TheoretiCS

Terminal Embeddings in Sublinear Time

Received Mar 4, 2022
Revised Nov 16, 2023
Accepted Jan 28, 2024
Published Mar 14, 2024

Key words and phrases
dimensionality reduction, terminal embeddings, adaptive data structures, nearest neighbor search, sublinear algorithms

Yeshwanth Cherapanamjeri [a]
Jelani Nelson [b]

a CSAIL, M.I.T

b UC Berkeley

ABSTRACT. Recently (Elkin, Filtser, Neiman 2017) introduced the concept of a *terminal embedding* from one metric space (X, d_X) to another (Y, d_Y) with a set of designated terminals $T \subset X$. Such an embedding f is said to have distortion $\rho \geqslant 1$ if ρ is the smallest value such that there exists a constant $C > 0$ satisfying

$$\forall x \in T\ \forall q \in X,\ C d_X(x, q) \leqslant d_Y(f(x), f(q)) \leqslant C\rho d_X(x, q).$$

When X, Y are both Euclidean metrics with Y being m-dimensional, recently (Narayanan, Nelson 2019), following work of (Mahabadi, Makarychev, Makarychev, Razenshteyn 2018), showed that distortion $1 + \varepsilon$ is achievable via such a terminal embedding with $m = O(\varepsilon^{-2} \log n)$ for $n := |T|$. This generalizes the Johnson-Lindenstrauss lemma, which only preserves distances within T and not to T from the rest of space. The downside of prior work is that evaluating their embedding on some $q \in \mathbb{R}^d$ required solving a semidefinite program with $\Theta(n)$ constraints in m variables and thus required some superlinear $\text{poly}(n)$ runtime. Our main contribution in this work is to give a new data structure for computing terminal embeddings. We show how to pre-process T to obtain an almost linear-space data structure that supports computing the terminal embedding image of any $q \in \mathbb{R}^d$ in sublinear time $O^*(n^{1-\Theta(\varepsilon^2)} + d)$. To accomplish this, we leverage tools developed in the context of approximate nearest neighbor search.

A preliminary version of this article appeared at FOCS 2021. The first author was supported by a Microsoft Research BAIR Commons Research Grant. The second author was supported by NSF award CCF-1951384, ONR grant N00014-18-1-2562, ONR DORECG award N00014-17-1-2127, and a Google Faculty Research Award.

Cite as Yeshwanth Cherapanamjeri, Jelani Nelson. Terminal Embeddings in Sublinear Time. TheoretiCS, Volume 3 (2024), Article 6, 1-52.

https://theoretics.episciences.org
DOI 10.46298/theoretics.24.6

1. Introduction

A *distortion-ρ terminal embedding* $f : X \to Y$ for two metric spaces (X, d_X), (Y, d_Y) and given terminal set $T \subseteq X$ is such that

$$\forall x \in T\ \forall q \in X,\ C d_X(x, q) \leqslant d_Y(f(x), f(q)) \leqslant C\rho d_X(x, q).$$

Recently Elkin, Filtser, and Neiman [5] showed the existence of such f when $X = \mathbb{R}^d, Y = \mathbb{R}^m$ with distortion arbitrarily close to $\sqrt{10}$ for $m = O(\log |T|)$. Following [9], the recent work of [10] showed distortion $1+\varepsilon$ is achievable with $m = O(\varepsilon^{-2} \log |T|)$, thus yielding a strict generalization of the Johnson-Lindenstrauss (JL) lemma [8].

One family of motivating applications for dimensionality-reducing terminal embeddings is to high-dimensional computational geometry static data structural problems. To make things concrete, one example to keep in mind is (approximate) nearest neighbor search in Euclidean space: we would like to pre-process a database $D = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ into a data structure to later answer approximate nearest neighbor queries for some other $q \in \mathbb{R}^d$. Given a terminal embedding f with terminal set $T = D$, we can build the data structure on $f(D)$ then later answer queries by querying the data structure on $f(q)$. In this way, we can save on memory and potentially query costs by working over a lower-dimensional version of the problem.

Prior to the introduction of terminal embeddings, the typical approach to applying dimensionality reduction in the above scenario was to observe that the JL lemma actually provides a distribution over (linear embeddings) $x \mapsto \Pi x$ for Π randomly drawn from a distribution independent of D such that $\|\Pi z\|_2 = (1 \pm \varepsilon)\|z\|_2$ with high probability for any fixed $z \in \mathbb{R}^d$. Thus if one wants to make a sequence of queries $q_1, \ldots, q_N$, one could set the failure probability to be $\ll 1/(nN)$ to then have by a union bound that $z_{i,j} = x_i - q_j$ has its norm preserved by Π for all i, j simultaneously with good probability. This approach though does not generally provide correctness guarantees when the sequence of queries is chosen *adaptively*, i.e. the jth query depends upon the responses to queries $q_1, \ldots, q_{j-1}$. This is because since those query responses depend on Π (and perhaps also the randomness of the data structure itself, if it is randomized), future adaptive queries are not independent of Π and the internal randomness of the data structure. Terminal embeddings circumvent this problem and can be used for adaptive queries, since if f is a terminal embedding it is guaranteed to preserve distances to *any* future query q, even one chosen adaptively.

At this point we observe the gap in the above motivation of terminal embeddings and the results of prior work: we want to speed up data structures for high-dimensional problems by allowing querying for $f(q)$ (a "simpler" query as it is lower-dimensional) instead of q, but the optimal terminal embedding of [10] require solving a semidefinite program with $\Theta(n)$ constraints and m variables to compute $f(q)$. Thus the bottleneck becomes computing $f(q)$ itself, which may be even slower than exactly solving the original data structural problem in the

higher-dimensional space (note nearest neighbor search can be solved exactly in linear $O(nd)$ time!). Even the $\sqrt{10}$-distortion terminal embedding construction of [5] (and all subsequent work) requires computing the nearest neighbor of q in D in order to compute $f(q)$, and thus clearly cannot be used to speed up the particular application of approximate nearest neighbor search.

Our Contribution. We give a new terminal embedding construction for Euclidean space into dimension $O(\varepsilon^{-2}\log n)$, where n is the number of terminals, together with a Monte Carlo procedure for computing $f(q)$ given any query $q \in \mathbb{R}^d$. Specifically, we compute $f(q)$ correctly with high probability, even if q is chosen adaptively, and our running time to compute $f(q)$ is *sublinear* in n. Specifically, we can compute $f(q)$ in time $O^*(n^{1-\Theta(\varepsilon^2)} + d)$. The space complexity of our data structure to represent the terminal embedding is at most $O^*(nd)$. Our techniques also yield faster query times if more space is allowed (see Theorem 3.1).

REMARK 1.1. Our main theorem statement (Theorem 3.1) for computing terminal embeddings does not provide the guarantee that if q is queried twice, the same image $f(q)$ will be computed each time. If this property is desired, it can be achieved by increasing the query time to $O^*(n^{1-\Theta(\varepsilon^2)}d)$.

Notation: Through the paper, for $x, y \in \mathbb{R}^d$, $\|x\|$ and $\langle x, y\rangle$ will denote the standard Euclidean norm and inner product. For $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$, we will frequently use calligraphic letters, say $\mathcal{S}$, to denote subsets of the powerset of X. We use $\mathcal{D}$ and (sub/super)scripted versions to denote data structures and use space$(\mathcal{D})$ to denote the space complexity of $\mathcal{D}$ and time$(\mathcal{D}(q))$ to denote the time taken by $\mathcal{D}$ to process query q. We use $\mathbb{B}(x, r)$ to denote the set $\{y : \|y - x\| \leqslant r\}$ and for $A \subset \mathbb{R}^d$, Vol(A) denotes its volume and Conv(A) denotes its convex hull. For a probabilistic event A, $\mathbf{1}\{A\}$ will denote the indicator function of the event. For $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{1, \ldots, n\}$. Finally, $O^*(\cdot)$ and $o^*(\cdot)$ hide factors of poly$(1/\varepsilon) \cdot (nd)^{o(1)}$ and poly(ε) respectively while $\widetilde{O}(\cdot)$ and $\widetilde{\Omega}(\cdot)$ hide poly-logarithmic factors in n and d.

Organization: In Section 2, we provide an overview of our procedure to compute terminal embeddings and the key ideas involved in the analysis. Then, in Section 3, we formally state and prove the main theorem of our paper and in Section 4, we develop a data structure for adaptive approximate nearest neighbor search which we use as a subroutine to prove our main result. Section 5 provides a dimensionality reduction technique tailored to the adaptive setting allowing further speedups in query time, Appendix A details a recursive point partitioning data structure crucial to our construction and Appendix B contains supporting results used in earlier sections.

2. Our Techniques

The starting point of our work is the construction of [9], tightly analyzed in [10]. Before we describe the construction and our generalization, we recall some definitions pertaining to the construction of terminal embeddings. The first is the precise parametrization of a terminal embedding used in our paper:

DEFINITION 2.1 (Terminal Embedding). Given $X \subset \mathbb{R}^d$ and $\varepsilon \in (0,1)$, we say $f : \mathbb{R}^d \to \mathbb{R}^k$ is an *ε-terminal embedding* for X if:

$$\forall x \in X, y \in \mathbb{R}^d : (1-\varepsilon)\|y-x\| \leqslant \|f(y)-f(x)\| \leqslant (1+\varepsilon)\|y-x\|.$$

Next, we recall the notion of an Outer Extension [5, 9]:

DEFINITION 2.2 (Outer Extension). Given $X \subset \mathbb{R}^d$ and $f : \mathbb{R}^d \to \mathbb{R}^k$, we say that $g : \mathbb{R}^d \to \mathbb{R}^{k'}$ for $k' > k$ is an *outer extension* of f on X if:

$$\forall x \in X : g(x) = (f(x), 0, \ldots, 0).$$

All previous approaches [5, 9, 10] as well as ours all construct terminal embeddings by extending a standard distance preserving embedding on X by a single coordinate. Therefore, in all our subsequent discussions, we restrict ourselves to the case where $k' = k+1$. However, [10] require the stronger property that the distance preserving embedding being extended satisfies ε-convex hull distortion, allowing them to obtain the optimal embedding dimension of $O(\varepsilon^{-2}\log n)$:

DEFINITION 2.3. Given $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$ and $\varepsilon > 0$, we say that a matrix $\Pi \in \mathbb{R}^{k\times d}$ satisfies ε-convex hull distortion for X if $\forall z \in \mathrm{Conv}(T) : |\|\Pi z\| - \|z\|| \leqslant \varepsilon$, where $T = \left\{\frac{x-y}{\|x-y\|} : x, y \in X\right\}$.

Furthermore, [10] show that a matrix with i.i.d. sub-Gaussian entries satisfies this property with high probability. We now formally describe the construction analyzed in [10]. Given query q and $\Pi \in \mathbb{R}^{k\times d}$ satisfying ε-convex hull distortion for X, they construct a terminal embedding for q by first finding $v \in \mathbb{R}^k$ satisfying the following constraints:

$$\|v - \Pi\hat{x}\| \leqslant \|q-\hat{x}\|$$
$$\forall x \in X : |\langle v - \Pi\hat{x}, \Pi(x-\hat{x})\rangle - \langle q-\hat{x}, x-\hat{x}\rangle| \leqslant \varepsilon\|q-\hat{x}\|\|x-\hat{x}\| \qquad \text{(Prog)}$$

where $\hat{x} = \arg\min_{x\in X}\|x-q\|$; that is, the closest neighbor of q in X. It is shown in [10], building upon [9], that such a point indeed exists and furthermore, the above set of constraints are convex implying the existence of polynomial time algorithms to find v. Given v^* satisfying the above constraints, it is then shown that one can set $f(q) := z_q = (v^*, \sqrt{\|q-\hat{x}\|^2 - \|v^* - \Pi\hat{x}\|^2})$.

While Prog is *not* the convex program we solve in our work, it is still instructive to analyze it to construct an algorithm with fast query time albeit with large memory. We do this in two steps:

1. We assume access to a separating oracle for the set $K := \{v \in \mathbb{R}^k : v \text{ satisfies Prog}\}$ and analyze the complexity of an optimization algorithm making queries to the oracle.
2. We then construct a separating oracle $\mathcal{O}$ for Prog.

First, observe that there exists a choice of $k = \Theta(\varepsilon^{-2} \log n)$, as was shown in [10]. Now let $\mathcal{O}$ be a separating oracle for the convex set $K = \{v \in \mathbb{R}^k : v \text{ satisfies Prog}\}$; that is, given $v \in \mathbb{R}^k$, $\mathcal{O}$ either reports that $v \in K$ or outputs $u \neq 0$ such that for all $y \in K$, we have $\langle y - v, u \rangle \geqslant 0$. Then standard results on the Ellipsoid Algorithm [11, 3] imply that one can find a feasible v by making $O^*(1)$ calls to $\mathcal{O}$ with each call incurring additional time $O^*(1)$. Hence, we may restrict ourselves to the task of designing a fast separating oracle for K.

To implement a fast separation oracle, first note that the first constraint in Prog can be checked explicitly in time $O(d)$ and if the constraint is violated, the oracle may return $\Pi\hat{x} - v$. If the first constraint is satisfied, consider the following re-arrangement of the second constraint:

$$\forall x \in X : \left| \left\langle \frac{v - \Pi\hat{x}}{\|q - \hat{x}\|}, \Pi\left(\frac{x - \hat{x}}{\|x - \hat{x}\|}\right) \right\rangle - \left\langle \frac{q - \hat{x}}{\|q - \hat{x}\|}, \frac{x - \hat{x}}{\|x - \hat{x}\|} \right\rangle \right| \leqslant \varepsilon.$$

By denoting $\widetilde{v}_y = \frac{(q-y, -(v - \Pi y))}{\|(q-y, -(v-\Pi y))\|}$ and $\widetilde{v}_{y,z} = \frac{(y-z, \Pi(y-z))}{\|(y-z, \Pi(y-z))\|}$, we see that the above constraint essentially tries to enforce that $\langle \widetilde{v}_{\hat{x}}, \widetilde{v}_{x,\hat{x}} \rangle$ is close to 0. Note that from the constraints that we have already checked previously, we have that $\|\widetilde{v}_{x,\hat{x}}\|, \|\widetilde{v}_{\hat{x}}\| = 1$ and hence, the condition $|\langle \widetilde{v}_{\hat{x}}, \widetilde{v}_{x,\hat{x}} \rangle| \approx 0$ is equivalent to $\|\widetilde{v}_{\hat{x}} \pm \widetilde{v}_{x,\hat{x}}\| \approx \sqrt{2}$. Conversely, $|\langle \widetilde{v}_{\hat{x}}, \widetilde{v}_{x,\hat{x}} \rangle| \gg C\varepsilon$ implies $\|\widetilde{v}_{\hat{x}} \pm \widetilde{v}_{x,\hat{x}}\| \leqslant \sqrt{2} - C\varepsilon$ for some signing. Since, $\widetilde{v}_{x,y}$ for $x, y \in X$ are independent of q, we may build a fast separating oracle by building n nearest neighbor data structures, one for each $x \in X$, with the point set $\{\pm\widetilde{v}_{y,x}\}_{y \in X}$ and at query time, constructing $\widetilde{v}_{\hat{x}}$ and querying the data structure corresponding to $\hat{x}$. Despite this approach yielding a fast separating oracle, it has three significant shortcomings:

1. Computing exact nearest neighbor in high dimensions is inefficient
2. Queries may be adaptive, violating guarantees of known randomized approximate nearest neighbor data structures
3. The space complexity of the separating oracle is rather large.

The first two points are easily resolved: the correctness guarantees can be straightforwardly extended to the setting where one works with an approximate nearest neighbor and adopting the framework from [4] in tandem with known reductions from Approximate Nearest Neighbor to Approximate Near Neighbor yield an adaptive approximate nearest neighbor algorithm.

For the third point, note that in the approach just described, we construct n data structures with n data points each. Hence, even if each data structure can be implemented in $O(n)$ space, this still yields a data structure of at least quadratic space complexity. The rest of our discussion is dedicated to addressing this difficulty.

We first generalize Prog, somewhat paradoxically, by adding more constraints:

$$\forall y \in X : \|v - \Pi y\| \leqslant (1+\varepsilon)\|q - y\|$$
$$\forall x, y \in X : |\langle v - \Pi y, \Pi\,(x - y)\rangle - \langle q - y, x - y\rangle| \leqslant \varepsilon\|q - y\|\|x - y\|. \qquad \text{(Gen-Prog)}$$

Despite these constraints (approximately) implying those in Prog, they are also implied by those from Prog. Hence, in some sense the two programs are equivalent but it will be more convenient to describe our approach as it relates to the generalized set of constraints. These constraints may be interpreted as a multi-centered characterization of the set of constraints in Prog. While Prog only has constraints corresponding to a centering of q with respect to $\hat{x}$ and its projection, Gen-Prog instead requires v to satisfy similar constraints irrespective of centering.

The first key observation behind the construction of our oracle is that it is not *necessary* to find a point satisfying all the constraints Gen-Prog. It suffices to construct an oracle, $\mathcal{O}$, satisfying:

1. If $\mathcal{O}$ outputs FAIL, v can be extended to a terminal embedding for q; that is v may be appended with one more element to form a valid distance-preserving embedding for q.
2. Otherwise, $\mathcal{O}$ outputs a separating hyperplane for Gen-Prog.

This is weaker than the one just constructed for Prog in two significant ways: the oracle may output FAIL even if v does *not* satisfy Prog, and the expanded set of constraints allow a greater range of candidate separating hyperplanes, hence, making the work of the oracle "easier". A technical benefit of introducing the program Gen-Prog is that it provides the definition of a "nice" convex body for which this oracle outputs a separating hyperplane when it does not output FAIL, hence, allowing use to inherit the convergence bounds from the use of the Ellipsoid method for convex optimization.

The second key observation underlying the design of our oracle is one that allows restricting the set of relevant constraints for each input substantially. Concretely, to ensure that v can be extended to a point preserving its distance to any $x \in X$, it is sufficient to satisfy the following two constraints for all $x \in X$:

$$\|v - \Pi y\| \leqslant (1+\varepsilon)\|q - y\|$$
$$|\langle v - \Pi y, \Pi(x - y)\rangle - \langle q - y, x - y\rangle| \leqslant \varepsilon\|q - y\|\|x - y\|$$

for any $y \in X$ satisfying $\|q - y\| = O(\|q - x\|)$. In particular, the point y may be much farther from q than $\hat{x}$ but still constitutes a good "centering point" for x. Therefore, we simply need to ensure that our oracle checks at least one constraint involving a valid centering point for x for all $x \in X$. However, at this point, several difficulties remain:

1. Which constraints should we satisfy for any input query q?
2. How do we build a succinct data structure quickly checking these constraints?

These difficulties are further exacerbated by the fact that the precise set of constraints may depend on the query q which may be chosen adaptively. In the next two subsections, we address these issues with the following strategy where we still make use of nearest neighbor data structures over $\{\widetilde{v}_{x,y}\}_{x,y\in X}$:

1. Each nearest neighbor data structure consists of points $\{\widetilde{v}_{y,x}\}_{y\in S}$ for some small $S \subset X$
2. Use a smaller number of data structures
3. Only a small number of relevant data structures are queried when presented with query q

For each of these three choices, we will exploit recent developments in approximate near neighbor search [2]. In Subsection 2.1, we describe our approach to construct an oracle for a fixed scale setting where the distance to the nearest neighbor is known up to a polynomial factor and finally, describe the reduction to the fixed scale setting in Subsection 2.2.

2.1 Fixed Scale Oracle

In this subsection, we outline the construction of a suitable separation oracle for all q satisfying $\widetilde{r} \leqslant \|q - \hat{x}\| \leqslant \text{poly}(n)\widetilde{r}$ for some *known* $\widetilde{r}$. For the sake of illustration, in this subsection we only consider the case where our oracle has space complexity $O^*(nd)$ though our results yield faster oracles if more space is allowed (see Theorem 3.1). In order to decide which points to use to construct our nearest neighbor data structures, we will make strong use of the following randomized partitioning procedures. Given $X \subset \mathbb{R}^d$, these data structures construct a set of subsets $\mathcal{S} = \{S_i \subseteq X\}_{i=1}^m$ and a hash function $h : \mathbb{R}^d \to 2^{[m]}$ such that for a typical input point, x, sets in $h(x)$ contain points that are close to x and exclude points far from x. The data structure is formally defined below recalling that space$(\mathcal{D})$ and time$(\mathcal{D})$ denote the space and time complexities of a data structure $\mathcal{D}$:

DEFINITION 2.4. We say a randomized data structure is an (ρ_u, ρ_c)-Approximate Partitioning (AP) if instantiated with a set $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$ and $r > 0$, produces, $\mathcal{D} = (h, \mathcal{S} = \{S_i\}_{i=1}^m)$, with $S_i \subseteq X$, $|S_i| > 0$ and $h : \mathbb{R}^d \to 2^{[m]}$ satisfying:

$$\mathbb{P}\left\{\text{space}(\mathcal{D}) \leqslant O^*(n^{1+\rho_u}d) \text{ and } \sum_{i=1}^m |S_i| \leqslant O^*(n^{1+\rho_u})\right\} = 1$$

$$\forall x \in \mathbb{R}^d : \mathbb{P}\left\{|h(x)| \leqslant O^*(n^{\rho_c}) \text{ and time}(h(x)) \leqslant O^*(n^{\rho_c}d)\right\} = 1$$

$$\forall x \in X : \mathbb{E}\left\{\sum_{i=1}^m \mathbf{1}\{x \in S_i\}\right\} \leqslant O^*(n^{\rho_u})$$

$$\forall x \in \mathbb{R}^d : \mathbb{E}\left\{\sum_{\substack{y\in X \\ \|x-y\|\geqslant 2r}} \sum_{i\in h(x)} \mathbf{1}\{y \in S_i\}\right\} \leqslant O^*(n^{\rho_c})$$

$$\forall x \in \mathbb{R}^d, y \in X \text{ such that } \|x - y\| \leqslant r : \mathbb{P}\left\{\sum_{i \in h(x)} \mathbf{1}\{y \in S_i\} \geqslant 1\right\} \geqslant 0.99$$

where the probability is taken over the random decisions used to construct $\mathcal{D}$.

The first condition in the above definition restricts the space complexity of the data structure and the sum of the number of points stored in all of the sets in $\mathcal{S}$ while the third condition states that each point is replicated very few times across all the sets in $\mathcal{S}$ in expectation. The second condition states that for any input x, h is computable quickly and maps x to not too many sets in $\mathcal{S}$. Finally, the last two conditions ensure that points far from x are rarely in the sets x maps to and that points close to x are likely to be found in these sets. Essentially, these data structures will allow us to partition our space such that most points in each partition are close to each other and hence, the constraints corresponding set from Gen-Prog can be checked with a small number of centering points while the few far away points may be checked explicitly.

