boldsymbol{x}'\|_1 - \mu\Big| \ge \frac{\mu/2}{\sigma}\sigma\right] \ge 1 - \left(\frac{\sigma}{\mu/2}\right)^2 = 1 - 4\frac{\mathbf{V}[X]}{\mu^2}.$$

By the above, we have that $\mathbf{V}[X]/\mu^2 \le r\,\|\boldsymbol{x}'\|_1 / \|\boldsymbol{x}'\|_1^2$. ∎

Definition 4.3.5. For a value $\nu > 0$, a point $\boldsymbol{x}$ is $(r, \boldsymbol{\xi}, \nu)$-***light*** (resp. ***heavy***) if $\|\boldsymbol{x}'\|_1 \le \nu$ (resp. $\|\boldsymbol{x}'\|_1 \ge \nu$), where $\boldsymbol{x}' = \mathrm{trc}_{r,\boldsymbol{\xi}}(\boldsymbol{x})$.

Lemma 4.3.6. *Consider a point $\boldsymbol{x} \in \mathbb{R}^d$ that is $(r, \boldsymbol{\xi}, R)$-heavy, for $R \ge 8r$, and a random $\boldsymbol{M} \in \mathcal{B}^t_{\boldsymbol{\xi}/\alpha}$. Then, we have that $\mathbf{Pr}\big[\|\boldsymbol{M}\boldsymbol{x}\|_1 \ge tR/8\big] \ge 1 - 1/n^{\beta/2}$.*

Proof: Let $\boldsymbol{M} = (\boldsymbol{m}_1, \ldots, \boldsymbol{m}_t)$. Let $X_i = \|\boldsymbol{m}_i\boldsymbol{x}'\|_1$, and let Y_i be an indicator variable that is one if $X_i \ge R/2$. We have, by Lemma 4.3.4, that

$$\begin{aligned}\mathbf{Pr}\Big[Y_i = 1\Big] &= \mathbf{Pr}\left[X_i \ge \frac{R}{2}\right] \ge \mathbf{Pr}\left[\|\boldsymbol{m}_i\boldsymbol{x}'\|_1 \ge \frac{R}{2}\right] \ge \mathbf{Pr}\left[\|\boldsymbol{m}_i\boldsymbol{x}'\|_1 \ge \frac{\mathbf{E}[\|\boldsymbol{m}_i\boldsymbol{x}'\|_1]}{2}\right] \ge 1 - \frac{4r}{\|\boldsymbol{x}'\|_1} \\ &= 1 - \frac{4r}{R} \ge \frac{1}{2},\end{aligned}$$

as $R \ge 8r$. By Chernoff inequality, we have that $\mathbf{Pr}\big[\sum_{i=1}^t Y_i \le t/4\big] \le \exp(-2(t/2)^2/t) = \exp(-t/2) = 1/n^{\beta/2}$ since $t = \beta \ln n$.

In particular, we have that $\|\boldsymbol{M}\boldsymbol{x}\|_1 \ge \sum_{i=1}^t Y_i \frac{R}{2}$, and as such

$$\mathbf{Pr}\big[\|\boldsymbol{M}\boldsymbol{x}\|_1 \ge tR/8\big] \ge \mathbf{Pr}\Big[\sum\nolimits_{i=1}^{t} Y_i \ge t/4\Big] \ge 1 - 1/n^{\beta/2}.$$ ∎

4.4. The result

Theorem 4.4.1. *Let P be a point set in $\mathbb{R}^d$, let $\boldsymbol{\xi} \in [0,1]^d$ be the cost of the coordinates, and let $\varepsilon > 0$ be a parameter. One can build a data-structure, such that given a query point $\boldsymbol{q}$, it can report a robust approximate nearest-neighbor under the costs of $\boldsymbol{\xi}$. Formally, if $\boldsymbol{q}^* = \mathsf{nn}(\boldsymbol{q})$ is the robust nearest-neighbor (see Eq. (4.3)$_{p10}$) when one is allowed to drop coordinates of total cost 1, and its distance to this point is $r = \mathsf{d}_{\boldsymbol{\xi}}(\boldsymbol{q}, \boldsymbol{q}^*)$ (see Eq. (4.2)), then the data-structure returns a point $\boldsymbol{x}$, such that $\boldsymbol{q} - \boldsymbol{x}$ is $(r, \boldsymbol{\xi}, 33(1+\varepsilon)r)$-light (see Definition 4.3.5). The data-structure has the following guarantees:*

(A) The preprocessing time and space is $\widetilde{O}(n^\delta)T_2(n)$, where $T_2(n)$ is the preprocessing time and space needed to build a single data-structure for answering (standard) 2-ANN queries in the L_1-norm for n points in d dimensions.

(B) *The query time is* $O(n^{\delta}Q_2 + n^{\delta}d^4/\varepsilon)$*, where* Q_2 *is the query time of answering* 2-ANN *queries in the above* ANN *data-structures.*