Data structures satisfying the above definition have been a cornerstone of LSH-based approaches for approximate nearest neighbor search which yield state-of-the-art results and nearly optimal time-space tradeoffs [7, 1, 2]. The conditions with probability 1 can be ensured by truncating the construction of the data structure if its space complexity grows too large or by truncating the execution of h on x if its runtime exceeds a certain threshold. Finally, the events with probability 0.99 can be boosted from arbitrary constant probability by repetition. In this subsection's setting of almost linear memory, any $(0, c)$-AP data structure suffices for any $c < 1$, and [2] show that an $(0, 7/16)$-AP data structure exists.

For the sake of exposition, assume that the second, third and fourth conditions in the above definition hold deterministically; that is, assume that each data point is only replicated in $O^*(n^{\rho_u})$ many of the S_i, for each x only $O^*(n^{\rho_c})$ many points farther away from x than $2r$ are present in the sets mapped to by h and each point in the dataset within r of x is present in one of the sets that x maps to. In our formal proof, we show that any issues caused due to the randomness are easily addressed. We now instantiate $O(\log(n)/\gamma)$ independent $(0, 7/16)$-AP data structures, $\mathcal{D}_i = (h_i, \mathcal{S}_i = \{S_j\}_{j=1}^{m_i})$ for the point set X with $r_i = (1+\gamma)^i \widetilde{r}$ and $\gamma \approx 1/\log^3 n$. Note that this only results in $O^*(1)$ data structures in total. Now, for each i and $S \in \mathcal{S}_i$, we pick $l \approx \log n$ random points from S, $\mathcal{Z}_{i,S} = \{z_j\}_{j=1}^{l}$, and instantiate nearest neighbor data structures for the points $\{\pm\widetilde{v}_{x,z_j}\}_{x \in S}$ and assign any point $y \in S$ within $4r_i$ of z_j to z_j for S. Note that these assignments are only for the set S; for a distinct set S' such that $y \in S'$, y need not be assigned to any point. Intuitively, this assignment will be used to determine which of the nearest-neighbor data structures are queried on an input query q for which a terminal embedding is required to be computed as we only instantiate these data structures for some points in S_i – in fact, we will choose the points in $\mathcal{Z}_{i,S}$ to which the nearest neighbor of q in X has been assigned. Each of these data structures only stores $O^*(n)$ points in total and the existence of near neighbor

data structures with space complexity $O^*(dn)$ (and query time $O^*(dn^{1-\widetilde{c}\varepsilon^2})$) [2] complete the bound on the space complexity of a data structure at a single scale. Since we only instantiate $O^*(1)$ many such data structures, this completes the bound on the space complexity of the data structure. By choosing the points randomly in this way, one can show that for any x, the sum total of the number of unassigned points over all the sets in $h_i(x)$ (including potentially duplicated points) is at most $O^*(n^{\rho_c})$ by partitioning X into a set with points close to x and those far away and analyzing their respective probabilities of being assigned in each set. Furthermore, note the total number of points stored in all the $h_i(x)$ is trivially at most $O^*(n)$.

At query time, suppose we are given query q, its nearest neighbor in X, $\hat{x}$, and a candidate v and we wish to check whether v can be extended to a valid terminal embedding for q. While having access to the exact nearest neighbor is an optimistic assumption, extending the argument to use an approximate nearest neighbor is straightforward. We query the data structure as follows, we query each data structure $\mathcal{D}_i$ with $\hat{x}$ and for each $S \in h_i(\hat{x})$, we check whether each unassigned point, x, satisfies $|\langle \widetilde{v}_{x,\hat{x}}, \widetilde{v}_{\hat{x}} \rangle| \approx 0$. Then, for each point in $z_j \in \mathcal{Z}_{i,S}$, we query its nearest neighbor data structure with $\widetilde{v}_{z_j}$. If any of the data structures report a point significantly violating the inner product condition, we return that data point as a violator to our set of constraints.

We now prove the correctness and bound the runtime of the oracle. We start by bounding the runtime of this procedure. For a single scale, r_i, we query at most $O^*(n^{\rho_c})$ unassigned points and their contribution to the runtime is correspondingly bounded. Intuitively, this is true because $h(x)$ contains at most $O^*(n^{\rho_c})$ points far from x and if there are more than $O^*(n^{\rho_c})$ points close to x, they tend to be assigned. For assigned points, there are at most $O^*(n)$ many of them (repetitions included) spread between $O^*(n^{\rho_c})$ many nearest neighbor data structures (as $|h_i(\hat{x})| \leqslant O^*(n^{\rho_c})$) and each of these data structures has query time $O^*(l^{1-\widetilde{c}\varepsilon^2})$ where l is the number of points assigned to the data structure. A simple convexity argument shows that the time taken to query all of these is at most $O^*(n^{1-\widetilde{c}\varepsilon^2})$. This bounds the query time of the procedure.

To establish its correctness, observe that when the algorithm outputs a hyperplane, the correctness is trivial. Suppose now that the oracle outputs FAIL. Note that any point, $x \in X$, *very* far away from $\hat{x}$ may be safely ignored (say, those $\text{poly}(n)\widetilde{r}$ away from $\hat{x}$) as an embedding that preserves distance to $\hat{x}$ also preserves distance to x by the triangle inequality. We now show that for any other x, it satisfies $|\langle \widetilde{v}_{x,y}, \widetilde{v}_y \rangle| \approx 0$ for some y with $\|q - y\| \leqslant C\|q - x\|$. Any such x must satisfy $\|x - \hat{x}\| \leqslant 2\|q - x\|$ by the triangle inequality and the fact that $\hat{x}$ is the nearest neighbor. As a consequence, there exists i such that $\|x - \hat{x}\| \leqslant r_i$ and $\|x - q\| \geqslant 0.5r_{i-1}$. For this i, there exists $S \in h_i(\hat{x})$ containing x. In the case that x is not assigned, we check $|\langle \widetilde{v}_{x,\hat{x}}, \widetilde{v}_{\hat{x}} \rangle| \approx 0$ and correctness is trivial. In case x is assigned, it is assigned to y with $\|y - x\| \leqslant 4r_i$ and we have:

$$\|y - q\| \leqslant \|x - q\| + \|y - x\| \leqslant \|x - q\| + 4r_i \leqslant 10\|x - q\|$$

where the final inequality follows from the fact that $\|x - q\| \geqslant 0.5 r_{i-1}$. This proves that the inner product condition for x is satisfied with respect to y with $\|q - y\| \leqslant 10\|x - q\|$. This concludes the proof in the second case where the oracle outputs FAIL.

The argument outlined in the last two paragraphs concludes the construction of our weaker oracle when an estimate of $\|q - \hat{x}\|$ is known in advance. The crucial property provided by the existence of the (ρ_u, ρ_c)-AP procedure is that there are at most $O^*(n)$ many points used to construct the near neighbor data structures for the points $\{\widetilde{v}_{x,y}\}$ (as opposed to n^2 for the previous construction). This crucially constrains us to having either a large number of near neighbor data structures with few points or a small number with a large number of points but not both. However, the precise choice of how the algorithm trades off these two competing factors is dependent on the set of data points and the scale being considered. The savings in query time follow from the fact that at most $O^*(n^c)$ of these data structures are consulted for any query for some $c < 1$.

2.2 Reduction to Fixed Scale Oracle

We reduce the general case to the fixed scale setting from the previous subsection.[1] To define our reduction, we will need a data structure we will refer to as a Partition Tree that has previously played a crucial part in reductions from the Approximate Nearest Neighbor to Approximate Near Neighbor [7]. We show that the same data structure also allows us to reduce the oracle problem from the general case to the fixed scale setting. Describing the data structure and our reduction, requires some definitions from [7]:

DEFINITION 2.5. Let $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$ and $r > 0$. We will use $\mathrm{GG}(X, r)$ to denote the graph with nodes indexed by x_i and an edge between x_i and x_j if $\|x_i - x_j\| \leqslant r$. The connected components of this graph will be denoted by $\mathrm{CC}(X, r)$; that is, $\mathrm{CC}(X, r) = \{C_j\}_{j=1}^m$ is a partitioning of X with $x \in C_j$ if and only if $\|x - y\| \leqslant r$ for some $y \in C_j \setminus \{x\}$.

Note from the above definition that $\mathrm{CC}(X, r)$ results in increasingly fine partitions of X as r decreases. This notion is made precise in the following straightforward definition:

DEFINITION 2.6. For a data set $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$, we say that a partition C *refines* a partition C' if for all $C \in C$, $C \subseteq C'$ for some C' in C'. This will be denoted by $C' \sqsubseteq C$.

Next, define $r_{\mathrm{med}}(X)$ as:

$$r_{\mathrm{med}}(X) = \min\{r > 0 : \exists C \in \mathrm{CC}(X, r) \text{ with } |C| \geqslant n/2\}.$$

We are now ready to define a Partition Tree:

1 Note that Subsection 2.1 already allows the construction of a separating oracle if one is willing to incur runtimes depending *logarithmically* in the aspect ratio of the dataset. However, the results in the paper allow for a construction with *no* such dependence.

DEFINITION 2.7. Given $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$, a Partition Tree of X is a tree, $\mathcal{T}$, whose nodes are labeled by $\left(Z, \{\mathcal{T}_C\}_{C \in \mathcal{C}_{\text{low}}}, \mathcal{T}_{\text{rep}}, \mathcal{C}_{\text{low}}, \mathcal{C}_{\text{high}}, C_{\text{rep}}, r_{\text{apx}}\right)$ where $Z, C_{\text{rep}} \subset X$, $\{\mathcal{T}_C\}_{C \in \mathcal{C}_{\text{low}}} \cup \{\mathcal{T}_{\text{rep}}\}$ represent its children, $\mathcal{C}_{\text{low}}, \mathcal{C}_{\text{high}}$ are partitions of Z and $r_{\text{apx}} > 0$ satisfying the following conditions:

$$\text{CC}(Z, 1000n^2 r_{\text{apx}}) \sqsubseteq \mathcal{C}_{\text{high}} \sqsubseteq \text{CC}(Z, r_{\text{apx}}) \sqsubseteq \text{CC}(Z, r_{\text{med}}) \sqsubseteq \text{CC}\left(Z, \frac{r_{\text{apx}}}{10n}\right) \sqsubseteq \mathcal{C}_{\text{low}} \sqsubseteq \text{CC}\left(Z, \frac{r_{\text{apx}}}{1000n^3}\right)$$
$$\forall C \in \mathcal{C}_{\text{high}} : \exists!\, z \in C_{\text{rep}} \text{ with } z \in C.$$

For the sake of notational simplicity, we will use $\mathcal{T}' \in \mathcal{T}$ both to refer to a node in the tree as well as the subtree rooted at that node and $\text{Size}(\mathcal{T}')$ to refer to the sum of the number of points stored in the subtree $\mathcal{T}'$. The above condition implies $\text{Size}(\mathcal{T}) \leqslant O(n \log n)$ [7].

While deterministic data structures with the same near linear runtime are also known [6], we include, for the sake of completeness, a simple probabilistic algorithm to compute a partition tree with probability $1 - \delta$ in time $O(nd \log^2 n/\delta)$. Having defined the data structure, we now describe how it may be used to define our reduction.

At a high level, the reduction traverses the data structure starting at the root and at each step either terminating at the node currently being explored or proceeds to one of its children. By the definition of the data structure, the number of points in the node currently being explored drops by at least a factor of 2 in each step. Therefore, the procedure explores at most $\lceil \log n \rceil$ nodes. For any node $\mathcal{T}' \in \mathcal{T}$, with associated point set Z, currently being traversed, we will aim to enforce the following two conditions:

1. An approximate nearest neighbor of q in Z is also an approximate nearest neighbor in X
2. A terminal embedding of q for Z is also valid for X.

For simplicity, we assume the existence of near neighbor data structures; that is, data structures which when instantiated with (X, r) and given a query q, output a candidate $y \in X$ such that $\|y - q\| \leqslant r$ if $\min_{x \in X} \|x - q\| \leqslant r$ (we refrain from assuming access to nearest neighbor data structures here as this reduction will also be used to construct our nearest neighbor data structures). For each node $\mathcal{T}' = \left(Z, \{\mathcal{T}_C\}_{C \in \mathcal{C}_{\text{low}}}, \mathcal{T}_{\text{rep}}, \mathcal{C}_{\text{low}}, \mathcal{C}_{\text{high}}, C_{\text{rep}}, r_{\text{apx}}\right)$, we first decide two thresholds $r_{\text{low}} = r_{\text{apx}}/\text{poly}(n)$ and $r_{\text{high}} = \text{poly}(n) r_{\text{apx}}$ and interpolate the range with roughly $m \approx (\log r_{\text{high}}/r_{\text{low}})/\gamma$ many near neighbor data structures, $\{\mathcal{D}_i\}_{i=0}^m$, with $r_i = (1+\gamma)^i r_{\text{low}}$ for the point set Z. Note, we set $\gamma \approx 1/\log^3 n$ which implies that we instantiate at most $O^*(1)$ many near neighbor data structures.

At query time, suppose we are at node $\mathcal{T}' = \left(Z, \{\mathcal{T}_C\}_{C \in \mathcal{C}_{\text{low}}}, \mathcal{T}_{\text{rep}}, \mathcal{C}_{\text{low}}, \mathcal{C}_{\text{high}}, C_{\text{rep}}, r_{\text{apx}}\right)$ with associated near neighbor data structures, $\mathcal{D}_i$. We query each of the data structures $\mathcal{D}_i$ with q and we have three possible cases:

1. The nearest neighbor to q is within a distance of r_{low}
2. The nearest neighbor to q is beyond r_{high} of it

3. The nearest neighbor to q is between r_{low} and r_{high} of q

The first case occurs when $\mathcal{D}_0$ returns a candidate nearest neighbor, the second when none of the near neighbor data structures return a candidate and the third when, $\mathcal{D}_i$ succeeds but $\mathcal{D}_{i-1}$ fails for some i. If the third case occurs, the reduction is complete. If the second case occurs, let $x \in C \in C_{\text{high}}$ and $\widetilde{x} \in C_{\text{rep}} \cap C$. We have by the triangle inequality and Definitions 2.5 and 2.7 $\|x - \widetilde{x}\| \leqslant \text{poly}(n) r_{\text{med}} \ll r_{\text{high}}$ and hence $\|q - \widetilde{x}\| \approx \|q - x\|$ and hence we recurse in C_{rep} still satisfying the two conditions stated above. For the first case, let $\widetilde{x} \in C \in C_{\text{low}}$ such that $\|q - \widetilde{x}\| \leqslant r_{\text{low}}$. From Definitions 2.5 and 2.7, any $x \notin C$ satisfies $\|x - \widetilde{x}\| \geqslant r_{\text{med}}/\text{poly}(n) \gg r_{\text{low}}$ and hence, both conditions are again maintained as the nearest neighbor of q in Z is in C and a terminal embedding for q in C is a terminal embedding in Z by the triangle inequality.

2.3 Runtime Improvements and Adaptivity

We conclude with other technical considerations glossed over in the previous discussion. As remarked before, we assumed access to exact nearest and near neighbor data structures which perform correctly even when faced with adaptive queries. While the arguments outlined in the previous two subsections extend straightforwardly to the setting where approximate nearest neighbors are used, the guarantees provided by previous approaches to the approximate near neighbors are not robust in the face of adaptivity.

DEFINITION 2.8. For $\rho_u, \rho_c \geqslant 0$ and $c > 1$, we say a randomized data structure is an (ρ_u, ρ_c, c)-Approximate Near Neighbor (ANN) data structure for Approximate Near Neighbor if instantiated with a set of data points $X = \{x_i\}_{i=1}^n$ and $r > 0$ constructs, $\mathcal{D}$, satisfying:

$$\mathbb{P}\left\{\text{space}(\mathcal{D}) \leqslant O^*(dn^{1+\rho_u})\right\} = 1$$
$$\forall q \in \mathbb{R}^d : \mathbb{P}\left\{\text{time}(\mathcal{D}(q)) \leqslant O^*(dn^{\rho_c})\right\} = 1$$
$$\forall q \in \mathbb{R}^d : \mathbb{P}\left\{\mathcal{D}(q) \text{ returns } x \in X \text{ with } \|q - x\| \leqslant cr \text{ if } \exists y \in X \text{ with } \|q - y\| \leqslant r\right\} \geqslant 0.99$$

where the probability is taken over *both* the random decisions used to construct $\mathcal{D}$ and those used by $\mathcal{D}$ to answer the query q. Additionally, q is assumed to be independent of $\mathcal{D}$.

A simple repetition argument can then be used to devise adaptive near neighbor data structure with similar guarantees [4]. Used in tandem with the reduction outlined in the previous subsection yields the adaptive nearest neighbor data structures used in Subsection 2.1.

Finally, the argument outlined previously enabled computing terminal embeddings in time $O^*(dn^\rho)$ for some $\rho < 1$ which while being sublinear in n is suboptimal in its interaction with the dimension. This is in contrast to state-of-the-art approaches to nearest neighbor search which yield runtimes scaling as $O^*(dn + n^\rho)$. However, all these approaches deduce this result by first projecting onto a lower dimensional space using a Johnson-Lindenstrauss (JL) projection

and building the data structure in the lower dimensional space. This is not feasible in our setting as JL projections are not robust to adaptivity and we require an alternate strategy.

To improve our runtime, suppose $\Pi' \in \mathbb{R}^d \to \mathbb{R}^k$ with $k = O^*(1)$ satisfies:

$$\forall x, y, z \in X \cup \{q\} : |\langle \Pi'(x-z), \Pi'(y-z)\rangle - \langle x-z, y-z\rangle| \leqslant o^*(1) \cdot \|x-z\|\|y-z\|. \quad (1)$$

Then, to construct a terminal embedding, we may construct the vectors $\widetilde{v}_{x,y}$ and $\widetilde{v}_x$ by using the vectors $(\Pi' x, \Pi' y, \Pi' q)$ instead of the vectors in the high dimensional space. Assuming the projections for $x \in X$ are pre-computed, we may do this projection in $O^*(d)$ time and the rest of the procedure to compute terminal embeddings takes time $O^*(n^\rho)$. When q is independent of the data structure, a standard JL-sketch satisfies Equation (1) with high probability but this is not true when q depends on $\mathcal{D}$. Thankfully, we show that if one draws $O^*(d)$ many JL-sketches, at least 95% satisfy Equation (1) for any query q with high probability. Note that the order of the quantifiers in the statement make the proof more challenging than previous work using such ideas [4] and requires a careful gridding argument which we carry out in Section 5.

3. Terminal Embeddings

In this section, prove the main theorem of the paper. Note that in the following theorem the query q can be chosen with full knowledge of the data structure. That is, conditioned on the successful creation of the data structure, the randomized construction of z_q is only over the random decisions taken at *query-time* and not during the creation of the data structure. The main theorem of the paper is stated below:

THEOREM 3.1. *Let $\varepsilon \in (0,1)$, $\rho_1, \rho_2, \rho_3, \rho_4, \rho_{\text{rep}} > 0$. Then, there is a randomized procedure which when instantiated with a dataset $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$, a (ρ_3, ρ_4)-Approximate Partitioning data structure, a $(\rho_1, \rho_2, (1+\varepsilon^\dagger))$-Approximate Near Neighbor data structure for $\varepsilon^\dagger = c\varepsilon$ for some small enough $c > 0$ and parameter ρ_{rep} constructs a data structure, $(\mathcal{D}, \Pi \in \mathbb{R}^{k\times d})$, satisfying the following guarantees:*

1. *Π has ε-convex hull distortion for X*
2. *Given $q \in \mathbb{R}^d$, $\mathcal{D}$ produces with probability at least $1 - 1/\text{poly}(n)$ over the randomness of $\mathcal{D}$ at query time, a vector $z_q \in \mathbb{R}^{k+1}$ such that:*

$$\forall x \in X : (1 - O(\varepsilon))\|q - x\| \leqslant \|z_q - (\Pi x, 0)\| \leqslant (1 + O(\varepsilon))\|q - x\|$$

3. *On any $q \in \mathbb{R}^d$, the runtime of $\mathcal{D}$ is $O^*(d + n^{\rho_2} + n^{\rho_4} + n^{\rho_4 + (1+\rho_3-\rho_4-\rho_{\text{rep}})\rho_2})$*
4. *The space complexity of $\mathcal{D}$ is $O^*(dn^{\rho_{\text{rep}}+(1+\rho_1)} + dn^{\rho_3+(1+\rho_1)})$*

with probability at least $1 - 1/\text{poly}(n)$ over the randomness during the instantiation of $\mathcal{D}$.

Before we proceed, we will instantiate the above theorem in three specific cases. We note that the state-of-the-art algorithms for approximate nearest neighbor search are implemented

in terms of approximate partitioning schemes as defined here [2] who show for any (ρ_u, ρ_c) satisfying:

$$c^2\sqrt{\rho_c} + (c^2-1)\sqrt{\rho_u} = \sqrt{2c^2-1},$$

there exist both a (ρ_u, ρ_c, c)-Approximate Near Neighbor data structure and specifically, a (ρ_u, ρ_c)-Approximate Partitioning scheme when $c = 2$. By instantiating an Approximate Partitioning data structure (by setting $c = 2$) and applying the results of Theorem 3.1, we get a data structure to compute ε-terminal embeddings for (possibly different) universal $\widetilde{c}, \widetilde{C} > 0$:

— Query time $O^*(d + n^{1-\widetilde{c}\varepsilon^2})$ and space complexity $O^*(nd)$ with $\rho_{\text{rep}} = \rho_1 = \rho_3 = 0$
— Query time $O^*(d + n^{1-\widetilde{c}\varepsilon})$ and space complexity $O^*(n^2 d)$ with $\rho_{\text{rep}} = \rho_3 = 0, \rho_1 = 1$
— Query time $O^*(d)$ and space complexity $dn^{\widetilde{C}/\varepsilon^2}$ with $\rho_4 = \rho_2 = 0$.

The previous three results capture a range of potential time space tradeoffs for data structures computing terminal embeddings. We now move on to the proof of Theorem 3.1. As discussed in Section 2, our algorithm operates by constructing a weak separating oracle for a convex program. In Subsection 3.1, we define the convex program for which we define our oracle, Subsection 3.2 defines our weak oracle for a fixed scale and in Subsection 3.3 we describe the data structure which enables the reduction of the general case to the fixed scale scenario and establish that it provides an appropriate separation oracle for the convex program introduced in Subsection 3.1.

A key data structure that will be extensively utilized and will, henceforth, be referred to as a (ρ_u, ρ_c, c)-Adaptive Approximate Nearest Neighbor (AANN) data structure is described in the following theorem. It supplies a data structure supporting several important functions. It enables approximate nearest neighbor queries on the dataset *even* for queries which are adaptively chosen based on the instantiation of the data structure (but not on the fresh randomness drawn at query time) and furthermore, constructs a partition tree, $\mathcal{T}$, with the property that for any query q, it suffices to construct a valid terminal embedding for the data points in node in the partition tree that the data structure returns, $\mathcal{T}_{\text{res}}$. The guarantees for all other points in the dataset are guaranteed as a consequence of the first claim of the theorem. The key use of this data structure is that it now effectively suffices to construct a valid terminal embedding for the node in the partition tree that is returned in response to the query. Furthermore, the distance bounds on the distance of q to its approximate nearest neighbor $\hat{x}$, allows us to use a small number of *fixed-scale* data structures for each node thus directly reducing the multi-scale setting to the fixed-scale setting. The proof of the theorem itself will be deferred to Section 4.

THEOREM 3.2. *Let $c > 1$ and $\rho_u, \rho_c > 0$. Then, there is a randomized procedure which when instantiated with a dataset $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$ and a (ρ_u, ρ_c, c)-Approximate Near Neighbor data structure (Definition 2.8) produces a data structure, $(\mathcal{D}, \mathcal{T})$, satisfying:*

1. *Given any $q \in \mathbb{R}^d$, $\mathcal{D}$ produces $(\hat{x} \in X, \mathcal{T}_{\text{res}} \in \mathcal{T})$ satisfying:*
   a. $\|q - \hat{x}\| \leqslant \min_{x \in X}(1 + o^*(1))c\|q - x\|$

b. $\hat{x} \in \mathcal{T}_{\text{res}}$

c. Furthermore, let $\mathcal{Y} = \{y_i\}_{i=1}^n \subset \mathbb{R}^k$ *satisfying for some* $\varepsilon^\dagger \in \left(\frac{1}{\sqrt{d}}, 1\right)$*:*

$$\forall i, j \in [n] : (1-\varepsilon^\dagger)\|x_i - x_j\| \leqslant \|y_i - y_j\| \leqslant (1+\varepsilon^\dagger)\|x_i - x_j\|$$

and for $\mathcal{T}_{\text{res}} = \left(Z, \{\mathcal{T}_C\}_{C \in C_{\text{low}}}, \mathcal{T}_{\text{rep}}, C_{\text{low}}, C_{\text{high}}, C_{\text{rep}}, r_{\text{apx}}\right)$*, let* $y \in \mathbb{R}^k$ *satisfy for* $\varepsilon^\ddagger \in [\varepsilon^\dagger, 1)$*:*

$$\forall x_i \in Z : (1-\varepsilon^\ddagger)\|q - x_i\| \leqslant \|y - y_i\| \leqslant (1+\varepsilon^\ddagger)\|q - x_i\|.$$

Then:

$$\forall x_i \in X : \left(1 - (1+o^*(1))\,\varepsilon^\ddagger\right)\|q - x_i\| \leqslant \|y - y_i\| \leqslant \left(1 + (1+o^*(1))\,\varepsilon^\ddagger\right)\|q - x_i\|.$$

and if $|Z| > 1$*:*

$$\Omega\left(\frac{1}{(nd)^{10}} r_{\text{apx}}\right) \leqslant \|q - \hat{x}\| \leqslant O((nd)^{10} r_{\text{apx}})$$

with probability at least $1 - 1/\text{poly}(n)$

2. $\mathcal{T}$ *is a valid Partition Tree of* X *(Definition 2.7)*
3. *The space complexity of* $\mathcal{D}$ *is* $O^*(dn^{1+\rho_u} \log 1/\delta)$
4. *The runtime of* $\mathcal{D}$ *on any* $q \in \mathbb{R}^d$ *is at most* $O^*(d + n^{\rho_c})$

with probability $1 - \delta$*.*

3.1 Generalized Characterization of Terminal Embeddings

To start, we recall a key lemma from [10]:

LEMMA 3.3. *Let* $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$*,* $\varepsilon \in \left(\frac{1}{\sqrt{n}}, 1\right)$ *and* $T = \left\{\frac{x-y}{\|x-y\|} : x \neq y \in X\right\} \bigcup \{0\}$*. Then for* $\Pi \in \mathbb{R}^{k \times d}$ *with* $k = \Omega\left(\frac{\log n + \log 1/\delta}{\varepsilon^2}\right)$ *with* $\Pi_{i,j} \sim \mathcal{N}(0, 1/k)$*,*

$$\mathbf{Pr}(\Pi \textit{ satisfies } \varepsilon\textit{-convex hull distortion for } T) \geqslant 1 - \delta.$$

Finally, the convex program for which we will construct our oracle is defined in the following generalization of [10, 9]. As remarked in Section 2, we generalize the convex program to an expanded set of constraints but crucially do not attempt to satisfy *all* of them in our algorithm. For the convergence guarantees for our weak oracle to hold (Lemma 3.8), we first need to show the feasibility of the convex program. Before, we proceed we require the following simple lemma which is proved in Appendix B.2.

LEMMA 3.4. *Let* $X = \{x_i\}_{i=1}^n$*,* $0 < \varepsilon < 1$ *and* $T = \left\{\frac{x-y}{\|x-y\|} : x \neq y \in X\right\} \bigcup \{0\}$*. Furthermore, suppose* $\Pi \in \mathbb{R}^{k \times d}$ *has* ε*-convex hull distortion for* X*. Then, we have:*

$$\forall x, y \in \text{Conv}(T) : |\langle \Pi x, \Pi y \rangle - \langle x, y \rangle| \leqslant 6\varepsilon.$$

We now establish feasibility in the following lemma.

LEMMA 3.5. *Suppose $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$ and $\Pi \in \mathbb{R}^{k\times d}$ satisfies ε-convex hull distortion for X. Then, for any $q \in \mathbb{R}^d$, there exists $z \in \mathbb{R}^k$ such that:*

$$\forall x, y \in X : |\langle z - \Pi x, \Pi(y-x)\rangle - \langle q - x, y - x\rangle| \leqslant 15\varepsilon\|q-x\|\|y-x\|$$
$$\forall x \in X : \|z - \Pi x\| \leqslant (1+8\varepsilon)\|q-x\|.$$

PROOF. Let $x^* = \arg\min_{x\in X}\|q-x\|$. As in [10, 9], we consider a bilinear game where λ is essentially selects which constraint is worst-violated and w is a candidate terminal embedding:

$$\max_{\|\lambda\|_1\leqslant 1}\min_{\|w\|\leqslant\|q-x^*\|}\sum_{x,y\in X}\lambda_{x,y}\left(\frac{\langle w,\Pi(y-x)\rangle - \langle q-x^*, y-x\rangle}{\|q-x^*\|\|y-x\|}\right)$$
$$= \min_{\|w\|\leqslant\|q-x^*\|}\max_{\|\lambda\|_1\leqslant 1}\sum_{x,y\in X}\lambda_{x,y}\left(\frac{\langle w,\Pi(y-x)\rangle - \langle q-x^*, y-x\rangle}{\|q-x^*\|\|y-x\|}\right)$$

where the exchange of the min and max follows from von Neumann's minimax theorem. Considering the first formulation, let $\|\lambda\|_1 \leqslant 1$. We have for such a λ:

$$\min_{\|w\|\leqslant\|q-x^*\|}\sum_{x,y\in X}\lambda_{x,y}\left(\frac{\langle w,\Pi(y-x)\rangle - \langle q-x^*, y-x\rangle}{\|q-x^*\|\|y-x\|}\right)$$
$$= \min_{\|w\|\leqslant\|q-x^*\|}\left(\left\langle\frac{w}{\|q-x^*\|}, \Pi\left(\sum_{x,y\in X}\lambda_{x,y}\cdot\frac{y-x}{\|y-x\|}\right)\right\rangle - \left\langle\frac{q-x^*}{\|q-x^*\|}, \left(\sum_{x,y\in X}\lambda_{x,y}\cdot\frac{y-x}{\|y-x\|}\right)\right\rangle\right)$$
$$\leqslant \min_{\|\widetilde{w}\|\leqslant 1}\left\langle\widetilde{w}, \Pi\left(\sum_{x,y\in X}\lambda_{x,y}\cdot\frac{y-x}{\|y-x\|}\right)\right\rangle + \left\|\sum_{x,y\in X}\lambda_{x,y}\frac{y-x}{\|y-x\|}\right\|$$
$$= -\left\|\Pi\left(\sum_{x,y\in X}\lambda_{x,y}\cdot\frac{y-x}{\|y-x\|}\right)\right\| + \left\|\sum_{x,y\in X}\lambda_{x,y}\cdot\frac{y-x}{\|y-x\|}\right\| \leqslant \varepsilon$$

where the first inequality follows from Cauchy-Schwarz and the final inequality follows from the fact that Π has ε-convex hull distortion. From the previous result, we may conclude that there exists $w \in \mathbb{R}^k$ satisfying:

$$\|w\| \leqslant \|q-x^*\|$$
$$\forall x, y \in X : |\langle w,\Pi(y-x)\rangle - \langle q-x^*, y-x\rangle| \leqslant \varepsilon\|q-x^*\|\|y-x\|.$$

Now consider the vector $z = w + \Pi x^*$. For this z, we have for any $x, y \in X$:

$$\begin{aligned}
&|\langle z-\Pi x, \Pi(y-x)\rangle - \langle q-x, y-x\rangle| \\
&\quad\leqslant |\langle z-\Pi x^*, \Pi(y-x)\rangle - \langle q-x^*, y-x\rangle| + |\langle \Pi(x^*-x), \Pi(y-x)\rangle - \langle x-x^*, y-x\rangle| \\
&\quad\leqslant \varepsilon\|q-x^*\|\|y-x\| + \|x^*-x\|\|y-x\|\left|\left\langle\Pi\frac{x^*-x}{\|x^*-x\|}, \Pi\frac{y-x}{\|y-x\|}\right\rangle - \left\langle\frac{x-x^*}{\|x-x^*\|}, \frac{y-x}{\|y-x\|}\right\rangle\right| \\
&\quad\leqslant \varepsilon\|q-x^*\|\|y-x\| + 6\varepsilon\|x^*-x\|\|y-x\| \\
&\quad= \varepsilon\|y-x\|\left(\|q-x^*\| + 6\|x^*-x\|\right) \\
&\quad\leqslant \varepsilon\|y-x\|\left(\|q-x^*\| + 6(\|x^*-q\| + \|q-x\|)\right)
\end{aligned}$$

$$\leqslant \varepsilon\|y-x\|\left(\|q-x^*\|+12\|q-x\|\right)$$
$$\leqslant 15\varepsilon\|y-x\|\|q-x\|$$

where the second inequality is due to the condition on z, the third inequality is a consequence of Lemma 3.4 and the second to last inequality follows from the fact that $\|q-x^*\| \leqslant \|q-x\|$. This establishes the first claim of the lemma. For the second claim, we have for any $x \in X$:

$$\begin{aligned}
\|z-\Pi x\|^2-\|q-x\|^2 &= \left(\|z-\Pi x^*+\Pi(x^*-x)\|^2-\|q-x^*+(x^*-x)\|^2\right)\\
&= \|z-\Pi x^*\|^2+2\langle z-\Pi x^*,\Pi(x^*-x)\rangle+\|\Pi(x^*-x)\|^2\\
&\quad -\|q-x^*\|^2-2\langle q-x^*,x^*-x\rangle-\|x^*-x\|^2\\
&\leqslant 2(\langle z-\Pi x^*,\Pi(x^*-x)\rangle-\langle q-x^*,x^*-x\rangle)+\|\Pi(x^*-x)\|^2-\|x^*-x\|^2\\
&\leqslant 2(\langle z-\Pi x^*,\Pi(x^*-x)\rangle-\langle q-x^*,x^*-x\rangle)+((1+\varepsilon)^2-1)\|x^*-x\|^2\\
&\leqslant 2(\langle z-\Pi x^*,\Pi(x^*-x)\rangle-\langle q-x^*,x^*-x\rangle)+3\varepsilon\|x^*-x\|^2\\
&\leqslant 2\varepsilon\|q-x^*\|\|x-x^*\|+3\varepsilon\|x^*-x\|^2 \leqslant 4\varepsilon\|q-x^*\|\|q-x\|+12\varepsilon\|q-x\|^2\\
&\leqslant 16\varepsilon\|q-x\|^2
\end{aligned}$$

where the first inequality follows from the fact that $\|z-\Pi x^*\| \leqslant \|q-x^*\|$, the second inequality follows from the fact that Π has ε-convex hull distortion for X, the fourth from our condition on z and the fifth follows from the triangle inequality and the fact that $\|q-x^*\| \leqslant \|q-x\|$. By re-arranging the above inequality and taking square roots, we get:

$$\|z-\Pi x\|^2 \leqslant (1+8\varepsilon)\|q-x\|$$

concluding the proof of the lemma. ■

3.2 Fixed Scale Violator Detection

Note that as a consequence of Theorem 3.2, we may assume that the function can instantiate $(\rho_1, \rho_2, (1+\varepsilon^\dagger))$-Adaptive Approximate Nearest Neighbor (AANN) data structures and (ρ_3, ρ_4)-Approximate Partitioning (AP) data structures. Our data structure for constructing an oracle at a fixed scale is constructed in Algorithm 1 and the query procedure is outlined in Algorithm 2. Algorithm 1 takes as input a set of data points X, a projection matrix Π, a memory parameter ρ_{rep} and a failure probability δ. When we instantiate this data structure, the point sets used to construct it will be the subsets of points corresponding to the nodes of the Partition Tree provided by Theorem 3.2. Algorithm 2 takes as input the data structure constructed by Algorithm 1, the query point q, a candidate solution to the convex program v, an approximate nearest neighbor of q in X and another tolerance parameter whose role will become clear when incorporating this data structure into the reduction.

To state the correctness guarantees for Algorithm 1, we start by introducing some notation. For $x \in X$ and $r > 0$, define the local neighborhood of x as follows: $\mathcal{N}_{\text{loc}}(x,r) = \{y \in X :$

$\|y - x\| \leqslant 2r\}$. For $x \in X$ such that $|\mathcal{N}_{\text{loc}}(x, r)| \geqslant n^{1-\rho_{\text{rep}}}$, our success event is simply that there exists $z \in \mathcal{Z}$ such that $\|z - x\| \leqslant 2r$ and that $\mathcal{D}_z$ is instantiated successfully. For $x \in X$ such that $|\mathcal{N}_{\text{loc}}(x, r)| \leqslant n^{1-\rho_{\text{rep}}}$, the success event is more complicated. Informally, we will require that most of AP data structures produce appropriate partitions for x and furthermore, that each $y \in X$ with $\|y - x\| \leqslant r$ is well represented in these data structures. This is formally described in the following lemma:

LEMMA 3.6. *Given $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$, $r > 0$, $\rho_{\text{rep}} \in [0, 1]$ and $\delta \in (0, 1)$, Algorithm 1 produces a data structure $\mathcal{D}$ with the following guarantees:*

1. *For $x \in X$ such that $|\mathcal{N}_{\text{loc}}(x, r)| \geqslant n^{1-\rho_{\text{rep}}}$, we have that there exists $z \in \mathcal{Z}$ such that $\|x - z\| \leqslant 2r$. That is, x is assigned in the first stage of the algorithm.*
2. *For $x \in X$ such that $|\mathcal{N}_{\text{loc}}(x, r)| < n^{1-\rho_{\text{rep}}}$, we have:*

$$\sum_{i=1}^{l} \mathbf{1}\left\{ \begin{array}{l} \sum\limits_{\substack{y \in X \\ \|y-x\| \geqslant 2r}} \sum\limits_{S \in h_i(x)} \mathbf{1}\{y \in S\} \leqslant O^*(n^{\rho_4}) \text{ and} \\ \sum\limits_{\substack{y \in X \\ \|y-x\| \leqslant 2r}} \sum\limits_{S \in h_i(x)} \mathbf{1}\{y \in S\} \leqslant O^*(n^{(1-\rho_{\text{rep}})+\rho_3}) \end{array} \right\} \geqslant 0.98l$$

$$\sum_{i=1}^{l} \mathbf{1}\left\{ \sum_{S \in h_i(x)} |S \setminus A_{i,S}| \leqslant O^*(n^{\rho_4}) \right\} \geqslant 0.98l$$

$$\forall y \in X \text{ such that } \|y - x\| \leqslant r : \sum_{i=1}^{l} \mathbf{1}\{\exists S \in h_i(x) \text{ such that } y \in S\} \geqslant 0.98l$$

3. *All the AANN data structures instantiated in the algorithm are instantiated successfully; that is, $\{\mathcal{D}_z\}_{z \in \mathcal{Z}}$ and $\{\mathcal{D}_{i,S,w}\}_{i \in [l], S \in \mathcal{D}_i, w \in \mathcal{W}_{i,S}}$ are instantiated successfully.*
4. *Finally, the space complexity of the data structure is $O^*(dn^{1+\rho_1}(n^{\rho_{\text{rep}}} + n^{\rho_3}) \log^2 1/\delta)$*

with probability at least $1 - \delta$.

PROOF. For the third claim, note that the total number of $\mathcal{D}_z$ instantiated is at most n_{rep}. Therefore, the probability that all the $\mathcal{D}_z$ are instantiated correctly is at least $1 - \delta/16$. As for the $\mathcal{D}_{i,S,w}$, note that at most $O^*(lpn^{1+\rho_3})$ many of these are instantiated as there are l AP data structures, each of which has at most $O^*(n^{1+\rho_3})$ subsets and each subset has at most p AANN data structures. Therefore, again by the union bound the probability that each of these are instantiated correctly is at least $1 - \delta/16$. Hence, the probability that all AANN data structures are instantiated correctly is at least $1 - \delta/8$.

For the last claim, note that the space occupied by each of the $\mathcal{D}_z$ data structures is at most $O^*(n^{1+\rho_1})$ and there are n_{rep} of these. The space required to store each of the l AP data structures

is at most $O^*(n^{1+\rho_3})$. Finally, the space occupied by the $\mathcal{D}_{i,S,w}$ data structures is given by:

$$\sum_{i=1}^{l} \sum_{S \in \mathcal{S}_i} p \cdot O^*(|S|^{1+\rho_1} \log 1/\delta) \leqslant O^*(n^{\rho_3+(1+\rho_1)} \log^2 1/\delta)$$

by the fact that $|S| \leqslant n$ for each S, $\sum_{S \in \mathcal{S}_i} |S| \leqslant O^*(n^{1+\rho_3})$ for each l and the convexity of the function $f(x) = x^{1+\rho_3}$. From the previously established bounds, the space complexity follows.

For the first claim, let $x \in X$ such that $|\mathcal{N}_{\text{loc}}(x, r)| \geqslant n^{1-\rho_{\text{rep}}}$ and $K_i = \mathbf{1}\{z_i \in \mathcal{N}_{\text{loc}}(x, r)\}$. We have $\mathbb{P}(K_i = 1) \geqslant n^{-\rho_{\text{rep}}}$. Therefore, we have that $\mathbb{E}\left[K \coloneqq \sum_{i=1}^{n_{\text{rep}}} K_i\right] \geqslant C \log(n/\delta)$ for some large constant C. By noting that $\mathbf{Var}(K) \leqslant \mathbb{E}[K]$, we have by Bernstein's inequality that with probability at least $\delta/(16n)$ that there exists $K_i = 1$ for some $i \in [n_{\text{rep}}]$. Therefore, there exists $z \in \mathcal{Z}$ such that $\|z - x\| \leqslant 2r$. By a union bound, this establishes the first claim with probability at least $1 - \delta/16$.

We now prove each of the three conclusions of the second claim of the lemma separately. First note that from Definition 2.4, the linearity of expectation and the fact that $|\mathcal{N}_{\text{loc}}(x, r)| \leqslant n^{1-\rho_{\text{rep}}}$, we have for any $i \in [l]$:

$$\mathbb{P}\left\{ \sum_{\substack{y \in X \\ \|y-x\| \geqslant 2r}} \sum_{S \in h_i(x)} \mathbf{1}\{y \in S\} \leqslant O^*(n^{\rho_4}) \right\} \geqslant 0.99 \text{ and}$$

$$\mathbb{E}\left\{ \sum_{\substack{y \in X \\ \|y-x\| \leqslant 2r}} \sum_{S \in h_i(x)} \mathbf{1}\{y \in S\} \right\} \leqslant O^*(n^{1-\rho_{\text{rep}}+\rho_3}).$$

By an application of Markov's Inequality on the second inequality, we have by the union bound:

$$\mathbb{P}\left\{ \sum_{\substack{y \in X \\ \|y-x\| \geqslant 2r}} \sum_{S \in h_i(x)} \mathbf{1}\{y \in S\} \leqslant O^*(n^{\rho_4}) \text{ and } \sum_{\substack{y \in X \\ \|y-x\| \leqslant 2r}} \sum_{S \in h_i(x)} \mathbf{1}\{y \in S\} \leqslant O^*(n^{(1-\rho_{\text{rep}})+\rho_3}) \right\} \geqslant 0.985.$$

Letting L_i be the indicator random variable for the above event for x in the data structure $\mathcal{D}_i$, we have by the Chernoff bound that $\mathbb{P}\left\{\sum_{i=1}^{l} L_i \geqslant 0.98l\right\} \geqslant 1 - \delta/(16n)$. Therefore, the first conclusion of the second claim holds for x with probability at least $1 - \delta/(16n)$. A union bound now establishes the lemma for all $x \in X$ with probability at least $1 - \delta/16$.