Proof: The proof is similar to Lemma 3.2.10 in the unweighted case. We set $\beta = 4$, $\alpha = 2\beta/\delta$, and $R \geq 32r$. By the same arguments as the unweighted case, and using Lemma 4.3.1, Lemma 4.3.2, and Markov's inequality, with high probability, there exists a data-structure D_i that does not sample any of the bad coordinates $B(\boldsymbol{q})$, and that $\|\boldsymbol{M}_i(\boldsymbol{q}-\boldsymbol{q}^*)\|_1 \leq 2tr$. By Lemma 4.3.6 and union bound, for all the points $\boldsymbol{x}$ such that $\boldsymbol{x}_{\setminus B(\boldsymbol{q})}$ (see Definition 2.1.3) is $(r, \boldsymbol{\xi}, 32r)$-heavy, we have $\|\boldsymbol{M}_i\boldsymbol{x}\|_1 \geq 4tr$. Thus by Lemma 4.1.3 no such point would be retrieved by D_i. Note that since for the reported point $\boldsymbol{x}$, we have that $\boldsymbol{x}_{\setminus B(\boldsymbol{q})}$ is $(r, \boldsymbol{\xi}, 32r)$-light, and that

$$\sum_{i\in B(\boldsymbol{q})} \frac{r}{\alpha/\xi_i - 1} \leq r \sum_{i\in B(\boldsymbol{q})} \frac{\xi_i}{\alpha - \xi_i} \leq \frac{r}{\alpha - 1} \sum_{i\in B(\boldsymbol{q})} \xi_i \leq \frac{r}{\alpha - 1} \leq r,$$

the point $\boldsymbol{x}$ is $(r, \boldsymbol{\xi}, 33r)$-light. Using Lemma 4.1.3 implies an additional $(1+\varepsilon)$ blowup in the computed distance, implying the result. ∎

Corollary 4.4.2. *Under the conditions and notations of Theorem 4.4.1, for the query point* $\boldsymbol{q}$ *and its returned point* $\boldsymbol{x}$*, there exists a set of coordinates* $I \subseteq [\![d]\!]$ *of cost at most* $O(1)$ *(i.e.,* $\sum_{i\in I}\xi_i = O(1)$*, such that* $\|(\boldsymbol{q}-\boldsymbol{x})_{\setminus I}\|_1 \leq O(r)$*. That is, we can remove a set of coordinates of cost at most* $O(1)$ *such that the distance of the reported point* $\boldsymbol{x}$ *from the query* $\boldsymbol{q}$ *is at most* $O(r)$*.*

Proof: Let $\boldsymbol{v} = \boldsymbol{q} - \boldsymbol{x}$ and by Theorem 4.4.1 $\boldsymbol{v}$ is $(r, \boldsymbol{\xi}, c'r))$-light for some constant c'. Let $\boldsymbol{v}' = \mathrm{trc}_{r,\boldsymbol{\xi}}(\boldsymbol{v})$ and let I be the set of coordinates being truncated (i.e., all i such that $|v_i| \geq \frac{r}{\alpha/\xi_i - 1}$). Clearly, the weight of the coordinates not being truncated is at most $\sum_{i\in[\![d]\!]\setminus I} |v_i| \leq \|\boldsymbol{v}'\|_1 \leq c'r$. Also for the coordinates in the set I, we have that $\sum_{i\in I} \frac{r}{\alpha/\xi_i - 1} \leq c'r$. Therefore, $\sum_{i\in I}\xi_i \leq \sum_{i\in I} \frac{\alpha}{\alpha-\xi_i}\cdot\xi_i \leq \alpha c' := c$, assuming that $\alpha \geq 2$, and noting that $0 \leq \xi_i \leq 1$. ∎

5. Application to data sensitive LSH queries

Given a set of n points P and a radius parameter r, in the approximate near neighbor problem, one has to build a data-structure, such that for any given query point $\boldsymbol{q}$, if there exists a point in P that is within distance r of $\boldsymbol{q}$, it reports a point from P which is within distance $r(1+\varepsilon)$ of the query point. In what follows we present a data structure based on our sampling technique whose performance depends on the relative distances of the query from all the points in the data-set.

5.1. Data structure

Input. The input is a set P of n points in the hamming space $\mathbb{H}^d = \{0,1\}^d$, a radius parameter r, and an approximation parameter $\varepsilon > 0$.

Remark. In the spirit of Remark 2.1.6, one can generate the projections in this case directly. Specifically, for any value of i, consider a random projection $\boldsymbol{M} = (m_1, \ldots, m_i) \in \mathcal{D}^i_{1/r}$. Two points $\boldsymbol{x}, \boldsymbol{v} \in \mathbb{H}^d$ ***collide*** under $\boldsymbol{M}$, if $m_j\boldsymbol{x} = m_j\boldsymbol{v}$, for $j = 1, \ldots, i$. Since we care only about collisions (and not distances) for the projected points, we only care what subset of the coordinates are being copied by this projection. That is, we can interpret this projection as being the projection $\cup_j m_j$, which can be sampled directly from $\mathcal{D}_\tau$, where $\tau = 1 - (1-1/r)^i$. As such, computing $\boldsymbol{M}$ and storing it takes $O(d)$ time. Furthermore, for a point $\boldsymbol{x} \in \mathbb{H}^d$, computing $\boldsymbol{M}\boldsymbol{x}$ takes $O(d)$ time, for any projection $\boldsymbol{M} \in \mathcal{D}^i_{1/r}$.

Preprocessing. For $i \geq 1$, let

$$\mathsf{f}_i(\ell) = (1-1/r)^{\ell i} \leq \exp(-\ell i/r) \qquad \text{and} \qquad \mathsf{s}(i) = \frac{\mathsf{c}\log n}{\mathsf{f}_i(r)} = O\big(e^i \log n\big), \tag{5.1}$$

where c is a sufficiently large constant. Here, f_i is the *collision probability function* of two points at distance ℓ under projection $m \in \mathcal{D}^i_{1/r}$, and $\mathsf{s}(i)$ is the number times one has to repeat an experiment with success probability $\mathsf{f}_i(r)$ till it succeeds with high probability. Let $\mathcal{N} = \frac{\ln n}{1+\varepsilon}$.