For the second conclusion of the second claim, let i be such that $L_i = 1$ from the preceding discussion. Let $S \in h_i(x)$ be such that $|\mathcal{N}_{\text{loc}}(x, r) \cap S| > |(X \setminus \mathcal{N}_{\text{loc}}(x, r)) \cap S|$; that is, S is a set in $h_i(x)$ with more points from the local neighborhood of x than far away points. For any such S, the probability that $w_j \in \mathcal{N}_{\text{loc}}(x, r)$ is at least $1/2$. Therefore, we have by the definition of p that with probability at least $1 - \delta^{\ddagger}$ that there exists $w \in \mathcal{W}_{i,S}$ such that $w \in \mathcal{N}_{\text{loc}}(x, r)$. Note that all the points in $\mathcal{N}_{\text{loc}}(x, r) \cap S$ are assigned to w in this case. By the union bound, the probability that this happens for all such $S \in h_i(x)$ is at least $1 - \delta/(ln^{2+(\rho_3+\rho_4)})$. Conversely, for S such that

$|\mathcal{N}_{\text{loc}}(x,r) \cap S| \leqslant |(X \setminus \mathcal{N}_{\text{loc}}(x,r)) \cap S|$, the total number of unassigned points is upper bounded by $2|(X \setminus \mathcal{N}_{\text{loc}}(x,r)) \cap S|$ and by summing over all such S the conclusion follows from the definition of L_i. Therefore, by a union bound, we get that the second conclusion of the second claim holds for $x \in X$ for all i such that $L_i = 1$ with probability at least $1 - \delta/(n^{2+(\rho_3+\rho_4)})$. The conclusion for all $x \in X$ follows from another union bound with probability at least $1 - \delta/16$.

Finally, for the last conclusion of the second claim, let $x, y \in X$ such that $\|x - y\| \leqslant r$ and $M_i = \mathbf{1}\{\exists S \in h_i(x) : y \in S\}$. From Definition 2.4, we have that $\mathbb{P}\{M_i = 1\} \geqslant 0.99$. Therefore, we have by the Chernoff bound that with probability at least $1 - \delta/(16n^3)$ that the conclusion holds for specific $x, y \in X$ satisfying $\|x - y\| \leqslant r$. Through a union bound, the conclusion holds for all $x, y \in X$ with $\|x - y\| \leqslant r$ with probability at least $1 - \delta/(16n)$.

A final union bound over all the events described in the proof gives us the required guarantees on the running of Algorithm 1 with probability at least $1 - \delta/2$. ■

Input: Dataset $X = \{x_i\}_{i=1}^n$, Projection Π, Scale $r > 0$, Repetition ρ_{rep}, Failure Probability δ

$n_{\text{rep}} \leftarrow \Theta(n^{\rho_{\text{rep}}} \log(n/\delta)), \delta^\dagger \leftarrow \Theta\left(\delta/n_{\text{rep}}\right)$

Pick n_{rep} main points, $\mathcal{Z} = \{z_i\}_{i=1}^{n_{\text{rep}}}$, uniformly at random from X

For each $z \in \mathcal{Z}$, instantiate independent $(\rho_1, \rho_2, (1+\varepsilon^\dagger))$-AANN data structures, $\mathcal{D}_z$, with point set $\left\{y_i = \widetilde{v}_{x_i,z}\right\}$ and failure probability $\delta^\dagger$ and for $x \in X$, assign x to z if $\|x - z\| \leqslant 2r$ and add x to A_z

$l \leftarrow \Theta\left(\log(n/\delta)\right), \delta^\ddagger \leftarrow \Theta\left(\delta/ln^{4+2(\rho_3+\rho_4)}\right)$

Instantiate l independent (ρ_3, ρ_4)-AP data structures, $\{\mathcal{D}_i = (h_i, \mathcal{S}_i)\}_{i=1}^l$ with pointset X

for $i \in l$ and $S \in \mathcal{S}_i$ **do**

  $p \leftarrow \Theta\left(\log 1/\delta^\ddagger\right)$

  Pick p points, $\mathcal{W}_{i,S} = \{w_j\}_{j=1}^p$, uniformly at random from S

  For each $w \in \mathcal{W}_{i,S}$, instantiate $(\rho_1, \rho_2, (1+\varepsilon^\dagger))$-AANN data structure, $\mathcal{D}_{i,S,w}$, with point set $\left\{\widetilde{v}_{x,w}\right\}_{x \in S}$ and failure probability $\delta^\ddagger/p$. Assign $x \in S$ to w if $\|x - w\| \leqslant 4r$ and add x to $A_{i,S}$

return

$$\mathcal{D} = \left(X, \mathcal{Z}, \{\mathcal{D}_z, A_z\}_{z \in \mathcal{Z}}, \{\mathcal{D}_i\}_{i=1}^l, \{\mathcal{W}_{i,S}\}_{i \in [l], S \in \mathcal{D}_i}, \left\{\left\{\mathcal{D}_{i,S,w}\right\}_{w \in \mathcal{W}_{i,S}}, A_{i,S}\right\}_{i \in [l], S \in \mathcal{D}_i}\right)$$

Algorithm 1. FixedScaleInstantiation$(X, \Pi, r, \rho_{\text{rep}}, \delta)$

We delay the analysis of the query procedure to the next subsection where we incorporate the fixed scale data structure into a multi-scale separating oracle and conclude the proof of Theorem 3.1.

Input: Data Structure $\mathcal{D}$, Query q, Candidate v, Approximate Nearest Nbr $\hat{x}$, Tolerance $\varepsilon^\dagger$

if $\|v - \Pi\hat{x}\| \geqslant (1 + 10\varepsilon^\dagger)\|q - \hat{x}\|$ **then**
  return $\hat{x}$
else if $\hat{x}$ is assigned to any $z \in \mathcal{Z}$ **then**
  if $\|v - \Pi z\| \geqslant (1 + 10\varepsilon^\dagger)\|q - z\|$ **then**
    return z
  if $|\langle \widetilde{v}_{z,\hat{x}}, \widetilde{v}_{\hat{x}} \rangle| \geqslant 20\varepsilon^\dagger$ **then**
    return $(z, \hat{x})$
  Query $\mathcal{D}_z$ with $\widetilde{v}_z$ and $-\widetilde{v}_z$ to obtain solutions $y_i = \widetilde{v}_{x_i,z}, y_j = \widetilde{v}_{x_j,z}$
  $u = (x_i, z)$ if $|\langle \widetilde{v}_z, y_i \rangle| \geqslant 20\varepsilon^\dagger$, (x_j, z) if $|\langle \widetilde{v}_z, y_j \rangle| \geqslant 20\varepsilon^\dagger$, FAIL otherwise
  return u
else
  for $i \in [l] : \sum_{S \in h_i(\hat{x})} |S \setminus A_{i,S}| \leqslant O^*(n^{\rho_4})$, $\sum_{S \in h_x(\hat{x})} |A_{i,S}| \leqslant O^*(n^{(1-\rho_{\text{rep}})+\rho_3})$ and $S \in h_i(\hat{x})$ **do**
    for $x \notin A_{i,S}$ **do**
      if $|\langle \widetilde{v}_{\hat{x}}, \widetilde{v}_{x,\hat{x}} \rangle| \geqslant 20\varepsilon^\dagger$ **then**
        return $(x, \hat{x})$
    for $w \in \mathcal{W}_{i,S}$ **do**
      if $\|v - \Pi w\| \geqslant (1 + 10\varepsilon^\dagger)\|q - w\|$ **then**
        return w
      if $|\langle \widetilde{v}_{w,\hat{x}}, \widetilde{v}_{\hat{x}} \rangle| \geqslant 20\varepsilon^\dagger$ **then**
        return $(w, \hat{x})$
      Query $\mathcal{D}_{i,S,w}$ with $\widetilde{v}_w$ and $-\widetilde{v}_w$ to obtain solutions $y_i = \widetilde{v}_{x_i,w}, y_j = \widetilde{v}_{x_j,w}$
      $u = (x_i, w)$ if $|\langle \widetilde{v}_{x_i,w}, \widetilde{v}_w \rangle| \geqslant 20\varepsilon^\dagger$, (x_j, w) if $|\langle \widetilde{v}_{x_j,w}, \widetilde{v}_w \rangle| \geqslant 20\varepsilon^\dagger$, FAIL otherwise
      return u
return FAIL

Algorithm 2. FixedScaleQuery$(\mathcal{D}, q, v, \hat{x}, \varepsilon^\dagger)$

3.3 Multi-Scale Reduction and Proof of Theorem 3.1

In this subsection, we wrap up the proof of Theorem 3.1 by incorporating the fixed scale violator detection method from Subsection 3.2 into a multi-scale procedure. We first define the auxiliary data structures that we will assume access to for the rest of the proof:

1. From Lemma 3.3, we may assume $\Pi \in \mathbb{R}^{k\times d}$ satisfies $\varepsilon^{\dagger}$-convex hull distortion for X with $\varepsilon^{\dagger} = \widetilde{c}\varepsilon$ for some suitably small constant $\widetilde{c} > 0$ with $k = O(\varepsilon^{-2}\log n)$
2. From Theorem 3.2, we may assume access to a partition tree $\mathcal{T}$ satisfying Definition 2.7
3. We may assume access to a $(\rho_3, \rho_4, (1+\varepsilon))$-Adaptive Approximate Nearest Neighbor data structure for X built on $\mathcal{T}$ from Theorem 3.2.

Recall the definitions of a partition tree from Subsection 2.2. The first is the partitioning of a data points into connected components of a graph constructed by thresholding the distances between points in the dataset.

DEFINITION 2.5. (Restated) Let $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$ and $r > 0$. We will use $\mathrm{GG}(X, r)$ to denote the graph with nodes indexed by x_i and an edge between x_i and x_j if $\|x_i - x_j\| \leqslant r$. The connected components of this graph will be denoted by $\mathrm{CC}(X, r)$; that is, $\mathrm{CC}(X, r) = \{C_j\}_{j=1}^m$ is a partitioning of X with $x \in C_j$ if and only if $\|x - y\| \leqslant r$ for some $y \in C_j \setminus \{x\}$.

The second is the notion of refinement between partitions.

DEFINITION 2.6. (Restated) For a data set $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$, we say that a partition C *refines* a partition C' if for all $C \in C$, $C \subseteq C'$ for some C' in C'. This will be denoted by $C' \sqsubseteq C$.

Next, define $r_{\mathrm{med}}(X)$ as:

$$r_{\mathrm{med}}(X) = \min\{r > 0 : \exists C \in \mathrm{CC}(X, r) \text{ with } |C| \geqslant n/2\}.$$

The last is the definition of the partition tree itself.

DEFINITION 2.7. (Restated) Given $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$, a Partition Tree of X is a tree, $\mathcal{T}$, whose nodes are labeled by $\left(Z, \{\mathcal{T}_C\}_{C\in C_{\mathrm{low}}}, \mathcal{T}_{\mathrm{rep}}, C_{\mathrm{low}}, C_{\mathrm{high}}, C_{\mathrm{rep}}, r_{\mathrm{apx}}\right)$ where $Z, C_{\mathrm{rep}} \subset X$, $\{\mathcal{T}_C\}_{C\in C_{\mathrm{low}}} \cup \{\mathcal{T}_{\mathrm{rep}}\}$ represent its children, $C_{\mathrm{low}}, C_{\mathrm{high}}$ are partitions of Z and $r_{\mathrm{apx}} > 0$ satisfying the following conditions:

$$\mathrm{CC}(Z, 1000n^2 r_{\mathrm{apx}}) \sqsubseteq C_{\mathrm{high}} \sqsubseteq \mathrm{CC}(Z, r_{\mathrm{apx}}) \sqsubseteq \mathrm{CC}(Z, r_{\mathrm{med}}) \sqsubseteq \mathrm{CC}\left(Z, \frac{r_{\mathrm{apx}}}{10n}\right) \sqsubseteq C_{\mathrm{low}} \sqsubseteq \mathrm{CC}\left(Z, \frac{r_{\mathrm{apx}}}{1000n^3}\right)$$
$$\forall C \in C_{\mathrm{high}} : \exists!\, z \in C_{\mathrm{rep}} \text{ with } z \in C.$$

For the sake of notational simplicity, we will use $\mathcal{T}' \in \mathcal{T}$ both to refer to a node in the tree as well as the subtree rooted at that node and $\mathrm{Size}(\mathcal{T}')$ to refer to the sum of the number of points stored in the subtree $\mathcal{T}'$. The above condition implies $\mathrm{Size}(\mathcal{T}) \leqslant O(n\log n)$ [7].

We first describe the data structure which reduces the multi-scale violator detection problem to the fixed scale setting using the partition tree that is returned by Theorem 3.2. The data structure implementing the multi-scale violator detector is constructed in Algorithm 3 and the corresponding query procedure is described in Algorithm 4. Algorithm 3 takes as input the projection matrix Π, the Partition Tree $\mathcal{T}$, a failure probability δ and a repetition factor ρ_{rep}. The following straightforward lemma is the only guarantee we will require of Algorithm 3:

LEMMA 3.7. *Let $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$, $\mathcal{T}$ be a valid Partition Tree of X, $\delta \in (0,1)$ and $\rho_{\text{rep}} \in [0,1]$. Then, the data structures output by Algorithm 3 on input $X, \mathcal{T}, \delta$ all satisfy the conclusions of Lemma 3.6 with probability at least $1 - \delta$.*

PROOF. The proof follows from a union bound over the nodes of the tree, the l data structures instantiated for each node, the setting of $\delta^\dagger$ in Algorithm 3 and Lemma 3.6. ■

Input: Projection Π, Partition Tree $\mathcal{T}$, Failure Probability δ, Repetition ρ_{rep}

$\delta^\dagger \leftarrow \Theta(\delta/(nd)^{10})$

for $\mathcal{T}' = \{Z, \{\mathcal{T}_C\}_{C \in \mathcal{C}_{\text{low}}}, \mathcal{T}_{\text{rep}}, \mathcal{C}_{\text{low}}, \mathcal{C}_{\text{high}}, C_{\text{rep}}, r_{\text{apx}}\} \in \mathcal{T}$ **do**

  $r_{\text{low}}^{\text{term}} \leftarrow \Theta(r_{\text{apx}}/(nd)^{20}), r_{\text{high}}^{\text{term}} \leftarrow \Theta((nd)^{20} r_{\text{apx}}), \gamma \leftarrow \Theta(1/\log^3 n)$

  for $i = 0, \ldots, \left\lceil \frac{\log r_{\text{high}}^{\text{term}}/r_{\text{low}}^{\text{term}}}{\gamma} \right\rceil = p$ **do**

    Instantiate, $\mathcal{D}_{\mathcal{T}',i} \leftarrow \text{FixedScaleInstantiation}(Z, \Pi, (1+\gamma)^i r_{\text{low}}^{\text{term}}, \rho_{\text{rep}}, \delta^\dagger)$

return $\{\mathcal{D}_{\mathcal{T}',i}\}_{\mathcal{T}' \in \mathcal{T}, i \in \{0\} \cup [p]}$

Algorithm 3. MultiScaleInstantiation$(\Pi, \mathcal{T}, \delta, \rho_{\text{rep}})$

We are now ready to conclude the proof of Theorem 3.1. Suppose $q \in \mathbb{R}^d$ and we are required to construct a valid terminal embedding for q with respect to the point set X. We may assume access to $(\hat{x}, \mathcal{T}')$ for $\hat{x}$ being an approximate nearest neighbor of q in X and $\mathcal{T}' \in \mathcal{T}$ satisfying the conclusion of Theorem 3.2. Also, it suffices to construct a valid terminal embedding for the set of points in $\mathcal{T}'$. We may assume $\mathcal{T}'$ has more than one element as in the one element case, any point on a sphere of radius $\|q - \hat{x}\|$ around $(\Pi\hat{x}, 0)$ suffices from Theorem 3.2. Our algorithm is based on the following set of convex constraints where Z is the set of points in $\mathcal{T}'$:

$$\forall x, y \in Z : |\langle z - \Pi x, \Pi(y - x)\rangle - \langle q - x, y - x\rangle| \leqslant 20\varepsilon^\dagger \|q - x\|\|y - x\|$$
$$\forall x \in Z : \|z - \Pi x\| \leqslant (1 + 10\varepsilon^\dagger)\|q - x\|. \qquad \text{(Req)}$$

Let K denote the convex subset of $\mathbb{R}^k$ satisfying the above set of constraints. Let $\hat{r} = \|q - \hat{x}\|$. By Lemma 3.5 and Cauchy-Schwarz, K is non-empty and there exists $\widetilde{z} \in K$ with $\mathbb{B}(\widetilde{z}, \varepsilon^\dagger \hat{r}) \subset K$ (there exists $\widetilde{z}$ satisfying the above constraints with the right-hand sides replaced by $15\varepsilon^\dagger$ and

$8\varepsilon^\dagger$ respectively). Also, note that $K \subseteq \mathbb{B}(\Pi\hat{x}, 2\hat{r})$ from the above set of constraints. While one can show a feasible point in K can be used to construct a valid terminal embedding, we construct a slightly weaker oracle. We construct an oracle $\mathcal{O}$ satisfying:

1. If $\mathcal{O}(v)$ outputs FAIL, v can be extended to a terminal embedding
2. Otherwise $\mathcal{O}(v)$ outputs a valid separating hyperplane for K.

To utilize the above oracle for constructing our terminal embeddings, we require the following guarantees on the ellipsoid algorithm which essentially states that for a convex set containing a ball of substantially large radius with an oracle that is guaranteed to output a correct separating hyperplane or FAIL, a point for which the oracle outputs FAIL may be found quickly. In the context of the proof, $\mathcal{O}$ will be our oracle and the convex body will be the feasible points of Req. The proof the statement is deferred to Appendix B.1.

LEMMA 3.8. *Suppose $\varepsilon > 0$ and $K \subset \mathbb{R}^d$ be a closed convex set such that there exists $x^* \in K$ such that for all $y \in \mathbb{B}(x^*, \varepsilon)$, $y \in K$. Furthermore, let $x \in \mathbb{R}^d$ and $R > 0$ be such that $K \subset \mathbb{B}(x, R)$. Suppose further that $\mathcal{O}$ satisfies for any input z, $\mathcal{O}$:*

1. *Outputs $v \neq 0$ such that for all $y \in K$, we have $\langle v, y - x\rangle \geqslant 0$ or*
2. *Outputs* FAIL.

Then, Algorithm 9 when instantiated with x, R and $\mathcal{O}$, outputs $\hat{x}$ satisfying $\mathcal{O}(\hat{x}) =$ FAIL. Furthermore, the number of iterations of the algorithm is bounded by $O\left(d^2 \log \frac{R}{\varepsilon}\right)$ and hence the total computational complexity is bounded by $O\left(d^4 \log \frac{R}{\varepsilon}\right)$.

From Lemma 3.8 and the fact that K contains a ball of radius $\varepsilon^\dagger \hat{r}$ and is contained in a ball of radius $2\hat{r}$, we see that there is a procedure which with $O^*(1)$ many queries to $\mathcal{O}$ and $O^*(1)$ total additional computation, outputs a point v^* for which $\mathcal{O}$ outputs FAIL. Our oracle implementing this property is defined simply by Algorithm 4. Through the rest of this subsection, we focus our attention on proving the correctness of the oracle.

Let $\mathcal{T}' = \{Z, \{\mathcal{T}_C\}_{C \in \mathcal{C}_{\text{low}}}, \mathcal{T}_{\text{rep}}, \mathcal{C}_{\text{low}}, \mathcal{C}_{\text{high}}, \mathcal{C}_{\text{rep}}, r_{\text{apx}}\}$ and $m = |Z|$. First, we focus on the easy case when Algorithm 4 does not output FAIL.

LEMMA 3.9. *If Algorithm 4 does not output* FAIL *on v, it outputs a separating hyperplane for v from K.*

PROOF. Note that this only happens when some x or pair (x, y) is returned by Algorithm 2. Now, Algorithm 2 may return an x in one of two ways:

Case 1: $\|v - \Pi x\| \geqslant (1 + 10\varepsilon^\dagger)\|q - x\|$ for x returned by the algorithm. In this case, the correctness of the procedure trivially follows from Req.

Case 2: For $x, y \in Z$, the vectors $v_1 = \widetilde{v}_x$ and $v_2 = \widetilde{v}_{y,x}$ satisfy $|\langle v_1, v_2\rangle| \geqslant 20\varepsilon^\dagger$. We have:

$$|\langle q - x, y - x\rangle - \langle v - \Pi x, \Pi(y - x)\rangle| \geqslant 20\varepsilon^\dagger \|(q - x, -(v - \Pi x))\| \|(y - x, \Pi(y - x))\|$$

$$\geqslant 20\varepsilon^{\dagger}\|q-x\|\|y-x\|$$

which is again a violator of Req which proves correctness in this case as well.

This concludes the proof of the lemma. ■

Next, we consider the alternate case where Algorithm 4 outputs FAIL, where we show that the input v can be used to construct a valid terminal embedding for q with respect to Z and consequently for X from the guarantees of Theorem 3.2. We recall the following crucial parameters from Algorithm 3 where r_{apx} is defined in Definition 2.7

$$r_{\text{low}}^{\text{term}} = \Theta(r_{\text{apx}}/(nd)^{20}) \qquad r_{\text{high}}^{\text{term}} = \Theta((nd)^{20} r_{\text{apx}}). \tag{RSET}$$

LEMMA 3.10. *If Algorithm 4 outputs* FAIL *on* v, $z_q = (v, \sqrt{\max(0, \|q-\hat{x}\|^2 - \|v - \Pi\hat{x}\|^2)})$ *satisfies:*

$$\forall z \in Z : (1-2000\varepsilon)\|q-z\| \leqslant \|z_q - (\Pi z, 0)\| \leqslant (1+2000\varepsilon)\|q-z\|.$$

PROOF. Note that since Algorithm 2 returned FAIL for all fixed scale data structures, we must have $\|v-\Pi\hat{x}\| \leqslant (1+10\varepsilon^{\dagger})\|q-\hat{x}\|$. As a consequence, we get that $\|z_q-(\Pi\hat{x},0)\| \leqslant (1+10\varepsilon^{\dagger})\|q-\hat{x}\|$. Now, we need to show that z_q is a valid terminal embedding of q for an arbitrary $x \in Z$. We first consider the case where $\|x-\hat{x}\| \geqslant 0.5 r_{\text{high}}^{\text{term}}$ as defined in Algorithm 3 for the node $\mathcal{T}'$. For this point, we have:

$$\|q-x\| \geqslant \|\hat{x}-x\| - \|q-\hat{x}\| \geqslant \left(1 - \frac{1}{(10nd)^9}\right)\|\hat{x}-x\|$$

where the last inequality comes from the condition on $\|q-\hat{x}\|$ from Theorem 3.2 and the definition of $r_{\text{high}}^{\text{term}}$. Now, we get:

$$\begin{aligned}
|\|x-q\| - \|z_q - (\Pi x, 0)\|| &\leqslant |\|\hat{x}-x\| - \|\Pi(\hat{x}-x)\|| + \|\hat{x}-q\| + \|z_q-(\Pi\hat{x},0)\| \\
&\leqslant \varepsilon^{\dagger}\|\hat{x}-x\| + (2+\varepsilon)\|\hat{x}-q\| \\
&\leqslant 2\varepsilon^{\dagger}\|q-x\| + (2+\varepsilon)\|\hat{x}-q\| \\
&\leqslant \|q-x\|\left(2\varepsilon^{\dagger} + 3\frac{\|\hat{x}-q\|}{\|q-x\|}\right) \leqslant \varepsilon\|x-q\|
\end{aligned}$$

where the second inequality follows from the fact that Π as $\varepsilon^{\dagger}$-convex full distortion for X, the third from the previous display and last inequality from the fact that $\|\hat{x}-q\| \leqslant O((nd)^{10} r_{\text{apx}})$ (Theorem 3.2) and $\|q-x\| \geqslant \|\hat{x}-x\|/2 \geqslant r_{\text{high}}^{\text{term}}/4$ and the values of r_{apx} and $r_{\text{high}}^{\text{term}}$ (RSET).

We now consider the alternative case where $\|x-\hat{x}\| \leqslant 0.5 r_{\text{high}}^{\text{term}}$. In this case, from the definition of p in Algorithm 3, let i^* be the smallest $i \in [p]$ such that $\|x-\hat{x}\| \leqslant (1+\gamma)^{i^*} r_{\text{low}}^{\text{term}}$. Note that i^* is finite from our condition on $\|x-\hat{x}\|$. For this i^*, consider the fixed scale data structure, $\mathcal{D}_{\mathcal{T}',i^*} = \left(Z, \mathcal{Z}, \{\mathcal{D}_z, A_z\}_{z\in\mathcal{Z}}, \{\mathcal{D}_i\}_{i=1}^{l}, \{\mathcal{W}_{i,S}\}_{i\in[l],S\in\mathcal{D}_i}, \left\{\{\mathcal{D}_{i,S,w}\}_{w\in\mathcal{W}_{i,S}}, A_{i,S}\right\}_{i\in[l],S\in\mathcal{D}_i}\right)$

constructed in Algorithm 3. Letting $\widetilde{r} = (1+\gamma)^{i^*} r_{\text{low}}^{\text{term}}$, we prove the lemma in two cases corresponding to the structure of $\mathcal{D}_{\mathcal{T}',i^*}$ as described in Lemma 3.6:

Case 1: $\hat{x} \in A_z$ for some $z \in \mathcal{Z}$. Note that this necessarily happens if $|\mathcal{N}_{\text{loc}}(\hat{x}, \widetilde{r})| \geqslant m^{1-\rho_{\text{rep}}}$.

Case 2: $\hat{x} \notin A_z$ for all $z \in \mathcal{Z}$. Recall, this implies $|\mathcal{N}_{\text{loc}}(\hat{x}, \widetilde{r})| < m^{1-\rho_{\text{rep}}}$.

In either case, the following two simple claims will be crucial in bounding the error terms:

CLAIM 3.11. *For all $x \in Z$, we have:*

$$\|x - q\| \geqslant \frac{1}{(2+\varepsilon+o^*(1))}\|\hat{x} - x\|.$$

Proof. We have

$$\|\hat{x} - x\| \leqslant \|\hat{x} - q\| + \|x - q\| \leqslant (2+\varepsilon+o^*(1))\|x - q\|.$$

By re-arranging the above inequality, we obtain our result. ◆

CLAIM 3.12. *We have:*

$$\|q - x\| \geqslant \frac{5}{12}\widetilde{r}.$$

Proof. If $i^* = 0$, we have:

$$\|q - x\| \geqslant (1+\varepsilon+o^*(1))^{-1}\|q - \hat{x}\| \geqslant (10nd)^5 r_{\text{low}}^{\text{term}}$$

from our assumption on $\|q - \hat{x}\|$ (Theorem 3.2). When $i^* > 0$, we have $(1+\gamma)^{-1}\widetilde{r} \leqslant \|\hat{x} - x\| \leqslant \widetilde{r}$ and the conclusion follows from Claim 3.11. ◆

In the first case, we get that $\hat{x}$ is assigned to some $z \in \mathcal{Z}$ with $\|\hat{x} - z\| \leqslant 2\widetilde{r}$ which implies by the triangle inequality $\|x - z\| \leqslant 3\widetilde{r}$.

We now prove another claim we will use through the rest of this proof:

CLAIM 3.13. *For all $w \in Z$, let $\widetilde{w} = \widetilde{v}_{w,\hat{x}}$ and $\widetilde{v} = \widetilde{v}_{\hat{x}}$. If $|\langle \widetilde{w}, \widetilde{v}\rangle| \leqslant 20\varepsilon^\dagger$:*

$$\|z_q - (\Pi w, 0)\| - \|w - q\| \leqslant 400\varepsilon^\dagger \|w - q\|.$$

Proof. We have:

$$\begin{aligned}
|\|w-q\|^2 - \|(\Pi w, 0) - z_q\|^2| &\leqslant |\|w-\hat{x}\|^2 - \|\Pi(w-\hat{x})\|^2| + |\|q-\hat{x}\|^2 - \|z_q - (\Pi\hat{x}, 0)\|^2| \\
&\quad + 2|\langle w-\hat{x}, q-\hat{x}\rangle - \langle \Pi(w-\hat{x}), v - \Pi\hat{x}\rangle| \\
&\leqslant 3\varepsilon^\dagger\|w-\hat{x}\|^2 + 25\varepsilon^\dagger\|q-\hat{x}\|^2 + 100\varepsilon^\dagger\|w-\hat{x}\|\|q-\hat{x}\| \\
&\leqslant 750\varepsilon^\dagger\|w-q\|^2
\end{aligned}$$

where the second inequality follows from the fact that Π satisfies $\varepsilon^\dagger$-convex hull distortion for X, our claim on $\|z_q - (\Pi\hat{x}, 0)\|$ and our condition on $\langle \widetilde{w}, \widetilde{v}\rangle$ while the final inequality follows from Claim 3.11 and the fact that $\hat{x}$ is a $(1+\varepsilon+o^*(1))$-Approximate Nearest Neighbor of q. By factorizing the LHS and dividing by $\|w-q\|$, we get the desired result. ◆

Returning to our proof, recall $\|x - z\| \leqslant 3\widetilde{r}$ and from Claim 3.12, $\|q - x\| \geqslant \frac{5}{12}\widetilde{r}$. We claim since $\mathcal{D}_{\mathcal{T}',i^*}$ returns FAIL that $|\langle \widetilde{v}_{x,z}, \widetilde{v}_z \rangle| \leqslant 4\varepsilon$. To see this, suppose $\langle \widetilde{v}_{x,z}, \widetilde{v}_z \rangle \geqslant 4\varepsilon$. In this case, we have:

$$\|\widetilde{v}_{x,z} - \widetilde{v}_z\|^2 \leqslant 2 - 8\varepsilon \implies \|\widetilde{v}_{x,z} - \widetilde{v}_z\| \leqslant (1 - 2\varepsilon)\sqrt{2}.$$

Therefore, $\mathcal{D}_z$ on input $\widetilde{v}_z$ returns y such that $\|y - \widetilde{v}_z\| \leqslant (1 + \varepsilon + o^*(1))(1 - 2\varepsilon)\sqrt{2}$ follows from the fact that $\mathcal{D}_z$ is successfully instantiated. From this, we get that $\|y - \widetilde{v}_z\| \leqslant \sqrt{2}(1 - 0.8\varepsilon)$. From this expression, we obtain:

$$\|y - \widetilde{v}_z\|^2 = 2 - 2\langle y, \widetilde{v}_z \rangle \implies \langle y, \widetilde{v}_z \rangle \geqslant 0.8\varepsilon$$

which contradicts the assumption that $\mathcal{D}_{\mathcal{T}',i^*}$ returns FAIL. The proof for $\langle \widetilde{v}_z, \widetilde{v}_{x,z} \rangle \leqslant -4\varepsilon$ is similar by replacing $\widetilde{v}_z$ by $-\widetilde{v}_z$ in the above proof and using the fact that we query $\mathcal{D}_z$ with $-\widetilde{v}_z$ as well. Hence, we have $|\langle \widetilde{v}_z, \widetilde{v}_{x,z} \rangle| \leqslant 4\varepsilon$. Finally, bound the deviation of $\|z_q - (\Pi x, 0)\|$ from $\|q - x\|$:

$$\begin{aligned}
|\|x - q\|^2 - \|(\Pi x, 0) - z_q\|^2| &\leqslant |\|x - z\|^2 - \|\Pi(x - z)\|^2| + |\|q - z\|^2 - \|z_q - (\Pi z, 0)\|^2| \\
&\quad + 2|\langle x - z, q - z \rangle - \langle \Pi(x - z), v - \Pi z \rangle| \\
&\leqslant 3\varepsilon^\dagger \|x - z\|^2 + 750\varepsilon^\dagger \|q - z\|^2 + 2 \cdot 4\varepsilon \cdot \sqrt{5} \cdot \|x - z\| \|q - z\| \\
&\leqslant 3\varepsilon^\dagger \|x - z\|^2 + 750\varepsilon^\dagger \|q - z\|^2 + 20\varepsilon \|x - z\| \|q - z\| \\
&\leqslant 27\varepsilon^\dagger \widetilde{r}^2 + 750\varepsilon^\dagger (\|q - x\| + \|x - z\|)^2 \\
&\quad + 20\varepsilon \|x - z\| (\|q - x\| + \|x - z\|) \\
&\leqslant 27\varepsilon^\dagger \widetilde{r}^2 + 1500\varepsilon^\dagger (\|q - x\|^2 + \|x - z\|^2) + 60\varepsilon\widetilde{r}(\|q - x\| + 3\widetilde{r}) \\
&\leqslant 1500\varepsilon \|q - x\|^2
\end{aligned}$$

where the second inequality follows from the fact that Π has $\varepsilon^\dagger$-convex hull distortion for X, Claim 3.13 and the fact that $|\langle \widetilde{v}_z, \widetilde{v}_{x,z} \rangle| \leqslant 4\varepsilon$, the fourth inequality follows from the fact that $\|x - z\| \leqslant 3\widetilde{r}$ and the last and second-to-last inequalities follow from the fact that $(a + b)^2 \leqslant 2(a^2 + b^2)$ and Claim 3.12. Factorizing the LHS now establishes the lemma in this case.

We now consider the alternate case where $|\mathcal{N}_{\text{loc}}(\hat{x}, \widetilde{r})| < m^{1-\rho_{\text{rep}}}$. For the fixed scale data structure, $\mathcal{D}_{\mathcal{T}',i^*}$, recall that we have from Lemma 3.6 that:

$$\sum_{i=1}^{l} \mathbf{1}\left\{ \begin{array}{l} \displaystyle\sum_{\substack{y \in Z \\ \|y - \hat{x}\| \geqslant 2\widetilde{r}}} \sum_{S \in h_i(\hat{x})} \mathbf{1}\{y \in S\} \leqslant O^*(m^{\rho_4}) \text{ and} \\ \displaystyle\sum_{\substack{y \in Z \\ \|y - \hat{x}\| \leqslant 2\widetilde{r}}} \sum_{S \in h_i(\hat{x})} \mathbf{1}\{y \in S\} \leqslant O^*(m^{(1-\rho_{\text{rep}})+\rho_3}) \end{array} \right\} \geqslant 0.98l$$

$$\sum_{i=1}^{l} \mathbf{1}\left\{ \sum_{S \in h_i(\hat{x})} |S \setminus A_{i,S}| \leqslant O^*(m^{\rho_4}) \right\} \geqslant 0.98l$$

$$\sum_{i=1}^{l} \mathbf{1}\{\exists S \in h_i(\hat{x}) \text{ such that } x \in S\} \geqslant 0.98l.$$

In particular, we have by the union bound, there exists a set $\mathcal{J} \subset [l]$ with $|\mathcal{J}| \geqslant 0.9l$ satisfying the indicators in all three of the above equations. For any $j \in \mathcal{J}$, from Algorithm 2, all the subsets in $h_j(\hat{x})$ are fully explored. For some $j \in \mathcal{J}$, consider $S \in h_j(\hat{x})$ such that $x \in S$. Now, if $x \notin A_{j,S}$ or $x \in \mathcal{W}_{j,S}$, conclusion of the lemma follows from Claim 3.13. Hence, the only case left to consider is the case where $x \in A_{j,S}$. Let x be assigned to $w \in \mathcal{W}_{j,S}$ such that $\|x - w\| \leqslant 4\widetilde{r}$. In this case, we proceed identically to the previous case where $\hat{x}$ is assigned to some $z \in \mathcal{Z}$. Since $\mathcal{D}_{\mathcal{T}',i^*}$ returns FAIL, $|\langle \widetilde{v}_w, \widetilde{v}_{x,w}\rangle| \leqslant 4\varepsilon$. To see this, assume $\langle \widetilde{v}_w, \widetilde{v}_{x,w}\rangle \geqslant 4\varepsilon$ and we have:

$$\|\widetilde{v}_w - \widetilde{v}_{x,w}\|^2 \leqslant 2 - 8\varepsilon \implies \|\widetilde{v}_w - \widetilde{v}_{x,w}\| \leqslant (1-2\varepsilon)\sqrt{2}.$$

Therefore, $\mathcal{D}_{j,S,w}$ on input $\widetilde{v}_w$ returns y such that $\|y - \widetilde{v}_w\| \leqslant (1 + \varepsilon + o^*(1))(1-2\varepsilon)\sqrt{2}$ follows from the fact that $\mathcal{D}_{j,S,w}$ is successfully instantiated. From this, we get $\|y - \widetilde{v}_w\| \leqslant \sqrt{2}(1 - 0.8\varepsilon)$. We obtain:

$$\|y - \widetilde{v}_w\|^2 = 2 - 2\langle y, \widetilde{v}_w\rangle \implies \langle y, \widetilde{v}_w\rangle \geqslant 0.8\varepsilon$$

which contradicts the assumption that $\mathcal{D}_{\mathcal{T}',i^*}$ returns FAIL. The proof when $\langle \widetilde{v}_w, \widetilde{v}_{x,w}\rangle \leqslant -4\varepsilon$ is similar by replacing $\widetilde{v}_w$ by $-\widetilde{v}_w$ and using the fact that we query $\mathcal{D}_{j,S,w}$ with $-\widetilde{v}_w$ as well. Hence, we have that $|\langle \widetilde{v}_w, \widetilde{v}_{x,w}\rangle| \leqslant 4\varepsilon$. As before, we bound the deviation of $\|z_q - (\Pi x, 0)\|$ from $\|q - x\|$:

$$\begin{aligned}
|\|x-q\|^2 - \|(\Pi x, 0) - z_q\|^2| &\leqslant |\|x-w\|^2 - \|\Pi(x-w)\|^2| + |\|q-w\|^2 - \|z_q - (\Pi w, 0)\|^2| \\
&\quad + 2|\langle x - w, q - w\rangle - \langle \Pi(x-w), v - \Pi w\rangle| \\
&\leqslant 3\varepsilon^{\dagger}\|x-w\|^2 + 750\varepsilon^{\dagger}\|q-w\|^2 + 2 \cdot 4\varepsilon \cdot \sqrt{5} \cdot \|x-w\|\|q-w\| \\
&\leqslant 3\varepsilon^{\dagger}\|x-w\|^2 + 750\varepsilon^{\dagger}\|q-w\|^2 + 20\varepsilon\|x-w\|\|q-w\| \\
&\leqslant 48\varepsilon^{\dagger}\widetilde{r}^2 + 750\varepsilon^{\dagger}(\|q-x\| + \|x-w\|)^2 \\
&\quad + 20\varepsilon\|x-w\|(\|q-x\| + \|x-w\|) \\
&\leqslant 48\varepsilon^{\dagger}\widetilde{r}^2 + 1500\varepsilon^{\dagger}(\|q-x\|^2 + \|x-w\|^2) + 80\varepsilon\widetilde{r}(\|q-x\| + 4\widetilde{r}) \\
&\leqslant 2000\varepsilon\|q-x\|^2
\end{aligned}$$

where the second inequality follows from the fact that Π has $\varepsilon^{\dagger}$-convex hull distortion for X, Claim 3.13 and the fact that $|\langle \widetilde{v}_w, \widetilde{v}_{x,w}\rangle| \leqslant 4\varepsilon$, the fourth inequality follows from the fact that $\|x - w\| \leqslant 4\widetilde{r}$ and the last and second-to-last inequalities follow from the fact that $(a+b)^2 \leqslant 2(a^2+b^2)$ and Claim 3.12. Factorizing the LHS now establishes the lemma in this case concluding the proof of the lemma. ■

Having proved the correctness of Algorithm 4, we finally bound its runtime. Before doing so, we require a data structure allowing a weak form of dimensionality reduction that allows a speed-up of our algorithm. Note that standard techniques for dimensionality reduction may

not be used in our setting as the queries may be chosen based on the data structure itself. For example, for the Johnson-Lindenstrauss lemma, the queries may be chosen orthogonal to the rows of the projection matrix violating its correctness guarantees. We first define an approximate inner product condition below for a matrix Π and $x, y, z \in \mathbb{R}^d$ and $\varepsilon > 0$

$$|\langle \Pi(x-z), \Pi(y-z)\rangle - \langle x-z, y-z\rangle| \leqslant \varepsilon \|x-z\|\|y-z\|. \tag{AP-IP}$$

For $x, y, z \in \mathbb{R}^d$ and $\varepsilon > 0$, we say a matrix Π satisfies $AP\text{-}IP(\varepsilon, x, y, z)$ if it satisfies AP-IP. For a dataset, $X \subset \mathbb{R}^d$, Π satisfies $AP\text{-}IP(\varepsilon, X)$ if it satisfies $AP\text{-}IP(\varepsilon, x, y, z)$ for all $x, y, z \in X$.

We utilize the following theorem which states that if one initializes $\widetilde{O}(d)$ many independent JL-sketches, for any input query, most of them satisfy the AP-IP condition for the dataset X augmented with the input query. This essentially allows us to reduce the dimensionality of the search problem by instead verifying the inner product condition in Gen-Prog on a randomly drawn sketch from this set as opposed to the d-dimensional space containing the dataset.

THEOREM 3.14. *Let $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$, $\varepsilon \in \left(\frac{1}{\sqrt{nd}}, 1\right)$ and $\delta \in (0,1)$. Furthermore, let $\{\Pi^i\}_{i=1}^m \subset \mathbb{R}^{k\times d}$ for $m \geqslant \Omega((d + \log 1/\delta)\log nd)$ and $k \geqslant \Omega\left(\frac{\log n}{\varepsilon^2}\right)$ with $\Pi^i_{j,l} \overset{iid}{\sim} \mathcal{N}(0, 1/k)$. Then, we have:*

$$\forall q \in \mathbb{R}^d : \sum_{i=1}^m \mathbf{1}\left\{\Pi^i \text{ satisfies } AP\text{-}IP(\varepsilon, X \cup \{q\})\right\} \geqslant 0.95m \tag{JL-Rep}$$

with probability at least $1 - \delta$.

We now use Theorem 3.14 to bound the runtime of Algorithm 4.

LEMMA 3.15. *Algorithm 4 is implementable in time $O^*(d + n^{\rho_2} + n^{\rho_4} + n^{\rho_4 + (1+\rho_3-\rho_4-\rho_{\text{rep}})\rho_2})$ with probability at least $1 - n^{-10}$.*

PROOF. Note that we may restrict ourselves to bounding the runtime of Algorithm 2 as we query at most $O^*(1)$ many of them. For a single fixed scale data structure, $\mathcal{D}_{\mathcal{T}',i}$ being queried, computing the sets $h_i(\hat{x})$ takes time $O^*(dn^{\rho_4})$. If $\hat{x}$ is assigned to a point z, the nearest neighbor procedure takes time $O^*(dn^{\rho_2})$. Otherwise, processing the unassigned points takes time $O^*(dn^{\rho_4})$ and for the assigned points, there are at most $O^*(n^{1-\rho_{\text{rep}}+\rho_3})$ in at most $O^*(n^{\rho_4})$ many sets. From the concavity of the function $f(x) = x^{\rho_2}$, we get that the maximum time taken to query all of these nearest neighbor data structures is at most $O^*(dn^{\rho_4+(1+\rho_3-\rho_4-\rho_{\text{rep}})\rho_2})$. We now discuss how to decouple the dimensionality term from the term dependent on n.

As explained in Subsection 2.3, it suffices for our argument to use $\Pi'(x-y)$ instead of $x-y$ as the first component in the construction of the vectors used to instantiate the data structures $\mathcal{D}_z$ and $\mathcal{D}_{i,S,w}$ in Algorithm 1 for any Π' satisfying:

$$\forall x, y, z \in X \cup \{q\} : |\langle \Pi'(x-z), \Pi'(y-z)\rangle - \langle x-z, y-z\rangle| \leqslant o^*(1) \cdot \|x-z\|\|y-z\|.$$

From Theorem 3.14, we get that an instantiation of $m = \widetilde{\Theta}(d)$ JL sketches, $\{\Pi^i\}_{i\in[m]}$, with $\Theta(\log n \log\log n)$ rows each satisfies the above condition for at least 95% of them with probability at least $1 - n^{-10}$. To construct our final violator detection subroutine, we instantiate m copies of our violator detection algorithm for the projections of the data points with respect to each of the sketches Π^i. At query time, we simply sample $\Theta(\log n)$ many of these sketches for a possible violator. We then check the validity of each of the returned candidates which can be done in time $\widetilde{O}(d)$. Since, 95% of the sketches satisfy the above condition, at least one of the sampled sketches will with probability at least $1 - n^{-10}$ and hence, satisfies the guarantees required of our oracle. ■

Proof of Theorem 3.1: Hence, Lemmas 3.9, 3.10 and 3.15 along with Theorems 3.2 and 3.14 and the preceding discussion conclude the proof of Theorem 3.1 via a union bound over the $O^*(1)$ rounds of the Ellipsoid algorithm. ■

Input: Data Structure $\left\{\mathcal{D}_{\mathcal{T}',i}\right\}_{\mathcal{T}'\in\mathcal{T},i\in\{0\}\cup[p]}$, Tree $\mathcal{T}$, Query q, Candidate v, AANN Output $(\hat{x},\mathcal{T}')$, Tolerance $\varepsilon^\dagger$

for $i = 0\ldots,p$ **do**
  if $x = \text{FixedScaleQuery}(\mathcal{D}_{\mathcal{T}',i}, q, v, \hat{x}, \varepsilon^\dagger) \neq \text{FAIL}$ **then**
    return x
return FAIL

Algorithm 4. MultiScaleQuery$(\left\{\mathcal{D}_{\mathcal{T}',i}\right\}_{\mathcal{T}'\in\mathcal{T},i\in\{0\}\cup[p]}, \mathcal{T}, q, v, (\hat{x},\mathcal{T}'), \varepsilon^\dagger)$

4. Adaptive Approximate Nearest Neighbor

In this section, we prove the following theorem regarding the existence of adaptive algorithms for approximate nearest neighbor search based on adapting ideas from [4] to the nearest neighbor to near neighbor reduction. These data structures will play a crucial role in designing algorithms to compute terminal embeddings. Note again that the probability of success only depends on the random choices made by data structure at *query-time* which may be made arbitrarily high by repetition assuming successful instantiation of the data structure. Also, the second property of the tuple returned by the data structure is irrelevant for computing an approximate nearest neighbor but plays a crucial part in our algorithm for computing terminal embeddings.