For $i = 1, \ldots, \mathcal{N}$, compute a set $\mathcal{H}_i = \left\{ m_1, m_2, \ldots, m_{\mathsf{s}(i)} \in \mathcal{D}^i_{1/r} \right\}$ of projections. For each projection $m \in \mathcal{H}_i$, we compute the set mP and store it in a hash table dedicated to the projection m. Thus, given a query point $\boldsymbol{q} \in \mathbb{H}^d$, the set of points *colliding* with $\boldsymbol{q}$ is the set $\mathcal{C}_m(\boldsymbol{q}) = \{\boldsymbol{x} \in P \mid m\boldsymbol{q} = m\boldsymbol{x}\}$, stored as a linked list, with a single entry in the hash table of m. Given $\boldsymbol{q}$, one can extract, using the hash table, in $O(d)$ time, the list representing $\mathcal{C}_m(\boldsymbol{q})$. More importantly, in $O(d)$ time, one can retrieve the size of this list; that is, the number $|\mathcal{C}_m(\boldsymbol{q})|$. For any $i \leq \mathcal{N}$, let DS_i denote the constructed data-structure.

Query. Given a query point $\boldsymbol{q} \in \mathbb{H}^d$, the algorithm starts with $i = 1$, and computes, the number of points colliding with it in $\mathcal{H}_i$. Formally, this is the number $X_i = \sum_{m \in \mathcal{H}_i} |\mathcal{C}_m(\boldsymbol{q})|$. If $X_i > \mathsf{c}_1\mathsf{s}(i)$, the algorithm increases i, and continues to the next iteration, where c_1 is any constant strictly larger than e.

Otherwise, $X_i \leq \mathsf{c}_1\mathsf{s}(i)$ and the algorithm extracts from the $\mathsf{s}(i)$ hash tables (for the projections of $\mathcal{H}_i$) the lists of these X_i points, scans them, and returns the closest point encountered in these lists.

The only remaining situation is when the algorithm had reached the last data-structure for $\mathcal{H}_{\mathcal{N}}$ without success. The algorithm then extracts the collision lists as before, and it scans the lists, stopping as soon as a point of distance $\leq (1+\varepsilon)r$ had been encountered. In this case, the scanning has to be somewhat more careful – the algorithm breaks the set of projections of $\mathcal{H}_{\mathcal{N}}$ into $T = \mathsf{c}\log n$ blocks $B_1, \ldots, B_T$, each containing $1/\mathsf{f}_{\mathcal{N}}(r)$ projections, see Eq. (5.1). The algorithm computes the total size of the collision lists for each block, separately, and sort the blocks in increasing order by the number of their collisions. The algorithm now scans the collision lists of the blocks in this order, with the same stopping condition as above.

Remark 5.1.2. There are various modifications one can do to the above algorithm to improve its performance in practice. For example, when the algorithm retrieves the length of a collision list, it can also retrieve some random element in this list, and compute its distance to $\boldsymbol{q}$, and if this distance is smaller than $(1+\varepsilon)r$, the algorithm can terminate and return this point as the desired approximate near-neighbor. However, the advantage of the variant presented above, is that there are many scenarios where it would return the *exact* nearest-neighbor. See below for details.

5.2. Analysis

5.2.1. Intuition

The expected number of collisions with the query point $\boldsymbol{q}$, for a single $m \in \mathcal{H}_i$, is

$$\gamma_i = \mathbf{E}\Big[|\mathcal{C}_m(\boldsymbol{q})|\Big] = \sum_{\boldsymbol{x} \in P} \mathsf{f}_i\big(\mathrm{d}_H(\boldsymbol{q}, \boldsymbol{x})\big) \leq \sum_{\boldsymbol{x} \in P} \exp\big(-\mathrm{d}_H(\boldsymbol{q}, \boldsymbol{x})i/r\big). \tag{5.2}$$

This quantity can be interpreted as a convolution over the point set. Observe that as $\mathsf{f}_i(\ell)$ is a monotonically decreasing function of i (for a fixed ℓ), we have that $\gamma_1 \geq \gamma_2 \geq \cdots$.

The expected number of collisions with $\boldsymbol{q}$, for all the projections of $\mathcal{H}_i$, is

$$\Gamma_i = \mathbf{E}[X_i] = \mathsf{s}(i)\gamma_i. \tag{5.3}$$

If we were to be naive, and just scan the lists in the ith level, the query time would be $O\big(d(X_i + \mathsf{s}(i))\big)$. As such, if $X_i = O(\mathsf{s}(i))$, then we are "happy" since the query time is small. Of course, a priori it is not clear whether X_i (or, more specifically, Γ_i) is small.

Intuitively, the higher the value i is, the stronger the data-structure "pushes" points away from the query point. If we are lucky, and the nearest neighbor point is close, and the other points are far, then we would need to push very little, to get X_i which is relatively small, and get a fast query time. The standard LSH analysis works according to the worst case scenario, where one ends up in the last layer $\mathcal{H}_{\mathcal{N}}$.

Example 5.2.1 (If the data is locally low dimensional). The quantity Γ_i depends on how the data looks like near the query. For example, assume that locally near the query point, the point set looks like a uniform low dimensional point set. Specifically, assume that the number of points in distance ℓ from the query is bounded by $\#(\ell) = O\big((\ell/r)^k\big)$, where k is some small constant and r is the distance of the nearest-neighbor to $\boldsymbol{q}$. We then have that

$$\Gamma_i = \mathsf{s}(i) \sum_{\boldsymbol{x} \in P} \mathsf{f}_i\big(\mathrm{d}_H(\boldsymbol{q}, \boldsymbol{x})\big) \leq O(e^i \log n) \sum_{j=1}^{\infty} \#(jr) \exp(-ji) = \sum_{j=1}^{\infty} O\big(j^k \exp(i - ji) \ln n\big).$$

By setting $i = k$, we have

$$\Gamma_k \leq \sum_{j=1}^{\infty} O\left(\left(\frac{j}{e^{j-1}}\right)^k \ln n\right) \leq \sum_{=1}^{\infty} O\left(\frac{j}{2^j} \ln n\right) \leq O(\ln n)$$

Therefore, the algorithm would stop in expectation after $O(k)$ rounds.