THEOREM 3.2. (Restated) *Let $c > 1$ and $\rho_u, \rho_c > 0$. Then, there is a randomized procedure which when instantiated with a dataset $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$ and a (ρ_u, ρ_c, c)-Approximate Near Neighbor data structure (Definition 2.8) produces a data structure, $(\mathcal{D}, \mathcal{T})$, satisfying:*

1. *Given any $q \in \mathbb{R}^d$, $\mathcal{D}$ produces $(\hat{x} \in X, \mathcal{T}_{\text{res}} \in \mathcal{T})$ satisfying:*
   a. $\|q - \hat{x}\| \leqslant \min_{x \in X}(1 + o^*(1))c\|q - x\|$
   b. $\hat{x} \in \mathcal{T}_{\text{res}}$
   c. *Furthermore, let $\mathcal{Y} = \{y_i\}_{i=1}^n \subset \mathbb{R}^k$ satisfying for some $\varepsilon^\dagger \in \left(\frac{1}{\sqrt{d}}, 1\right)$:*

$$\forall i, j \in [n] : (1 - \varepsilon^\dagger)\|x_i - x_j\| \leqslant \|y_i - y_j\| \leqslant (1 + \varepsilon^\dagger)\|x_i - x_j\|$$

   and for $\mathcal{T}_{\text{res}} = \left(Z, \{\mathcal{T}_C\}_{C \in C_{\text{low}}}, \mathcal{T}_{\text{rep}}, C_{\text{low}}, C_{\text{high}}, C_{\text{rep}}, r_{\text{apx}}\right)$, let $y \in \mathbb{R}^k$ satisfy for $\varepsilon^\ddagger \in [\varepsilon^\dagger, 1)$:

$$\forall x_i \in Z : (1 - \varepsilon^\ddagger)\|q - x_i\| \leqslant \|y - y_i\| \leqslant (1 + \varepsilon^\ddagger)\|q - x_i\|.$$

   Then:

$$\forall x_i \in X : \left(1 - (1 + o^*(1))\,\varepsilon^\ddagger\right)\|q - x_i\| \leqslant \|y - y_i\| \leqslant \left(1 + (1 + o^*(1))\,\varepsilon^\ddagger\right)\|q - x_i\|.$$

   and if $|Z| > 1$:

$$\Omega\left(\frac{1}{(nd)^{10}} r_{\text{apx}}\right) \leqslant \|q - \hat{x}\| \leqslant O((nd)^{10} r_{\text{apx}})$$

   with probability at least $1 - 1/\text{poly}(n)$
2. *$\mathcal{T}$ is a valid Partition Tree of X (Definition 2.7)*
3. *The space complexity of $\mathcal{D}$ is $O^*(dn^{1+\rho_u} \log 1/\delta)$*
4. *The runtime of $\mathcal{D}$ on any $q \in \mathbb{R}^d$ is at most $O^*(d + n^{\rho_c})$*

with probability $1 - \delta$.

For to establish Theorem 3.2, we require a construction of a partition tree. Recall again the definitions relevant to a partition tree from Subsection 2.2. The first is the partitioning of a data points into connected components of a graph constructed by thresholding the distances between points in the dataset.

DEFINITION 2.5. (Restated) Let $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$ and $r > 0$. We will use $\text{GG}(X, r)$ to denote the graph with nodes indexed by x_i and an edge between x_i and x_j if $\|x_i - x_j\| \leqslant r$. The connected components of this graph will be denoted by $\text{CC}(X, r)$; that is, $\text{CC}(X, r) = \{C_j\}_{j=1}^m$ is a partitioning of X with $x \in C_j$ if and only if $\|x - y\| \leqslant r$ for some $y \in C_j \setminus \{x\}$.

The second is the notion of refinement between partitions.

DEFINITION 2.6. (Restated) For a data set $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$, we say that a partition C *refines* a partition C' if for all $C \in C$, $C \subseteq C'$ for some C' in C'. This will be denoted by $C' \sqsubseteq C$.

Next, define $r_{\text{med}}(X)$ as:

$$r_{\text{med}}(X) = \min\{r > 0 : \exists C \in \text{CC}(X, r) \text{ with } |C| \geqslant n/2\}.$$

The last is the definition of the partition tree itself.

DEFINITION 2.7. (Restated) Given $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$, a Partition Tree of X is a tree, $\mathcal{T}$, whose nodes are labeled by $\left(Z, \{\mathcal{T}_C\}_{C \in C_{\text{low}}}, \mathcal{T}_{\text{rep}}, C_{\text{low}}, C_{\text{high}}, C_{\text{rep}}, r_{\text{apx}}\right)$ where $Z, C_{\text{rep}} \subset X$, $\{\mathcal{T}_C\}_{C \in C_{\text{low}}} \cup \{\mathcal{T}_{\text{rep}}\}$ represent its children, $C_{\text{low}}, C_{\text{high}}$ are partitions of Z and $r_{\text{apx}} > 0$ satisfying the following conditions:

$$\text{CC}(Z, 1000n^2 r_{\text{apx}}) \sqsubseteq C_{\text{high}} \sqsubseteq \text{CC}(Z, r_{\text{apx}}) \sqsubseteq \text{CC}(Z, r_{\text{med}}) \sqsubseteq \text{CC}\left(Z, \frac{r_{\text{apx}}}{10n}\right) \sqsubseteq C_{\text{low}} \sqsubseteq \text{CC}\left(Z, \frac{r_{\text{apx}}}{1000n^3}\right)$$
$$\forall C \in C_{\text{high}} : \exists!\, z \in C_{\text{rep}} \text{ with } z \in C.$$

For the sake of notational simplicity, we will use $\mathcal{T}' \in \mathcal{T}$ both to refer to a node in the tree as well as the subtree rooted at that node and $\text{Size}(\mathcal{T}')$ to refer to the sum of the number of points stored in the subtree $\mathcal{T}'$. The above condition implies $\text{Size}(\mathcal{T}) \leqslant O(n \log n)$ [7].

We refer to the size of a subtree $\mathcal{T}' = \left(Z, \{\mathcal{T}_C\}_{C \in C_{\text{low}}}, \mathcal{T}_{\text{rep}}, C_{\text{low}}, C_{\text{high}}, C_{\text{rep}}, r_{\text{apx}}\right) \in \mathcal{T}$ by the following recursive formula:

$$\text{Size}(\mathcal{T}) = |Z| + \sum_{C \in C_{\text{low}}} \text{Size}(\mathcal{T}_C) + \text{Size}(\mathcal{T}_{\text{rep}}).$$

We state a result proved in Appendix A which allows the construction of a partition tree in time near-linear in the dataset size.

LEMMA 4.1. *Let $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$ and $\delta \in (0, 1)$. Then, Algorithm 8 when given X, δ and n, runs in time $\widetilde{O}(nd \log(1/\delta))$ and constructs, $\mathcal{T}$, satisfying:*

$$\forall \mathcal{T}' = \left(Z, \{\mathcal{T}_C\}_{C \in C_{\text{low}}}, \mathcal{T}_{\text{rep}}, C_{\text{low}}, C_{\text{high}}, C_{\text{rep}}, r_{\text{apx}}\right) \in \mathcal{T} :$$
$$\text{CC}(Z, 1000n^2 r_{\text{apx}}) \sqsubseteq C_{\text{high}} \sqsubseteq \text{CC}(Z, r_{\text{apx}}) \sqsubseteq \text{CC}(Z, r_{\text{med}}) \sqsubseteq \text{CC}\left(Z, \frac{r_{\text{apx}}}{10n}\right) \sqsubseteq C_{\text{low}} \sqsubseteq \text{CC}\left(Z, \frac{r_{\text{apx}}}{1000n^3}\right)$$

with probability at least $1 - \delta$. Furthermore, as a consequence we have for all $n \geqslant 3$:

$$\text{Size}(\mathcal{T}) \leqslant Cn \log n,\ \forall C \in C_{\text{low}} \cup \{C_{\text{rep}}\} : |C| \leqslant \frac{|Z|}{2} \text{ and } r_{\text{med}} \leqslant r_{\text{apx}} \leqslant n r_{\text{med}}.$$

Through the rest of the section, we prove Theorem 3.2. In Subsection 4.1, we overview the construction of our data structure given a partition tree constructed by say, Lemma 4.1 and in Subsection 4.2, we show how to query the data structure where we show that it is sufficient to construct terminal embeddings for the set of points in the node we terminate our traversal of the Partition Tree.

Intuitively, through a union bound and a repetition argument, one would expect to show that for $\widetilde{\Omega}(d)$ independently instantiated approximate near neighbor data structures, most of

them (say 90%) will correctly answer a near neighbor query for *any* (even perhaps, adaptively chosen based on the instantiations of the data structures) query $q \in \mathbb{R}^d$. However, care must be taken to choose the grid in the covering number argument to avoid dependence on the aspect ratio of the dataset. Additionally, we discretize the input to one of the points in the chosen grid as we assume no continuity properties on the near neighbor data structures. Furthermore, we require stronger additional properties that enable these results to be used to construct the terminal embeddings in Section 3. These technical considerations complicate the proof as it requires interleaving the choice of the grid with the design of the partition tree and its use in the nearest neighbor to near neighbor reduction. The remainder of this section illustrates and circumvents these difficulties.

4.1 Constructing the Data Structure

In this subsection, we describe how the data structure for our adaptive nearest neighbor algorithm is constructed. The procedure is outlined in Algorithm 5. The procedure takes as input a partition tree produced by *Algorithm* 8 satisfying the conclusion of Lemma 4.1, a failure probability $\delta \in (0,1)$ and the total number of points n.

Input: Partition Tree $\mathcal{T}$, Failure Probability δ, Total Number of points n

$\delta^\dagger \leftarrow \frac{c_{\text{prob}}\delta}{(nd)^{10}}, \gamma \leftarrow \frac{c_{\text{step}}\cdot\varepsilon^3}{\log^3 n}$

for $\mathcal{T}' = \left(Z, \{\mathcal{T}_C\}_{C\in\mathcal{C}_{\text{low}}}, \mathcal{T}_{\text{rep}}, C_{\text{low}}, C_{\text{high}}, C_{\text{rep}}, r_{\text{apx}}\right) \in \mathcal{T}$ **do**

  $r_{\text{low}} \leftarrow \frac{r_{\text{apx}}}{C_{\text{range}}\cdot(nd)^{10}}, r_{\text{high}} \leftarrow C_{\text{range}} \cdot (nd)^{10} r_{\text{apx}}$

  $l \leftarrow \left\lceil \frac{\log r_{\text{high}}/r_{\text{low}}}{\gamma} \right\rceil, \; s \leftarrow C_{\text{rep}}(d + \log l/\delta^\dagger)\log(nd)$

  For $i \in \{0\} \cup [l], j \in [s]$, let $\mathcal{D}_{i,j}$ be i.i.d ANN data structures with $(Z, (1+\gamma)^i r_{\text{low}})$

  Let $\mathcal{D}_{\mathcal{T}'} = \{\mathcal{D}_{i,j}\}_{i\in\{0\}\cup[l], j\in[s]}$

return $\{\mathcal{D}_{\mathcal{T}'}\}_{\mathcal{T}'\in\mathcal{T}}$

Algorithm 5. ConstructAANN$(\mathcal{T}, \delta, n)$

To state the correctness guarantees on the data structure, we will require most of the ANN data structures to be accurate for an appropriately chosen discretization of $\mathbb{R}^d$. The reason for this will become clear when defining the operation of the query procedure on the data structure produced by the algorithm. To construct the discrete set of points, consider a particular node $\mathcal{T}' \in \mathcal{T}$. Let $\mathcal{T}' = \left(Z, \{\mathcal{T}_C\}_{C\in\mathcal{C}_{\text{low}}}, \mathcal{T}_{\text{rep}}, C_{\text{low}}, C_{\text{high}}, C_{\text{rep}}, r_{\text{apx}}\right) \in \mathcal{T}$ and $\mathcal{G}(v)$ be the discrete subset of $\mathbb{R}^d$ whose coordinates are integral multiples of $v = \frac{\gamma}{1000(nd)^{20}} \cdot r_{\text{low}}$ where $r_{\text{low}} = \frac{r_{\text{apx}}}{C_{\text{range}}\cdot(nd)^{10}}$ and $\gamma = \frac{c_{\text{step}}\cdot\varepsilon^3}{\log^3 n}$ as in Algorithm 5. Again, letting $r_{\text{high}} = C_{\text{range}} \cdot (nd)^{10} r_{\text{apx}}$ as in Algorithm 5, we define

the grid of points corresponding to $\mathcal{T}'$ as follows:

$$\mathcal{H}(\mathcal{T}') = \bigcup_{x \in Z} \mathbb{B}(x, 10^6 \cdot (nd)^{20} \cdot r_{\text{high}}) \cap \mathcal{G}(v).$$

That is $\mathcal{H}$ corresponds to the set of points in $\mathcal{G}$ within $10^6 \cdot (nd)^{20} r_{\text{high}}$ of some point in Z. Finally, the set of points where we would like to ensure the correctness of our procedure is given by:

$$\mathcal{J} = \bigcup_{\mathcal{T}' \in \mathcal{T}} \mathcal{H}(\mathcal{T}'). \tag{AANN-Grid}$$

We now state the main result concerning the correctness of Algorithm 5:

LEMMA 4.2. *Let $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$, $\delta \in (0,1)$ and $\mathcal{T}$ be a Partition Tree of X satisfying the conclusion of Lemma 4.1. Then, Algorithm 5 when run with input X, δ and n, returns $\mathcal{D} = \{\mathcal{D}_{\mathcal{T}'}\}_{\mathcal{T}' \in \mathcal{T}}$ satisfying:*

$$\forall \mathcal{D}_{\mathcal{T}'} = \{\mathcal{D}_{i,j}\}_{i \in \{0\} \cup [l], j \in [s]} \in \mathcal{D},\ i \in \{0\} \cup [l],\ z \in \mathcal{J} : \sum_{j=1}^{s} \mathbf{1}\left\{\mathcal{D}_{i,j} \text{ answers } z \text{ correctly}\right\} \geqslant 0.95s$$

with probability at least $1 - \delta$. Furthermore, the space complexity of $\mathcal{D}$ is $O^*(n^{1+\rho_u}(d + \log 1/\delta))$.

PROOF. We start by bounding the amount of space occupied by the data structure. For the nearest neighbor data structures instantiated, note that the space utilization of a single data structure with n points scales as $O^*(n^{1+\rho_u+f(n)})$ where $f(n) = o(1)$. Let T be such that $f(n) \leqslant 1$ for all $n \geqslant T$ and let $M = \max_{i \in [T]} f(i)$. To account for the space occupied by the data structure, first consider nodes in the tree with less than $\log n$ points. Let $\mathcal{D} = \left(Z, \{\mathcal{T}_C\}_{C \in C_{\text{low}}}, \mathcal{T}_{\text{rep}}, C_{\text{low}}, C_{\text{high}}, C_{\text{rep}}, r_{\text{apx}}\right)$ be such a node. For this node, the space occupied by the nearest neighbor data structures is at most $O^*(|Z|^{1+\rho_u+M}(d + \log 1/\delta))$ as we instantiate $O^*(d + \log 1/\delta)$ nearest neighbor data structures in each node. Since, $|Z| \leqslant \log n$ and there are at most $O^*(n \log n)$ many of these nodes (Lemma 4.1), the total amount of space occupied by these nodes is at most $O^*(n(d + \log 1/\delta))$. Now, consider the alternate case where $|Z| \geqslant \log n$. Through similar reasoning, the total amount of space occupied by such a node is $O^*(|Z|^{1+\rho_u+f(|Z|)}(d + \log 1/\delta))$. Now, summing over all the nodes in the tree and by using the convexity of the function $f(x) = x^{1+\rho}$ for $\rho > 0$, we get that the total amount of space occupied by nodes with more than $\log n$ points is at most $O^*(n^{1+\rho_u}(d + \log 1/\delta))$. This completes the bound on the space complexity of the data structure produced by Algorithm 5.

We will now establish the correctness guarantees required of $\mathcal{D}$. To bound the size of $\mathcal{J}$, first consider a single $\mathcal{H}(\mathcal{T}')$ in the definition of $\mathcal{J}$. For a single term in the definition of $\mathcal{H}(\mathcal{T}')$, $\mathcal{V} = \mathbb{B}\left(x, 10^6 \cdot (nd)^{20} \cdot r_{\text{high}}\right) \cap \mathcal{G}(v)$, note that $\mathcal{V}$ is a v packing of $\mathbb{B}\left(x, (10^6 + 1) \cdot (nd)^{20} \cdot r_{\text{high}}\right)$. Therefore, from standard bounds on packing and covering numbers and the definitions of v and r_{high} in terms of r_{apx}, we get that $|\mathcal{V}| \leqslant (nd)^{O(d)}$ [12, Section 4.2]. By taking a union bound over the $O^*(n \log n)$ nodes in the tree and the at most n points in each node, we get that $|\mathcal{J}| \leqslant (nd)^{O(d)}$. Now, for a particular $z \in \mathcal{J}$ and a particular node $\mathcal{T}' \in \mathcal{T}$ with data structure $\mathcal{D}_{\mathcal{T}'} = \{\mathcal{D}_{i,j}\}_{i \in \{0\} \cup [l], j \in [s]}$, the probability that $\mathcal{D}_{i,j}$ incorrectly answers the Approximate Near

Neighbor query is at most 0.01. Therefore, we have by Hoeffding's inequality that the probability that more than $0.05s$ of the $\mathcal{D}_{i,j}$ answer z incorrectly is at most $\delta/(10 \cdot |\mathcal{T}| \cdot |\mathcal{J}|)$ from our setting of s and our bounds on $|\mathcal{J}|$ and $|\mathcal{T}|$. A union bound over $\mathcal{J}$ and the nodes of the tree establishes the lemma. ■

4.2 Querying the Data Structure and Proof of Theorem 3.2

The procedure to query the data structure returned by Algorithm 5 is described in Algorithm 6. The procedure takes as inputs the partition tree, $\mathcal{T}$, and the data structure output by Algorithm 5 which has a separate data structure for each $\mathcal{T}' \in \mathcal{T}$. The procedure recursively explores the nodes of $\mathcal{T}$ starting at the root and moving down the tree and stops when the approximate nearest neighbor found is within $poly(n)r_{\text{apx}}$ of the nodes in the tree. In addition, in anticipation of its application towards computing terminal embeddings, we show that it is sufficient to construct a terminal embedding for the set of points in the node the algorithm terminates in. We now state the main lemma of this subsection which shows both that the data point x returned by Algorithm 6 is an approximate nearest neighbor of q as well as establishing that it suffices to construct a terminal embedding for data points in the node of the Partition tree the algorithm terminates in.

LEMMA 4.3. *Let $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$, $\mathcal{T}$ be a valid Partition Tree of X and $\mathcal{D} = \left\{\mathcal{T}, \{\mathcal{D}_{\mathcal{T}'}\}_{\mathcal{T}' \in \mathcal{T}}\right\}$ be a data structure satisfying the conclusion of Lemma 4.2. Then, Algorithm 6, when run with inputs $\mathcal{D}$ and any $q \in \mathbb{R}^d$ returns a tuple $(\hat{x}, \mathcal{T}_{\text{res}})$ satisfying:*

$$\|q - \hat{x}\| \leqslant (1 + o^*(1))c \min_{x \in X} \|q - x\| \text{ and } \hat{x} \in \mathcal{T}_{\text{res}}.$$

Furthermore, let $\mathcal{Y} = \{y_i\}_{i=1}^n \subset \mathbb{R}^k$ satisfying for some $\varepsilon^\dagger \in \left(\frac{1}{\sqrt{d}}, 1\right)$:

$$\forall i, j \in [n] : (1 - \varepsilon^\dagger)\|x_i - x_j\| \leqslant \|y_i - y_j\| \leqslant (1 + \varepsilon^\dagger)\|x_i - x_j\|$$

and for $\mathcal{T}_{\text{res}} = \left(Z, \{\mathcal{T}_C\}_{C \in \mathcal{C}_{\text{low}}}, \mathcal{T}_{\text{rep}}, \mathcal{C}_{\text{low}}, \mathcal{C}_{\text{high}}, \mathcal{C}_{\text{rep}}, r_{\text{apx}}\right)$, let $y \in \mathbb{R}^k$ satisfy for some $\varepsilon^\ddagger \in [\varepsilon^\dagger, 1)$:

$$\forall x_i \in Z : (1 - \varepsilon^\ddagger)\|q - x_i\| \leqslant \|y - y_i\| \leqslant (1 + \varepsilon^\ddagger)\|q - x_i\|.$$

Then, we have that:

$$\forall x_i \in X : \left(1 - (1 + o^*(1))\,\varepsilon^\ddagger\right)\|q - x_i\| \leqslant \|y - y_i\| \leqslant \left(1 + (1 + o^*(1))\,\varepsilon^\ddagger\right)\|q - x_i\|$$

and if $|Z| > 1$:

$$\Omega\left(\frac{1}{(nd)^{10}} r_{\text{apx}}\right) \leqslant \|q - \hat{x}\| \leqslant O((nd)^{10} r_{\text{apx}}).$$

Additionally, Algorithm 6 runs in time $O^*(n^{\rho_c}(d + \log 1/\delta))$.

PROOF. We first set up some notation. Note that Algorithm 6 traverses the partition tree, $\mathcal{T}$, using the data structure $\mathcal{D}$ defined in Algorithm 5. Let $\mathcal{T}^{(0)}, \ldots, \mathcal{T}^{(K)}$ denote the sequence of

nodes traversed by the algorithm with $\mathcal{T}^{(i)} = \left(Z^{(i)}, \{\mathcal{T}_C\}^{(i)}_{C \in C^{(i)}_{\text{low}}}, \mathcal{T}^{(i)}_{\text{rep}}, C^{(i)}_{\text{low}}, C^{(i)}_{\text{high}}, C^{(i)}_{\text{rep}}, r^{(i)}_{\text{apx}}\right)$. Note that $K \leqslant \lceil \log n \rceil + 1$ as the number of data points drops by at least a factor of 2 each time the algorithm explores a new node. To prove the first claim regarding the correctness of $\hat{x}$, let $r^* = \min_{x \in X} \|q - x\|$. We now have the following claim:

CLAIM 4.4. *For all $i \in \{0, \ldots, K\}$, we have:*

$$\exists z \in Z^{(i)} : \|q - z\| \leqslant \left(1 + \frac{1}{(10nd)^5}\right)^i r^*.$$

Proof. We will prove the claim via induction on i. The base case where $i = 0$ is trivially true. Now, suppose that the claim holds true for all nodes up to $\mathcal{T}^{(k)}$ and we wish to establish the claim from $\mathcal{T}^{(k+1)}$. For the algorithm to reach node $\mathcal{T}^{(k+1)}$ from node $\mathcal{T}^{(k)}$, one of the following two cases must have occurred:

1. Either $\mathcal{T}^{(k+1)} = \mathcal{T}^{(k)}_{\text{rep}}$ or
2. $\mathcal{T}^{(k+1)} = \mathcal{T}_C$ for some $C \in C^{(k)}_{\text{low}}$

Now, let $\widetilde{q}^{(k)}$ be the discretization of q when $\mathcal{T}^{(k)}$ is being processed. We first handle the case where $\widetilde{q}^{(k)} \notin \mathcal{J}$, we have by the triangle inequality:

$$r^* \geqslant \min_{x \in X} \|\widetilde{q}^{(k)} - x\| - \|q - \widetilde{q}^{(k)}\| \geqslant 5 \cdot 10^5 \cdot (nd)^{20} \cdot r_{\text{high}}$$

and the first case occurs. Now assume that $\widetilde{q}^{(k)} \in \mathcal{J}$. If the first case occurs, we again have by the triangle inequality:

$$\min_{z \in Z^{(k)}} \|q - z\| \geqslant \min_{z \in Z^{(k)}} \|\widetilde{q}^{(k)} - z\| - \|q - \widetilde{q}^{(k)}\| \geqslant 0.9 r_{\text{high}}$$

by the fact that Algorithm 6 recurses on $C^{(k)}_{\text{rep}}$ and the conclusion of Lemma 4.2 which ensures that $\widetilde{q}^{(k)}$ does not have a neighbor in $Z^{(k)}$ within a distance of $r^{(k)}_{\text{high}}$. Now, let z^* be the closest neighbor to q in $Z^{(k)}$; that is, $z^* = \arg\min_{z \in Z^{(k)}} \|q - z\|$. We know that there exists $C \in C_{\text{high}}$ such that $z^* \in C$. Furthermore, we have by the triangle inequality and the fact that $\text{CC}(Z^{(k)}, 1000n^2 r^{(k)}_{\text{apx}}) \sqsubseteq C^{(k)}_{\text{high}}$ (Lemma 4.1) that $\|z - z^*\| \leqslant 1000n^3 r^{(k)}_{\text{apx}}$ for all $z \in C$. Note that C has representative $\hat{z}$ in the construction of $C^{(k)}_{\text{rep}}$. For $\hat{z}$, we have

$$\frac{\|q - \hat{z}\|}{\|q - z^*\|} \leqslant 1 + \frac{\|\hat{z} - z^*\|}{\|q - z^*\|} \leqslant 1 + \frac{1000n^3 r_{\text{apx}}}{0.9 r_{\text{high}}} \leqslant \left(1 + \frac{1}{(10nd)^5}\right).$$

Along with the induction hypothesis, the above fact concludes the proof of the claim in this case. Finally, for the second case, again let $z^* = \arg\min_{z \in Z^{(k)}} \|q - z\|$ and $C \in C^{(k)}_{\text{low}}$ such that $z^* \in C$. From the definition of $C^{(k)}_{\text{low}}$ and Lemma 4.1, we also know for all $z \in Z^{(k)} \setminus C$, $\|z^* - z\| \geqslant \frac{r_{\text{apx}}}{(1000n^3)}$ as $C_{\text{low}} \sqsubseteq \text{CC}\left(Z^{(k)}, \frac{r_{\text{apx}}}{1000n^3}\right)$. We have by the triangle inequality for all $z \in Z^{(k)} \setminus C$:

$$\|\widetilde{q}^{(k)} - z\| \geqslant \|z - z^*\| - \|q - z^*\| - \|q - \widetilde{q}^{(k)}\| = \|z - z^*\| - \min_{z \in Z^{(k)}} \|q - z\| - \|q - \widetilde{q}^{(k)}\|$$

$$\geqslant \|z - z^*\| - \min_{z \in Z^{(k)}} (\|\widetilde{q}^{(k)} - z\| + \|\widetilde{q}^{(k)} - q\|) - \|q - \widetilde{q}^{(k)}\|$$
$$\geqslant \frac{r_{\text{apx}}^{(k)}}{1000n^3} - cr_{\text{low}}^{(k)} - 2\sqrt{d}v^{(k)} \geqslant c(10nd)^7 r_{\text{low}}^{(k)}.$$

Therefore, all $x \in Z^{(k)}$ such that $\|\widetilde{q} - x\| \leqslant cr_{\text{low}}^{(k)}$ must belong to C. Consequently, we recurse on $\mathcal{T}_C$ with $z^* \in C \in \mathcal{C}_{\text{low}}$ which establishes the inductive hypothesis in this case as well. ◆

To finish the proof of the first claim, let $\mathcal{T}^{(K)} = \left(Z, \{\mathcal{T}_C\}_{C \in \mathcal{C}_{\text{low}}}, \mathcal{T}_{\text{rep}}, \mathcal{C}_{\text{low}}, \mathcal{C}_{\text{high}}, \mathcal{C}_{\text{rep}}, r_{\text{apx}}\right)$ with associated data structure $\mathcal{D}_{\mathcal{T}^{(K)}} = \{\mathcal{D}_{i,j}\}_{i \in \{0\} \cup [l], j \in [s]}$. If $|Z| = 1$, a direct application of Claim 4.4 establishes the lemma. Alternatively, we must have $0 < i^* \leqslant l$ when $\mathcal{T}^{(K)}$ is being processed by Algorithm 6. From the guarantees on $\mathcal{D}_{i,j}$ (Lemma 4.2), we get $\min_{z \in Z} \|z - \widetilde{q}\| > (1+\gamma)^{i^*-1} r_{\text{low}}$. Letting $x \in Z$ such that $\|x - \widetilde{q}\| \leqslant c(1+\gamma)^{i^*} r_{\text{low}}$ returned by Algorithm 6, we get by another application of the triangle inequality:

$$\frac{\|q - x\|}{\min_{z \in Z} \|q - z\|} \leqslant \frac{\|\widetilde{q} - x\| + \|\widetilde{q} - q\|}{\min_{z \in Z} \|\widetilde{q} - z\| - \|\widetilde{q} - q\|} \leqslant \frac{c(1+\gamma)^{i^*} r_{\text{low}} + \sqrt{d}v}{(1+\gamma)^{i^*-1} r_{\text{low}} - \sqrt{d}v}$$
$$\leqslant c(1+\gamma) + \frac{2c\sqrt{d}v}{(1+\gamma)^{i^*-1} r_{\text{low}} - \sqrt{d}v} \leqslant c(1+\gamma)^2$$

where the first inequality is valid as $\min_{z \in Z} \|\widetilde{q} - z\| - \|\widetilde{q} - q\| > 0$ from our setting of v and the condition on $\min_{z \in Z} \|q - z\|$ and the final inequality similarly follows from our setting of v. Another application of Claim 4.4 with the fact that $K \leqslant \lceil \log n \rceil + 1$, establishes the first claim of the lemma.

To prove the second claim of the lemma, we prove an analogous claim to Claim 4.4 for terminal embeddings:

CLAIM 4.5. *We have for all $i \in \{0, \ldots, K\}$:*

$$\forall z_j \in Z^i : \left(1 - \left(1 + \frac{1}{(10nd)^5}\right)^{K-i} \varepsilon^{\ddagger}\right) \|q - x_j\| \leqslant \|y - y_j\| \leqslant \left(1 + \left(1 + \frac{1}{(10nd)^5}\right)^{K-i} \varepsilon^{\ddagger}\right) \|q - x_j\|.$$

Proof. We will prove the claim by reverse induction on i. For $i = K$, the claim is implied by the assumptions on y. Now, suppose the claim holds for $i = k+1$ and we wish to establish the claim for $i = k$. As in the proof of Claim 4.4, we have two cases when $\mathcal{T}^{(k)}$ is being processed:

1. Either $\mathcal{T}^{(k+1)} = \mathcal{T}_{\text{rep}}^{(k)}$ or
2. $\mathcal{T}^{(k+1)} = \mathcal{T}_C^{(k)}$ for some $C \in \mathcal{C}_{\text{low}}^{(k)}$.

As in Claim 4.4, when the first case occurs, we have $\min_{z \in Z^{(k)}} \|q - z\| \geqslant 0.9 r_{\text{high}}^{(k)}$. Now, for any $z_i \in Z^{(k)}$, let $z_j \in C_{\text{rep}}^{(k)}$ such that $z_i, z_j \in C \in \mathcal{C}_{\text{high}}^{(k)}$. Note that we must have by the triangle inequality and the fact that $\text{CC}(Z^{(k)}, 1000n^2 r_{\text{apx}}^{(k)}) \sqsubseteq \mathcal{C}_{\text{high}}$ that $\|z_i - z_j\| \leqslant 1000n^3 r_{\text{apx}}^{(k)}$. From the fact that $C_{\text{rep}}^{(k)} = Z^{(k+1)}$, the inductive hypothesis and the assumption on y_i, y_j from the lemma,

we get:

$$\begin{aligned}
\|y - y_i\| &\geqslant \|y - y_j\| - \|y_j - y_i\| \geqslant \left(1 - \left(1 + \frac{1}{(10nd)^5}\right)^{K-(k+1)} \varepsilon^{\ddagger}\right) \|q - z_j\| - 2000n^3 r_{\text{apx}}^{(k)} \\
&\geqslant \left(1 - \left(1 + \frac{1}{(10nd)^5}\right)^{K-(k+1)} \varepsilon^{\ddagger}\right) (\|q - z_i\| - \|z_i - z_j\|) - 2000n^3 r_{\text{apx}}^{(k)} \\
&\geqslant \left(1 - \left(1 + \frac{1}{(10nd)^5}\right)^{K-(k+1)} \varepsilon^{\ddagger}\right) \|q - z_i\| - 4000n^3 r_{\text{apx}}^{(k)} \\
&\geqslant \left(1 - \left(1 + \frac{1}{(10nd)^5}\right)^{K-(k+1)} \varepsilon^{\ddagger}\right) \|q - z_i\| - \frac{1}{(10nd)^5}\varepsilon^{\ddagger}\|q - z_i\| \\
&\geqslant \left(1 - \left(1 + \frac{1}{(10nd)^5}\right)^{K-k} \varepsilon^{\ddagger}\right) \|q - z_i\|
\end{aligned}$$

where the last inequality follows from the fact that $(1+a)^b \geqslant (1+a)^{b-1} + a$ for $a \geqslant 0, b \geqslant 1$. For the other direction, we have by a similar calculation:

$$\begin{aligned}
\|y - y_i\| &\leqslant \|y - y_j\| + \|y_j - y_i\| \leqslant \left(1 + \left(1 + \frac{1}{(10nd)^5}\right)^{K-(k+1)} \varepsilon^{\ddagger}\right) \|q - z_j\| + 2000n^3 r_{\text{apx}}^{(k)} \\
&\leqslant \left(1 + \left(1 + \frac{1}{(10nd)^5}\right)^{K-(k+1)} \varepsilon^{\ddagger}\right) (\|q - z_i\| + \|z_i - z_j\|) + 2000n^3 r_{\text{apx}}^{(k)} \\
&\leqslant \left(1 + \left(1 + \frac{1}{(10nd)^5}\right)^{K-(k+1)} \varepsilon^{\ddagger}\right) \|q - z_i\| + 4000n^3 r_{\text{apx}}^{(k)} \\
&\leqslant \left(1 + \left(1 + \frac{1}{(10nd)^5}\right)^{K-(k+1)} \varepsilon^{\ddagger}\right) \|q - z_i\| + \frac{1}{(10nd)^5}\varepsilon^{\ddagger}\|q - z_i\| \\
&\leqslant \left(1 + \left(1 + \frac{1}{(10nd)^5}\right)^{K-k} \varepsilon^{\ddagger}\right) \|q - z_i\|.
\end{aligned}$$

This establishes the claim in the first case. For the second case, let $z^* = \arg\min_{z \in Z^{(k)}} \|q - z\|$. As in the proof of Claim 4.5, we have $z^* \in Z^{(k+1)} = C \in C_{\text{low}}^{(k)}$ and for all $z \in Z^{(k)} \setminus C$:

$$\begin{aligned}
\|q - z\| &\geqslant \|z^* - z\| - \|q - z^*\| = \|z^* - z\| - \min_{z \in Z^{(k)}} \|q - z\| \\
&\geqslant \|z^* - z\| - \min_{z \in Z^{(k)}} \|\widetilde{q}^{(k)} - z\| - \|\widetilde{q}^{(k)} - q\| \\
&\geqslant \frac{r_{\text{apx}}^{(k)}}{1000n^3} - c r_{\text{low}}^{(k)} - \sqrt{d}\nu^{(k)} \geqslant c(10nd)^7 r_{\text{low}}^{(k)}
\end{aligned}$$

and furthermore, we have:

$$\|q - z^*\| = \min_{z \in Z^{(k)}} \|q - z^*\| \leqslant \min_{z \in Z^{(k)}} \|\widetilde{q}^{(k)} - z^*\| + \|\widetilde{q}^{(k)} - q\| \leqslant c r_{\text{low}}^{(k)} + r_{\text{low}}^{(k)} \leqslant 2c r_{\text{low}}^{(k)}.$$

Now, the claim is already established for all $x_i \in Z^{(k+1)}$. For $z_i \in Z^{(k)} \setminus Z^{(k+1)}$, letting $y^* = y_j$ for $x_j = z^*$:

$$\begin{aligned}\|y_i - y\| &\leqslant \|y_i - y_j\| + \|y - y_j\| \leqslant (1+\varepsilon^\dagger)\|x_i - x_j\| + \left(1 + \left(1 + \frac{1}{(10nd)^5}\right)^{K-(k+1)} \varepsilon^\ddagger\right)\|q - z^*\| \\ &\leqslant (1+\varepsilon^\dagger)\|x_i - x_j\| + 3cr_{\text{low}}^{(k)} \leqslant (1+\varepsilon^\dagger)\|x_i - q\| + (1+\varepsilon^\dagger)\|q - x_j\| + 3cr_{\text{low}}^{(k)} \\ &\leqslant \left(1 + \left(1 + \frac{6cr_{\text{low}}^{(k)}}{\varepsilon^\dagger \cdot \|x_i - q\|}\right)\varepsilon^\dagger\right)\|x_i - q\| \leqslant \left(1 + \left(1 + \frac{1}{(10nd)^5}\right)\varepsilon^\dagger\right)\|x_i - q\|\end{aligned}$$

where the third inequality is due to the fact that $K \leqslant \lceil \log n \rceil + 1$ and the fact that $(1+a)^b \leqslant e^{ab}$ for $a, b \geqslant 0$ and the final two inequalities follow from the upper bound on $\|q - z^*\|$ and the lower bound on $\|x_i - q\|$ established previously. For the lower bound, we have by a similar computation:

$$\begin{aligned}\|y_i - y\| &\geqslant \|y_i - y_j\| - \|y - y_j\| \geqslant (1-\varepsilon^\dagger)\|x_i - x_j\| - \left(1 + \left(1 + \frac{1}{(10nd)^5}\right)^{K-(k+1)} \varepsilon^\ddagger\right)\|q - z^*\| \\ &\geqslant (1-\varepsilon^\dagger)\|x_i - x_j\| - 3cr_{\text{low}}^{(k)} \geqslant (1-\varepsilon^\dagger)\|x_i - q\| - (1-\varepsilon^\dagger)\|q - x_j\| - 3cr_{\text{low}}^{(k)} \\ &\geqslant \left(1 - \left(1 + \frac{6cr_{\text{low}}^{(k)}}{\varepsilon^\dagger \cdot \|x_i - q\|}\right)\varepsilon^\dagger\right)\|x_i - q\| \geqslant \left(1 - \left(1 + \frac{1}{(10nd)^5}\right)\varepsilon^\dagger\right)\|x_i - q\|.\end{aligned}$$

The assumption that $\varepsilon^\ddagger \geqslant \varepsilon^\dagger$ finishes the proof of the claim by induction. ◆

Claim 4.5 establishes the second conclusion of lemma from the fact that $K \leqslant \lceil \log n \rceil + 1$. When $|Z| > 1$ in $\mathcal{T}_{\text{res}}$, we get by the triangle inequality, the fact that $\|\widetilde{q} - \hat{x}\| \geqslant r_{\text{low}}$ and the definition of $\widetilde{q}$ in Algorithm 6 that:

$$\Omega\left(\frac{1}{(nd)^{10}} r_{\text{apx}}\right) \leqslant \|q - \hat{x}\| \leqslant O((nd)^{10} r_{\text{apx}}).$$

Finally, for the bound on the runtime, note that Algorithm 6 queries at most $Kl(s+1)$ near neighbor data structures built with at most n points. Therefore, each query takes time at most $O^*(n^{\rho_c})$. This proves our bound on the runtime of Algorithm 6. ■

Proof of Theorem 3.2: Lemmas 4.1 to 4.3 now establish Theorem 3.2 barring the decoupling of d and the n^{ρ_c} term in the runtime guarantee. To do this, note that we may simply use the Median-JL data structure (see Theorem 3.14 in Section 5) by instantiating $l = \widetilde{O}(d)$ many JL-sketches and for each of them, instantiate an adaptive approximate nearest neighbor data structure in the low dimensional space. At query time, we simply pick $\Omega(\log n)$ of these sketches uniformly at random, query each of the corresponding nearest neighbor data structures with the projection of q and return the best answer. This yields the improved runtimes from Theorem 3.2. ■

Input: Data Structure $\mathcal{D} = \left\{\mathcal{T}, \{\mathcal{D}_{\mathcal{T}'}\}_{\mathcal{T}' \in \mathcal{T}}\right\}$, Query q
$\mathcal{T}_{\text{cur}} \leftarrow \mathcal{T}$
while TRUE **do**
  Let $\mathcal{T}_{\text{cur}} = \left(Z, \{\mathcal{T}_C\}_{C \in \mathcal{C}_{\text{low}}}, \mathcal{T}_{\text{rep}}, \mathcal{C}_{\text{low}}, \mathcal{C}_{\text{high}}, C_{\text{rep}}, r_{\text{apx}}\right) \in \mathcal{T}$
  if $|Z| = 1$ **then**
    return $(x, \mathcal{T}_{\text{cur}})$ for $x \in Z$
  $\nu \leftarrow \frac{\gamma}{1000(nd)^{20}} \cdot r_{\text{low}}$
  $\widetilde{q}_i \leftarrow \left\lfloor \frac{q_i}{\nu} \right\rfloor \nu$
  $i^* \leftarrow$
  $\min \left\{ i : \exists j \text{ s.t } \mathcal{D}_{i,j}(\widetilde{q}) \text{ returns } x \in Z \text{ satisfying } \|\widetilde{q} - x\| \leqslant c(1+\gamma)^i r_{\text{low}} \right\}$
  if $0 < i^* \leqslant l$ **then**
    Let $x \in Z$ be such that $\|x - \widetilde{q}\| \leqslant c(1+\gamma)^{i^*} r_{\text{low}}$
    Return: $(x, \mathcal{T}_{\text{cur}})$
  else if $i^* = 0$ **then**
    Let $x \in Z$ be such that $\|x - \widetilde{q}\| \leqslant c r_{\text{low}}$
    Let $C \in \mathcal{C}_{\text{low}}$ be such that $x \in C$
    $\mathcal{T}_{\text{cur}} \leftarrow \mathcal{T}_C$
  else if $i^* = \infty$ **then**
    $\mathcal{T}_{\text{cur}} \leftarrow \mathcal{T}_{\text{rep}}$

Algorithm 6. QueryAANN$(\mathcal{D}, q)$

5. Median - JL

In this section, we will prove the following lemma which enabled speeding up our algorithms by effectively projecting onto a low dimensional subspace essentially decoupling the terms that depend on d and n.

THEOREM 3.14. **(Restated)** *Let $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$, $\varepsilon \in \left(\frac{1}{\sqrt{nd}}, 1\right)$ and $\delta \in (0,1)$. Furthermore, let $\{\Pi^i\}_{i=1}^m \subset \mathbb{R}^{k \times d}$ for $m \geqslant \Omega((d + \log 1/\delta) \log nd)$ and $k \geqslant \Omega\left(\frac{\log n}{\varepsilon^2}\right)$ with $\Pi^i_{j,l} \overset{iid}{\sim} \mathcal{N}(0, 1/k)$. Then, we have:*

$$\forall q \in \mathbb{R}^d : \sum_{i=1}^m \mathbf{1}\left\{\Pi^i \text{ satisfies AP-IP}(\varepsilon, X \cup \{q\})\right\} \geqslant 0.95m \qquad \text{(JL-Rep)}$$

with probability at least $1 - \delta$.

By setting $x = y$ in AP-IP, we see that the above theorem is a generalization of the standard Johnson-Lindenstrauss condition where in addition to maintaining distances between points, Π

is also required to approximately maintain relative inner products between the points in the augmented dataset. To begin the proof, we start by recalling the standard Johnson-Lindenstrauss lemma (see, for example, [12]).

LEMMA 5.1. *Let $\Pi \in \mathbb{R}^{k\times d}$ be distributed according to $\Pi_{i,j} \overset{iid}{\sim} \mathcal{N}(0,1/k)$, $\delta \in (0,1)$ and $k \geqslant C\frac{\log 1/\delta}{\varepsilon^2}$ for some absolute constant $C > 0$. Then, for any $v \in \mathbb{R}^d$, we have:*

$$\mathbb{P}\left\{(1-\varepsilon)\|v\|^2 \leqslant \|\Pi v\|^2 \leqslant (1+\varepsilon)\|v\|^2\right\} \geqslant 1-\delta.$$

We obtain via a union bound over all pairs of points in the dataset, X, the JL guarantee:

COROLLARY 5.2. *Let $X = \{x_i\}_{i=1}^n$, $\Pi \in \mathbb{R}^{k\times d}$ with $\Pi_{i,j} \overset{iid}{\sim} \mathcal{N}(0,1/k)$ with $k \geqslant C\frac{(\log n+\log 1/\delta)}{\varepsilon^2}$. Then, we have:*

$$\forall x_i, x_j \in X : (1-\varepsilon)\|x_i - x_j\| \leqslant \|\Pi x_i - \Pi x_j\| \leqslant (1+\varepsilon)\|x_i - x_j\|$$

with probability at least $1-\delta$.