Namely, if the data near the query point locally behaves like a low dimensional uniform set, then the expected query time is going to be $O(d \log n)$, where the constant depends on the data-dimension k.

5.2.2. Analysis

Lemma 5.2.2. *If there exists a point within distance r of the query point, then the algorithm would compute, with high probability, a point $\boldsymbol{x}$ which is within distance $(1+\varepsilon)r$ of the query point.*

Proof: Let $\boldsymbol{q}^* \in P$ be the nearest neighbor to the query $\boldsymbol{q}$. For any data-structure DS_i, the probability that $\boldsymbol{q}^*$ does not collide with $\boldsymbol{q}$ is at most $(1 - \mathsf{f}_i(r))^{\mathsf{s}(i)} \leq \exp(-\mathsf{f}_i(r)\mathsf{s}(i)) = \exp(-O(\log n)) \leq 1/n^{O(1)}$. Since the algorithm ultimately stops in one of these data-structures, and scans all the points colliding with the query point, this implies that the algorithm, with high probability, returns a point that is in distance $\leq (1+\varepsilon)r$. ∎

Remark 5.2.3. An interesting consequence of Lemma 5.2.2 is that if the data-structure stops before it arrives to $\mathsf{DS}_{\mathcal{N}}$, then it returns the *exact* nearest-neighbor – since the data-structure accepts approximation only in the last level. Stating it somewhat differently, only if the data-structure gets overwhelmed with collisions it returns an approximate answer.

Remark 5.2.4. One can duplicate the coordinates $O(c)$ times, such that the original distance r becomes cr. In particular, this can be simulated on the data-structure directly without effecting the performance. As such, in the following, it is safe to assume that r is a sufficiently large – say larger than $\log n \geq \mathcal{N}$.

Lemma 5.2.5. *For any $i > 0$, we have $\mathsf{s}(i) = O(e^i \log n)$.*

Proof: We have that $\mathsf{f}_i(r) = (1-1/r)^{(r-1)ri/(r-1)} \geq \exp(-ri/(r-1)) = \exp(-i - i/(r-1))$ since for any positive integer m, we have $(1-1/m)^{m-1} \geq 1/e \geq (1-1/m)^m$. As such, since we can assume that $r > \log n \geq i$, we have that $\mathsf{f}_i(r) = \Theta(\exp(-i))$. Now, we have $\mathsf{s}(i) = O(\frac{\log n}{\mathsf{f}_i(r)}) = O(e^i \log n)$. ∎

Lemma 5.2.6. *For a query point $\boldsymbol{q}$, the worst case query time is $O(dn^{1/(1+\varepsilon)} \log n)$, with high probability.*

Proof: The worst query time is realized when the data-structure scans the points colliding under the functions of $\mathcal{H}_{\mathcal{N}}$.

We partition P into two point sets:

(i) The close points are $P_{\leq} = \{\boldsymbol{x} \in P \mid \mathrm{d}_H(\boldsymbol{q}, \boldsymbol{x}) \leq (1+\varepsilon)r\}$, and
(ii) the far points are $P_{>} = \{\boldsymbol{x} \in P \mid \mathrm{d}_H(\boldsymbol{q}, \boldsymbol{x}) > (1+\varepsilon)r\}$.

Any collision with a point of $P_{\leq}$ during the scan terminates the algorithm execution, and is thus a *good* collision. A *bad* collision is when the colliding point belongs to $P_{>}$.

Let $B_1, \ldots, B_T$ be the partition of the projections of $\mathcal{H}_{\mathcal{N}}$ into blocks used by the algorithm. For any j, we have $|B_j| = O(\mathsf{s}(\mathcal{N})/\log n) = O\big(e^{\mathcal{N}}\big) = O\big(n^{1/(1+\varepsilon)}\big)$, since $\mathcal{N} = (\ln n)/(1+\varepsilon)$ and by Lemma 5.2.5. Such a block, has probability of $p = (1 - \mathsf{f}_{\mathcal{N}}(r))^{1/\mathsf{f}_{\mathcal{N}}(r)} \leq 1/e$ to not have the nearest-neighbor to $\boldsymbol{q}$ (i.e., $\boldsymbol{q}^*$) in its collision lists. If this event happens, we refer to the block as being *useless*.

For a block B_j, let Y_j be the total size of the collision lists of $\boldsymbol{q}$ for the projections of B_j when ignoring good collisions altogether. We have that

$$\mathbf{E}[Y_j] \leq |B_j| \cdot |P_{>}| \cdot \mathsf{f}_{\mathcal{N}}\big((1+\varepsilon)r\big) \leq |B_j|\, n \exp(-(1+\varepsilon)r\mathcal{N}/r) = |B_j|\, n \exp(-(1+\varepsilon)\mathcal{N}) = |B_j| = O\big(n^{1/(1+\varepsilon)}\big),$$

since $\mathcal{N} = (\ln n)/(1+\varepsilon)$. In particular, the jth block is *heavy*, if $Y_i \geq 10\,|B_i|$. The probability for a block to be heavy, is $\leq 1/10$, by Markov's inequality.

In particular, the probability that a block is heavy or useless, is at most $1/e + 1/10 \leq 1/2$. As such, with high probability, there is a light and useful block. Since the algorithm scans the blocks by their collision lists size, it follows that with high probability, the algorithm scans only light blocks before it stops the scanning, which is caused by getting to a point that belongs to $P_{\leq}$. As such, the query time of the algorithm is $O(dn^{1/(1+\varepsilon)} \log n)$. ∎

Next, we analyze the data-dependent running time of the algorithm.

Lemma 5.2.7. *Let $\ell_i = \mathrm{d}_H(\boldsymbol{q}, \boldsymbol{x}_i)$, for $i = 1, \ldots, n$, where $P = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\} \subseteq \mathbb{H}^d$. Let Δ be the smallest value such that $\sum_{i=1}^{n} \exp(-\Delta \ell_i/r) \leq 1$. Then, the expected query time of the algorithm is $O\big(de^{\Delta} \log n\big)$.*

Proof: The above condition implies that $\gamma_j \leq \gamma_{\Delta} \leq 1$, for any $j \geq \Delta$. By Eq. (5.3), for $j \geq \Delta$, we have that $\Gamma_j = \mathbf{E}[X_j] = \gamma_j \mathsf{s}(j) \leq \mathsf{s}(j)$. Thus, by Markov's inequality, with probability at least $1 - 1/\mathsf{c}_1$, we have that $X_j \leq \mathsf{c}_1 \mathsf{s}(j)$, and the algorithm would terminate in this iteration. As such, let Y_j be an indicator variable that is one of the algorithm reached the jth iteration. However, for that to happen, the algorithm has to fail in iterations $\Delta, \Delta+1, \ldots, j-1$. As such, we have that $\mathbf{Pr}[Y_j = 1] = 1/{\mathsf{c}_1}^{j-\Delta}$.

The jth iteration of the algorithm, if it happens, takes $O(d\mathsf{s}(j))$ time, and as such, the overall expected running time is proportional to $\mathbf{E}\big[d \sum_{j=1}^{\mathcal{N}} Y_j \mathsf{s}(j)\big]$. Namely, the expected running time is bounded by

$$O\left(d \sum_{j=1}^{\Delta} \mathsf{s}(j) + d \sum_{j=\Delta+1}^{\mathcal{N}} \frac{\mathsf{s}(j)}{(\mathsf{c}_1)^{j-\Delta}}\right) = O\left(de^{\Delta} \log n + de^{\Delta} \sum_{j=\Delta+1}^{\mathcal{N}} \frac{e^{j-\Delta} \log n}{(\mathsf{c}_1)^{j-\Delta}}\right) = O\big(de^{\Delta} \log n\big),$$

using the bound $\mathsf{s}(j) = O(e^j \log n)$ from Lemma 5.2.5, and since $\mathsf{c}_1 > e$. ∎

5.3. The result

Theorem 5.3.1. *Given a set $P \subseteq \mathbb{H}^d = \{0,1\}^d$ of n points, and parameters $\varepsilon > 0$ and r, one can preprocess the point set, in $O\big(dn^{1+1/(1+\varepsilon)} \log n\big)$ time and space, such that given a query point $\boldsymbol{q} \in \mathbb{H}^d$, one can decide (approximately) if $\mathrm{d}_H(\boldsymbol{q}, P) \leq r$, in $O\big(dn^{1/(1+\varepsilon)} \log n\big)$ expected time, where d_H is the Hamming distance. Formally, the data-structure returns, either:*

- *"$\mathrm{d}_H(\boldsymbol{q}, P) \leq (1+\varepsilon)r$", and the data-structure returns a witness $\boldsymbol{x} \in P$, such that $\mathrm{d}_H(\boldsymbol{q}, \boldsymbol{x}) \leq (1+\varepsilon)r$. This is the result returned if $\mathrm{d}_H(\boldsymbol{q}, \boldsymbol{x}) \leq r$.*
- *"$\mathrm{d}_H(\boldsymbol{q}, P) > (1+\varepsilon)r$", and this is the result returned if $\mathrm{d}_H(\boldsymbol{q}, P) > (1+\varepsilon)r$.*

The data-structure is allowed to return either answer if $r \leq \mathrm{d}_H(\boldsymbol{q}, P) \leq (1+\varepsilon)r$. The query returns the correct answer, with high probability.

Furthermore, if the query is "easy", the data-structure would return the exact *nearest neighbor. Specifically, if $\mathrm{d}_H(\boldsymbol{q}, P) \leq r$, and there exists $\Delta < (\ln n)/(1+\varepsilon)$, such that $\sum_{\boldsymbol{x} \in P} \exp(-\mathrm{d}_H(\boldsymbol{q}, \boldsymbol{x})\Delta/r) \leq 1$, then the data-structure would return the exact nearest-neighbor in $O\big(d \exp(\Delta) \log n\big)$ expected time.*

Remark 5.3.2. If the data is k dimensional, in the sense of having bounded growth (see Example 5.2.1), then the above data-structure solves approximate LSH in $O(d \log n)$ time, where the constant hidden in the O depends (exponentially) on the data dimension k.

This result is known, see Datar *et al.* [DIIM04, Appendix A]. However, our data-structure is more general, as it handles this case with no modification, while the data-structure of Datar *et al.* is specialized for this case.

Remark 5.3.3. Fine tuning the LSH scheme to the hardness of the given data is not a new idea. In particular, Andoni *et al.* [ADI+06, Section 4.3.1] suggest fine tuning the LSH construction parameters for the set of queries, to optimize the overall query time.

Contrast this with the new data-structure of Theorem 5.3.1, which, conceptually, adapts the parameters on the fly during the query process, depending on how hard the query is.

6. Conclusions

Ultimately, our data-structure is a prisoner of our underlying technique of sampling coordinates. Thus, the main challenge is to come up with a different approach that does not necessarily rely on such an idea. In particular, our current technique does not work well for points that are sparse, and have only few non-zero coordinates. We believe that this problem should provide fertile ground for further research.

Acknowledgments. The authors thank Piotr Indyk for insightful discussions about the problem and also for the helpful comments on the presentation of this paper. The authors also thank Jen Gong, Stefanie Jegelka, and Amin Sadeghi for useful discussions on the applications of this problem.

References

[AC06] N. Ailon and B. Chazelle. *Approximate nearest neighbors and the fast Johnson-Lindenstrauss transform*. *Proc. 38th Annu. ACM Sympos. Theory Comput.* (STOC), 557–563, 2006.

[ADI+06] A. Andoni, M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. *Locality-sensitive hashing using stable distributions*. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. Ed. by T. Darrell, P. Indyk, and G. Shakhnarovich. MIT Press, 2006, pp. 61–72.

[AI08] A. Andoni and P. Indyk. *Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions*. *Commun. ACM* 51.1 (2008), pp. 117–122.

[AIKN09] A. Andoni, P. Indyk, R. Krauthgamer, and H. L. Nguyen. *Approximate line nearest neighbor in high dimensions*. *Proc. 20th ACM-SIAM Sympos. Discrete Algs.* (SODA), 293–301, 2009.

[AINR14] A. Andoni, P. Indyk, H. L. Nguyen, and I. Razenshteyn. *Beyond Locality-Sensitive Hashing*. *Proc. 25th ACM-SIAM Sympos. Discrete Algs.* (SODA), 1018–1028, 2014.

[AMN+98] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. *An optimal algorithm for approximate nearest neighbor searching in fixed dimensions*. *J. Assoc. Comput. Mach.* 45.6 (1998), pp. 891–923.

[AR15] A. Andoni and I. Razenshteyn. *Optimal Data-Dependent Hashing for Approximate Near Neighbors*. *Proc. 47th Annu. ACM Sympos. Theory Comput.* (STOC), 793–801, 2015.

[CFV+13] F. Cismondi, A. S. Fialho, S. M. Vieira, S. R. Reti, J. M. C. Sousa, and S. N. Finkelstein. *Missing data in medical databases: Impute, delete or classify?* *AI in Medicine* 58.1 (2013), pp. 63–72.

[CLMW11] E. J. Cands, X. Li, Y. Ma, and J. Wright. *Robust principal component analysis?* *J. Assoc. Comput. Mach.* 58.3 (2011), p. 11.

[CR10] A. Chakrabarti and O. Regev. *An Optimal Randomized Cell Probe Lower Bound for Approximate Nearest Neighbor Searching*. *SIAM J. Comput.* 39.5 (Feb. 2010), pp. 1919–1940.

[CS89] K. L. Clarkson and P. W. Shor. *Applications of random sampling in computational geometry, II*. *Discrete Comput. Geom.* 4 (1989), pp. 387–421.

[DIIM04] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. *Locality-sensitive hashing scheme based on p-stable distributions*. *Proc. 20th Annu. Sympos. Comput. Geom.* (SoCG), 253–262, 2004.

[Har01] S. Har-Peled. *A Replacement for Voronoi Diagrams of Near Linear Size*. *Proc. 42nd Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), 94–103, 2001.

[HE07] J. Hays and A. A. Efros. *Scene completion using millions of photographs*. *Trans. Graphics* 26.3 (2007), p. 4.

[HIM12] S. Har-Peled, P. Indyk, and R. Motwani. *Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality*. *Theory Comput.* 8 (2012). Special issue in honor of Rajeev Motwani, pp. 321–350.

[IM98] P. Indyk and R. Motwani. *Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality*. *Proc. 30th Annu. ACM Sympos. Theory Comput.* (STOC), 604–613, 1998.

[Ind04] P. Indyk. *Nearest Neighbors in High-dimensional Spaces*. *Handbook of Discrete and Computational Geometry*. Ed. by J. E. Goodman and J. O'Rourke. 2nd. CRC Press LLC, 2004. Chap. 39, pp. 877–892.

[Ind06] P. Indyk. *Stable distributions, pseudorandom generators, embeddings, and data stream computation*. *J. Assoc. Comput. Mach.* 53.3 (2006), pp. 307–323.

[Isl09] M. T. Islam. *Approximation algorithms for minimum knapsack problem*. https://www.uleth.ca/dspace/handle/10133/1304. MA thesis. Dept. Math & Comp. Sci., Univ. Lethbridge, 2009.

[KL04] R. Krauthgamer and J. R. Lee. *Navigating nets: simple algorithms for proximity search*. *Proc. 15th ACM-SIAM Sympos. Discrete Algs.* (SODA), 798–807, 2004.

[Kle97] J. Kleinberg. *Two algorithms for nearest-neighbor search in high dimensions*. *Proc. 29th Annu. ACM Sympos. Theory Comput.* (STOC), 599–608, 1997.

[KOR00] E. Kushilevitz, R. Ostrovsky, and Y. Rabani. *Efficient Search for Approximate Nearest Neighbor in High Dimensional Spaces*. *SIAM J. Comput.* 2.30 (2000), pp. 457–474.

[Mah15] S. Mahabadi. *Approximate Nearest Line Search in High Dimensions*. *Proc. 26th ACM-SIAM Sympos. Discrete Algs.* (SODA), 337–354, 2015.

[MNSS15] W. Mulzer, H. L. Nguyn, P. Seiferth, and Y. Stein. *Approximate k-flat Nearest Neighbor Search*. *Proc. 47th Annu. ACM Sympos. Theory Comput.* (STOC), 783–792, 2015.

[Pan06] R. Panigrahy. *Entropy based nearest neighbor search in high dimensions*. *Proc. 17th ACM-SIAM Sympos. Discrete Algs.* (SODA), 1186–1195, 2006.

[Sam05] H. Samet. *Foundations of Multidimensional and Metric Data Structures*. The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling. Morgan Kaufmann Publishers Inc., 2005.

[SDI06] G. Shakhnarovich, T. Darrell, and P. Indyk. *Nearest-Neighbor Methods in Learning and Vision: Theory and Practice*. Neur. Info. Proc. The MIT Press, 2006.

[SWC+09] J. A. C. Sterne, I. R. White, J. B. Carlin, M. Spratt, P. Royston, M. G. Kenward, A. M. Wood, and J. R. Carpenter. *Multiple imputation for missing data in epidemiological and clinical research: potential and pitfalls*. *BMJ* 338 (2009), b2393.

[WCNK13] B. J. Wells, K. M. Chagin, A. S. Nowacki, and M. W. Kattan. *Strategies for handling missing data in electronic health record derived data*. *eGEMs* 1.3 (2013).

A. For the L_1-case: Trading density for proximity

The tail of a point in Lemma 3.2.10 has to be quite heavy, for the data-structure to reject it as an ANN. It is thus natural to ask if one can do better, that is – classify a far point as far, even if the threshold for being far is much smaller (i.e., ultimately a factor of $1+\varepsilon$). Maybe surprisingly, this can be done, but it requires that such a far point would be quite dense, and we show how to do so here. For the sake of simplicity of exposition the result of this section is provided only under the L_1 norm.

The algorithm is the same as the one presented in Section 3.1, except that for the given parameter ϵ we use $(1+O(\epsilon))$-ANN data-structures. We will specify it more precisely at the end of this section. Also the total number of ANN data-structures is $L = O\big(n^{\delta} \cdot \log \frac{n}{\varepsilon}\big)$.

A.1. A tail of two points

We start by proving a more nuanced version of Lemma 3.2.6.

Lemma A.1.1. *Let $\xi, \varepsilon \in (0,1)$ be parameters, and let $\boldsymbol{x}$ be a point in $\mathbb{R}^d$ that is $(\xi r, (1+\varepsilon)r)$-heavy. Then* $\mathbf{Pr}\Big[\|\boldsymbol{m}\boldsymbol{x}\|_1 \geq (1+\varepsilon/4)\tau r\Big] \geq 1 - \dfrac{9\xi}{\varepsilon^2\tau}$,

Proof: Let $\psi = \xi r$ be a parameter and consider the ψ-truncated point $\boldsymbol{v} = \boldsymbol{x}_{\leq \psi}$. Since $\boldsymbol{x}$ is $(\xi r, (1+\varepsilon)r)$-heavy, we have that $\|\boldsymbol{x}\|_1 \geq \|\boldsymbol{v}\|_1 \geq (1+\varepsilon)r$. Now, we have

$$\mu = \mathbf{E}\Big[\|m\boldsymbol{v}\|_1\Big] = \tau \|\boldsymbol{v}\|_1 \geq (1+\varepsilon)\tau r \qquad \text{and} \qquad \sigma^2 = \mathbf{V}\Big[\|m\boldsymbol{v}\|_1\Big] = \tau(1-\tau)\|\boldsymbol{v}\|_2^2 .$$

Now, by Chebyshev's inequality, we have that

$$\begin{aligned}
\mathbf{Pr}\Big[\|m\boldsymbol{x}\|_1 \geq (1+\varepsilon/4)\tau r\Big] &\geq \mathbf{Pr}\Big[\|m\boldsymbol{v}\|_1 \geq (1+\varepsilon/4)\tau r\Big] \geq \mathbf{Pr}\left[\|m\boldsymbol{v}\|_1 \geq \frac{(1+\varepsilon)\tau r}{1+\varepsilon/2}\right] \\
&\geq \mathbf{Pr}\left[\|m\boldsymbol{v}\|_1 \geq \frac{\mu}{1+\varepsilon/2}\right] = \mathbf{Pr}\left[\|m\boldsymbol{v}\|_1 - \mu \geq \left(\frac{1}{1+\varepsilon/2} - 1\right)\mu\right] \\
&= \mathbf{Pr}\left[-\|m\boldsymbol{v}\|_1 + \mu \leq \frac{\varepsilon/2}{1+\varepsilon/2}\mu\right] = 1 - \mathbf{Pr}\left[-\|m\boldsymbol{v}\|_1 + \mu \geq \frac{\varepsilon}{2+\varepsilon}\mu\right] \\
&\geq 1 - \mathbf{Pr}\left[\Big|\|m\boldsymbol{v}\|_1 - \mu\Big| \geq \frac{\varepsilon}{2+\varepsilon}\cdot\frac{\mu}{\sigma}\cdot\sigma\right] \geq 1 - \left(\frac{2+\varepsilon}{\varepsilon}\right)^2\left(\frac{\sigma}{\mu}\right)^2 \\
&= 1 - \frac{9}{\varepsilon^2}\cdot\frac{\tau(1-\tau)\|\boldsymbol{v}\|_2^2}{\tau^2\|\boldsymbol{v}\|_1^2} \geq 1 - \frac{9}{\varepsilon^2}\frac{(1-\tau)\xi}{\tau(1+\varepsilon)} \geq 1 - \frac{9\xi}{\varepsilon^2\tau},
\end{aligned}$$

by Lemma 3.2.5. Now by setting $\xi \leq \varepsilon^3\tau/90$, this probability would be at least $1-\varepsilon/10$. ∎

Similar to the Lemma 3.2.8 by Markov's inequality we have the following lemma.

Lemma A.1.2. *Consider a point $\boldsymbol{x}$ such that $\|\boldsymbol{x}_{\setminus k}\|_1 \leq r$. Conditioned on the event of Lemma 3.2.7, we have that $\mathbf{Pr}\Big[\|\boldsymbol{M}\boldsymbol{x}\|_1 \leq (1+\varepsilon/32)t\tau r\Big] \geq 1 - \frac{1}{1+\varepsilon/32} \geq \varepsilon/33$, where $\boldsymbol{M} \in \mathcal{D}_\tau^t$.*

Lemma A.1.3. *Let $U = (1+\varepsilon/16)t\tau r$. If $\boldsymbol{x}$ is $(\xi r, R)$-heavy point, then $\mathbf{Pr}\Big[\|\boldsymbol{M}\boldsymbol{x}\|_1 \geq U\Big] \geq 1 - 2n^{-\beta\varepsilon^2/256}$, assuming $R \geq (1+\varepsilon)r$ and $\xi \leq \frac{\varepsilon^3}{90(\alpha k)}$.*

Proof: For all i, let $Y_i = \|m_i\boldsymbol{x}\|_1$. By Lemma A.1.1, with probability at least $(1-\varepsilon/10)$, we have that $Y_i \geq (1+\varepsilon/4)\tau r$. In particular, let $Z_i = \min(Y_i, (1+\varepsilon/4)\tau r)$. We have that

$$\mu = \mathbf{E}[Z] = \mathbf{E}\left[\sum_{i=1}^{t} Z_i\right] \geq (1+\varepsilon/4)t\tau r(1-\varepsilon/10) \geq (1+\varepsilon/8)t\tau r.$$

Now, by Hoeffding's inequality, we have that

$$\begin{aligned}
\mathbf{Pr}\Big[\|\boldsymbol{M}\boldsymbol{x}\|_1 \leq U\Big] \leq \mathbf{Pr}\Big[Z \leq U\Big] \leq \mathbf{Pr}\Big[|Z-\mu| \geq \frac{\varepsilon}{16}t\tau r\Big] &\leq 2\exp\left(-\frac{2(t\tau r\varepsilon/16)^2}{t((1+\varepsilon/4)\tau r)^2}\right) \\
&\leq 2\exp\left(-\frac{t\varepsilon^2}{16^2}\right) \leq \frac{2}{n^{\beta\varepsilon^2/256}}.
\end{aligned}$$

∎

Lemma A.1.4. *Let $\delta, \varepsilon \in (0,1)$ be two parameters. For a query point $\boldsymbol{q} \in \mathbb{R}^d$, let $\boldsymbol{n} \in P$ be its k-fold nearest neighbor in P, and let $r = \|(\boldsymbol{n}-\boldsymbol{q})_{\setminus k}\|_1$. Then, with high probability, the algorithm returns a point $\boldsymbol{v} \in P$, such that $\boldsymbol{q}-\boldsymbol{v}$ is a $(\xi r, r(1+2\varepsilon))$-light, where $\xi = O(\delta\varepsilon^5/k)$. The data-structure performs $O\big(n^\delta \cdot \frac{\log n}{\varepsilon}\big)$ of $(1+\varepsilon/64)$-ANN queries.*

Proof: As before, we set $\beta = 512/\varepsilon^2$ and $\alpha = \beta/\delta$. Also, by conditions of Lemma A.1.3 we have $\xi \leq \varepsilon^3/(90 \cdot \alpha k) = \varepsilon^3\delta/(90\beta k) = \varepsilon^5\delta/(90 \cdot 512 \cdot k)$. Also, let $L = (n^\delta \cdot \frac{\log n}{\varepsilon})$. Let S be the set of k largest coordinates in $\boldsymbol{z} = \boldsymbol{q} - \boldsymbol{n}$. By similar arguments as in Lemma 3.2.10, there exists $m = \Omega(\frac{\log n}{\varepsilon})$ data structures say $D_1, \ldots, D_m$ such that $\boldsymbol{M}_i \in \mathcal{D}^t_\tau$ does not contain any of the coordinates of S.

Since by assumption $\|\boldsymbol{z}_{\setminus S}\|_1 = \|\boldsymbol{z}_{\setminus\setminus k}\|_1 \leq r$, and by Lemma A.1.2, with probability at least $\varepsilon/33$ the distance of $\boldsymbol{M}_i\boldsymbol{n}$ from $\boldsymbol{M}_i\boldsymbol{q}$ is at most $(1+\varepsilon/32)t\tau r$. Since there are $\Omega(\frac{\log n}{\varepsilon})$ such structures, we know that, with high probability, for one of them, say D_1, this holds. By Lemma A.1.3, any point $\boldsymbol{x}_{\setminus S}$ (of P) that is $(\xi r, (1+\varepsilon)r)$-heavy, would be in distance at least $(1+\varepsilon/16)t\tau r$ in the projection $\boldsymbol{M}_1$ from the projected $\boldsymbol{q}$, and since D_1 is a $(1+\varepsilon/64)$-ANN data-structure under the L_1 norm, we conclude that no such point can be returned. Note that since for the reported point $\boldsymbol{v}$, the point $\boldsymbol{v}_{\setminus S}$ cannot be $(\xi r, r(1+\varepsilon))$-heavy, and the coordinates in S can contribute at most $k\xi r \leq \delta\varepsilon^5 r \leq \varepsilon r$, the point $\boldsymbol{v}$ cannot be $(\xi r, r(1+2\varepsilon))$-heavy. Thus, the data-structure returns the desired point with high probability.

As for the query performance, the data-structure performs L queries of $(1+\varepsilon/64)$-ANN data-structures. ∎

By Lemma 3.2.3, we get the following corollary.

Corollary A.1.5. *Given a query point $\boldsymbol{q} \in \mathbb{R}^d$, let $\boldsymbol{n} \in P$ be its k-robust nearest neighbor in P, and $r = \|(\boldsymbol{n}-\boldsymbol{q})_{\setminus\setminus k}\|_1$. Then, with high probability, the algorithm returns a point $\boldsymbol{v} \in P$, such that $\left\|(\boldsymbol{q}-\boldsymbol{v})_{\setminus\setminus O(\frac{k}{\varepsilon^5\delta})}\right\|_1 \leq r(1+2\varepsilon)$.*