A second corollary we will make frequent use of is the following where we show that Π also approximately preserves inner products.

COROLLARY 5.3. *Let $\Pi \in \mathbb{R}^{k\times d}$ be distributed according to $\Pi_{i,j} \overset{iid}{\sim} \mathcal{N}(0,1/k)$ and $k \geqslant C\frac{\log 1/\delta}{\varepsilon^2}$. Then, for any $x, y \in \mathbb{R}^d$, we have:*

$$\mathbb{P}\left\{|\langle \Pi x, \Pi y\rangle - \langle x, y\rangle| \leqslant \varepsilon\|x\|\|y\|\right\} \geqslant 1-\delta.$$

PROOF. If either x or y are 0, the conclusion follows trivially. Assume, $x, y \neq 0$. By scaling both sides by $\|x\|\|y\|$, we may assume that $\|x\| = \|y\| = 1$. We now have the following:

$$\langle \Pi x, \Pi y\rangle = \frac{1}{4}\left(\|\Pi(x+y)\|^2 - \|\Pi(x-y)\|^2\right)$$
$$\langle x, y\rangle = \frac{1}{4}\left(\|(x+y)\|^2 - \|(x-y)\|^2\right).$$

By subtracting both equations, we get:

$$|\langle \Pi x, \Pi y\rangle - \langle x, y\rangle| \leqslant \frac{1}{4}\left(\left|\|\Pi(x+y)\|^2 - \|x+y\|^2\right| + \left|\|\Pi(x-y)\|^2 - \|x-y\|^2\right|\right)$$

By the union bound, the triangle inequality and Lemma 5.1, we get with probability at least $1-\delta$:

$$\left|\|\Pi(x+y)\|^2 - \|x+y\|^2\right| \leqslant \frac{\varepsilon}{4}\|x+y\|^2 \leqslant \varepsilon \text{ and}$$
$$\left|\|\Pi(x-y)\|^2 - \|x-y\|^2\right| \leqslant \frac{\varepsilon}{4}\|x-y\|^2 \leqslant \varepsilon.$$

The inequalities in the previous display imply the lemma. ■

We start by establishing a simple lemma on the norms of the matrices Π^i.

LEMMA 5.4. *Assume the setting of Theorem 3.14. Then, we have:*

$$\sum_{i=1}^{m} \mathbf{1}\left\{\|\Pi^i\|_F \leqslant O(\sqrt{d})\right\} \geqslant 0.99m$$

with probability at least $1 - \delta/4$.

PROOF. Let $W_i = \mathbf{1}\left\{\|\Pi^i\|_F \leqslant O(\sqrt{d})\right\}$. We have by an application of Bernstein's inequality that $\mathbb{P}(W_i = 1) \geqslant 0.995$ as $\sum_{j\in[k],l\in[d]}(\Pi^i_{j,l})^2$ is a χ^2 random variable with mean d. Since, the W_i are iid, the conclusion follows by an application of Hoeffding's inequality applied to the random variable $W = \sum_{i=1}^m W_i$. ■

LEMMA 5.5. *Assume the setting of Theorem 3.14. Then, we have:*

$$\forall \|v\| = 1 : \sum_{i=1}^{m} \mathbf{1}\left\{\begin{array}{c}(1-\varepsilon/128) \leqslant \|\Pi^i v\|^2 \leqslant (1+\varepsilon/128)\\ \forall x, y \in X : |\langle \Pi^i v, \Pi^i(x-y)\rangle - \langle v, x-y\rangle| \leqslant \dfrac{\varepsilon}{128}\cdot\|x-y\|\\ \|\Pi^i\|_F \leqslant O(\sqrt{d})\end{array}\right\} \geqslant 0.98m$$

with probability at least $1 - \delta/2$.

PROOF. Let $\mathcal{G}$ be a γ-net over $\mathbb{S}^{d-1}$ with $\gamma = \frac{c}{(nd)^{10}}$ for some small enough constant c. Furthermore, we may assume $|\mathcal{G}| \leqslant (nd)^{O(d)}$. Now, for $u \in \mathcal{G}$, let:

$$W_i(u) = \mathbf{1}\left\{\begin{array}{c}(1-\varepsilon/256) \leqslant \|\Pi^i u\|^2 \leqslant (1+\varepsilon/256)\\ \forall x, y \in X : |\langle \Pi^i u, \Pi^i(x-y)\rangle - \langle u, x-y\rangle| \leqslant \dfrac{\varepsilon}{256}\cdot\|x-y\|\end{array}\right\}.$$

We have from Corollary 5.3, that $\mathbb{P}(W_i(u) = 1) \geqslant 0.995$. Therefore, we have by an application of Hoeffding's inequality and a union bound over $\mathcal{G}$ that:

$$\mathbb{P}\left\{\forall u \in \mathcal{G} : \sum_{i=1}^{m} W_i(u) \geqslant 0.99m\right\} \geqslant 1 - \delta/4.$$

We now condition on the event from the previous equation and the conclusion of Lemma 5.4. To extend from the net $\mathcal{G}$ to the whole sphere, consider $v \in \mathbb{S}^{d-1}$ and its nearest neighbor $u \in \mathcal{G}$. Note that $\|v - u\| \leqslant \gamma$. Let $i \in [m]$ be such that $W_i(u) = 1$ and $\|\Pi^i\|_F \leqslant O(\sqrt{d})$. We have:

$$|\|\Pi^i v\|^2 - \|\Pi^i u\|^2| = |(v+u)^\top (\Pi^i)^\top \Pi^i (v-u)| \leqslant 2\cdot\|\Pi^i\|_F^2 \cdot \gamma \leqslant \varepsilon/256.$$

Furthermore, we have for all $x, y \in X$:

$$\begin{aligned}
&|\langle \Pi^i v, \Pi^i(x-y)\rangle - \langle v, x-y\rangle| \\
&\quad\leqslant |\langle \Pi^i u, \Pi^i(x-y)\rangle - \langle u, x-y\rangle| + |\langle \Pi^i(v-u), \Pi^i(x-y)\rangle| + |\langle v-u, x-y\rangle| \\
&\quad\leqslant \frac{\varepsilon}{256}\cdot\|x-y\| + \gamma\cdot\|\Pi^i\|_F^2\cdot\|x-y\| + \gamma\cdot\|x-y\| \leqslant \frac{\varepsilon}{128}\cdot\|x-y\|.
\end{aligned}$$

Since, for any $u \in \mathcal{G}$, at least $0.98m$ of the Π^i satisfy $W_i(u) = 1$ and $\|\Pi^i\|_F \leqslant O(\sqrt{d})$ with probability at least $1 - \delta/2$, the conclusion of the lemma follows. ■

Proof of Theorem 3.14: Finally, to establish Theorem 3.14, we will need to use a more intricate multi-scale gridding argument than the one used to prove Lemma 5.5. Using a single grid of resolution γ does not suffice as the dataset, X, may contain pairs of points separated by much less than γ. Bounding the error of the embedding of q in terms of its nearest neighbor in the net does not suffice in such situations. On the other hand, using a finer net whose resolution is comparable to the minimum distance between the points in the dataset leads to a choice of m dependent on the aspect ratio of X. The multi-scale argument presented here allows us to circumvent these difficulties.

To define the grid, let $r_{ij} = \|x_i - x_j\|$ for $x_i, x_j \in X$ and $\mathcal{G}_{ij}$ be a γ-net of $\mathbb{B}(x_i, 2(Cnd)^{10} r_{ij})$ with $\gamma = (Cnd)^{-10} \cdot r_{ij}$ for some large enough constant C. The grid in our argument will consist of the union of all the $\mathcal{G}_{ij}$; that is $\mathcal{G} = \bigcup_{i,j\in[n]} \mathcal{G}_{ij}$. Now, for $u \in \mathcal{G}$, define $W_i(u)$ as follows:

$$W_i(u) = \mathbf{1}\left\{\Pi^i \text{ satisfies } AP\text{-}IP(\varepsilon/256, X \cup \{u\})\right\}.$$

From Corollary 5.3, we have $\mathbb{P}(W_i(u) = 1) \geqslant 0.995$. Noting that $|\mathcal{G}| \leqslant (2nd)^{O(d)}$, we have by Hoeffding's Inequality and the union bound that with probability at least $1 - \delta/4$, we have for all $u \in \mathcal{G}$: $\sum_{i=1}^{m} W_i(u) \geqslant 0.99m$. For the rest of the argument, we will also condition on the conclusions of Lemma 5.5. Note, that this event occurs with probability at least $1 - \delta$ from the union bound. Therefore, we have by the union bound with probability at least $1 - \delta$:

$$\forall u \in \mathcal{G}, \forall \|v\| = 1 : \sum_{i=1}^{m} \mathbf{1}\left\{\begin{array}{c} (1-\varepsilon/128) \leqslant \|\Pi^i v\|^2 \leqslant (1+\varepsilon/128) \\ \forall x, y \in X : |\langle \Pi^i v, \Pi^i(x-y)\rangle - \langle v, x-y\rangle| \leqslant \dfrac{\varepsilon}{128} \cdot \|x-y\| \\ \Pi^i \text{ satisfies } AP\text{-}IP(\varepsilon/256, X \cup \{u\}) \\ \|\Pi^i\|_F \leqslant O(\sqrt{d}) \end{array}\right\} \geqslant 0.95m.$$

Letting $Y_i(u, v)$ denote the indicator in the above expression, we now condition on the above event for the rest of the proof. Let $q \in \mathbb{R}^d$ and $x_q = \arg\min_{x\in X} \|q - x\|$ (its closest neighbor in X). Note that the case where $q = x_q$ is already covered by the condition on the W_i. Therefore, we assume $q \neq x_q$. With $v_q = \frac{(q-x_q)}{\|q-x_q\|}$ and $\widetilde{q} = \arg\min_{u\in\mathcal{G}} \|q - u\|$, let $\mathcal{J}(q) = \{i : Y_i(\widetilde{q}, v_q) = 1\}$. We will now prove for all $i \in \mathcal{J}(q)$:

$$\forall \widetilde{x}, \widetilde{y}, \widetilde{z} \in X \cup \{q\} : |\langle \Pi^i(\widetilde{x} - \widetilde{z}), \Pi^i(\widetilde{y} - \widetilde{z})\rangle - \langle \widetilde{x} - \widetilde{z}, \widetilde{y} - \widetilde{z}\rangle| \leqslant \varepsilon \|\widetilde{x} - \widetilde{z}\| \|\widetilde{y} - \widetilde{z}\|.$$

When $\widetilde{x} = \widetilde{z}$ or $\widetilde{y} = \widetilde{z}$, the conclusion is trivial. Furthermore, for $\widetilde{x}, \widetilde{y}, \widetilde{z} \in X$, the conclusion follows from the definition of $\mathcal{J}$. Hence, we may restrict ourselves to cases where $\widetilde{x}, \widetilde{y}$ are distinct from $\widetilde{z}$ and at least one of $\widetilde{x}, \widetilde{y}, \widetilde{z}$ are q. We first tackle the cases where $\widetilde{x}, \widetilde{y}, \widetilde{z}$ are distinct and we have the following subcases:

Case 1: $\widetilde{x} = q$ and $\widetilde{y}, \widetilde{z} \in X$. In this case, we have from the definition of $\mathcal{J}$:

$$\begin{aligned} &|\langle \Pi^i(q - \widetilde{z}), \Pi^i(\widetilde{y} - \widetilde{z})\rangle - \langle q - \widetilde{z}, \widetilde{y} - \widetilde{z}\rangle| \\ &\quad \leqslant |\langle \Pi^i(q - x_q), \Pi^i(\widetilde{y} - \widetilde{z})\rangle - \langle q - x_q, \widetilde{y} - \widetilde{z}\rangle| + |\langle \Pi^i(x_q - \widetilde{z}), \Pi^i(\widetilde{y} - \widetilde{z})\rangle - \langle x_q - \widetilde{z}, \widetilde{y} - \widetilde{z}\rangle| \end{aligned}$$

$$\leqslant \frac{\varepsilon}{128} \cdot \|q - x_q\| \cdot \|\widetilde{y} - \widetilde{z}\| + \frac{\varepsilon}{256} \cdot \|x_q - \widetilde{z}\| \cdot \|\widetilde{y} - \widetilde{z}\|$$
$$\leqslant \frac{\varepsilon}{128} \cdot \|q - \widetilde{z}\| \cdot \|\widetilde{y} - \widetilde{z}\| + \frac{\varepsilon}{256} \cdot 2\|q - \widetilde{z}\| \cdot \|\widetilde{y} - \widetilde{z}\| \leqslant \frac{\varepsilon}{64} \cdot \|q - \widetilde{z}\| \cdot \|\widetilde{y} - \widetilde{z}\|$$

concluding the proof in this case.

Case 2: $\widetilde{z} = q$ and $\widetilde{x}, \widetilde{y} \in X$. We have by algebraic expansion:

$$\begin{aligned}
|\langle \Pi^i(\widetilde{x} - q), \Pi^i(\widetilde{y} - q)\rangle - \langle \widetilde{x} - q, \widetilde{y} - q\rangle| &= |(\widetilde{x} - q)^\top((\Pi^i)^\top\Pi^i - I)(\widetilde{y} - q)| \\
&\leqslant |(\widetilde{x} - x_q)^\top((\Pi^i)^\top\Pi^i - I)(\widetilde{y} - x_q)| + |(x_q - q)^\top((\Pi^i)^\top\Pi^i - I)(\widetilde{y} - x_q)| + \\
&\quad |(\widetilde{x} - x_q)^\top((\Pi^i)^\top\Pi^i - I)(x_q - q)| + |(x_q - q)^\top((\Pi^i)^\top\Pi^i - I)(x_q - q)| \\
&\leqslant \frac{\varepsilon}{256} \cdot \|x_q - \widetilde{x}\|\|x_q - \widetilde{y}\| + \frac{\varepsilon}{128} \cdot \|x_q - q\|\|\widetilde{y} - x_q\| \\
&\quad + \frac{\varepsilon}{128} \cdot \|\widetilde{x} - x_q\|\|q - x_q\| + \frac{\varepsilon}{128} \cdot \|x_q - q\|^2 \\
&\leqslant \frac{\varepsilon}{256} \cdot 2\|\widetilde{x} - q\| \cdot 2\|\widetilde{y} - q\| + \frac{\varepsilon}{128} \cdot \|\widetilde{x} - q\| \cdot 2\|\widetilde{y} - q\| \\
&\quad + \frac{\varepsilon}{128} \cdot 2\|\widetilde{x} - q\| \cdot \|\widetilde{y} - q\| + \frac{\varepsilon}{128} \cdot \|\widetilde{x} - q\|\|\widetilde{y} - q\| \\
&\leqslant \frac{\varepsilon}{16} \cdot \|\widetilde{x} - q\| \cdot \|\widetilde{y} - q\|
\end{aligned}$$

establishing the statement in this case as well.

We now move on to the case where $\widetilde{x}, \widetilde{y}, \widetilde{z}$ are not distinct. As remarked before, it suffices to consider $x = \widetilde{x} = \widetilde{y} \neq \widetilde{z}$ and one of $x, \widetilde{z}$ are q. Without loss of generality, we may assume $\widetilde{z} = q$. When $x = x_q$ the conclusion follows from the definition of $\mathcal{J}(q)$. As a consequence, our goal simplifies to establishing for all $i \in \mathcal{J}(q)$ and $x \neq x_q$:

$$|\|\Pi^i(q - x)\|^2 - \|q - x\|^2| \leqslant \varepsilon\|q - x\|^2.$$

Let $r = \|x - x_q\|$ and here again, we have two cases:

Case 1: $\|q - x_q\| \leqslant 2(Cnd)^{10}r$. From the construction of $\mathcal{G}$, we have $\|\widetilde{q} - q\| \leqslant (Cnd)^{-10}r$. Furthermore, we have from the fact that $r \leqslant 2\|q - x\|$:

$$\begin{aligned}
|\|q - x\|^2 - \|\widetilde{q} - x\|^2| &= |\langle q - \widetilde{q}, (q - x) + (\widetilde{q} - x)\rangle| \\
&\leqslant (Cnd)^{-10} \cdot r \cdot (\|q - x\| + \|\widetilde{q} - x\|) \leqslant \frac{\varepsilon}{1024} \cdot \|q - x\|^2.
\end{aligned}$$

Additionally, we have:

$$\begin{aligned}
&|(\|\Pi^i(x - q)\|^2 - \|x - q\|^2) - (\|\Pi^i(x - \widetilde{q})\|^2 - \|x - \widetilde{q}\|^2)| \\
&\quad = |(x - q)^\top((\Pi^i)^\top\Pi^i - I)(x - q) - (x - \widetilde{q})^\top((\Pi^i)^\top\Pi^i - I)(x - \widetilde{q})| \\
&\quad = |(\widetilde{q} - q)^\top((\Pi^i)^\top\Pi^i - I)((x - q) + (x - \widetilde{q}))| \\
&\quad \leqslant \|\widetilde{q} - q\|(\|\Pi^i\|_F^2 + 1)(\|x - q\| + \|x - \widetilde{q}\|) \leqslant \frac{\varepsilon}{1024} \cdot \|x - q\|^2.
\end{aligned}$$

The conclusion now follows from the inequalities in the previous two displays and the following condition on Π^i in the definition of $\mathcal{J}$:

$$|\|\Pi^i(\widetilde{q}-x)\|^2 - \|\widetilde{q}-x\|^2| \leqslant \frac{\varepsilon}{256}\cdot\|\widetilde{q}-x\|^2 \leqslant \frac{\varepsilon}{128}\|q-x\|^2.$$

Case 2: $\|q-x_q\| \geqslant 2(Cnd)^{10}r$. We have similarly to the previous case:

$$\begin{aligned}|\|q-x\|^2 - \|q-x_q\|^2| &= |\langle x_q - x, (q-x)+(q-x_q)\rangle| \\ &\leqslant r\cdot(\|q-x\|+\|q-x_q\|) \leqslant \frac{\varepsilon}{1024}\cdot\|q-x\|^2.\end{aligned}$$

Now, we have:

$$\begin{aligned}&|(\|\Pi^i(q-x)\|^2 - \|q-x\|^2) - (\|\Pi^i(q-x_q)\|^2 - \|q-x_q\|^2)| \\ &\quad= |(q-x)^\top((\Pi^i)^\top\Pi^i - I)(q-x) - (q-x_q)^\top((\Pi^i)^\top\Pi^i - I)(q-x_q)| \\ &\quad= |(x_q-x)^\top((\Pi^i)^\top\Pi^i - I)((q-x)+(q-x_q))| \\ &\quad\leqslant \|x_q-x\|(\|\Pi^i\|_F^2+1)(\|q-x\|+\|q-x_q\|) \leqslant \frac{\varepsilon}{1024}\cdot\|q-x\|^2.\end{aligned}$$

Similarly, we conclude from the following inequality implied by the definition of $\mathcal{J}$:

$$|\|\Pi^i(q-x_q)\|^2 - \|q-x_q\|^2| \leqslant \frac{\varepsilon}{128}\cdot\|q-x_q\|^2 \leqslant \frac{\varepsilon}{128}\cdot\|q-x\|^2.$$

The four cases just enumerated establish the theorem assuming $|\mathcal{J}(q)| \geqslant 0.95m$ for all $q \in \mathbb{R}^d$. As shown before, this occurs with probability at least $1-\delta$ concluding the proof of the theorem. ■

6. Acknowledgements

The authors would like to thank Sidhanth Mohanty for enlightening conversations in the course of this project and helpful feedback in the preparation of this manuscript.

References

[1] **Alexandr Andoni and Piotr Indyk**. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, 2008. DOI (8)

[2] **Alexandr Andoni**, **Thijs Laarhoven**, **Ilya P. Razenshteyn**, **and Erik Waingarten**. Optimal hashing-based time-space trade-offs for approximate near neighbors. *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 47–66. SIAM, 2017. DOI (7–9, 14)

[3] **Dimitri P. Bertsekas**. Nonlinear programming. Athena Scientific Optimization and Computation Series. Athena Scientific, Belmont, MA, third edition, 2016., pages xviii+861 (5, 50, 51)

[4] **Yeshwanth Cherapanamjeri and Jelani Nelson**. On adaptive distance estimation. *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL (5, 12, 13, 30)

[5] **Michael Elkin**, **Arnold Filtser**, **and Ofer Neiman**. Terminal embeddings. *Theor. Comput. Sci.* 697:1–36, 2017. DOI (2–4)

[6] **Sariel Har-Peled**. A replacement for voronoi diagrams of near linear size. *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 94–103. IEEE Computer Society, 2001. DOI (11)

[7] **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. DOI (8, 10, 11, 22, 32, 47, 49)

[8] **William B. Johnson and Joram Lindenstrauss**. Extensions of Lipschitz mappings into a Hilbert space, *Conference in modern analysis and probability (New Haven, Conn., 1982)*. Volume 26, Contemp. Math. Pages 189–206. Amer. Math. Soc., Providence, RI, 1984. DOI (2)

[9] **Sepideh Mahabadi**, **Konstantin Makarychev**, **Yury Makarychev**, **and Ilya P. Razenshteyn**. Nonlinear dimension reduction via outer bi-lipschitz extensions. *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 1088–1101. ACM, 2018. DOI (2, 4, 15, 16)

[10] **Shyam Narayanan and Jelani Nelson**. Optimal terminal dimensionality reduction in euclidean space. *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 1064–1069. ACM, 2019. DOI (2, 4, 5, 15, 16)

[11] **A. S. Nemirovsky and D. B. and Yudin**. Problem complexity and method efficiency in optimization. A Wiley-Interscience Publication. John Wiley & Sons, Inc., New York, 1983., pages xv+388. Translated from the Russian and with a preface by E. R. Dawson, Wiley-Interscience Series in Discrete Mathematics (5, 50, 51)

[12] **Roman Vershynin**. High-dimensional probability, volume 47 of *Cambridge Series in Statistical and Probabilistic Mathematics*. Cambridge University Press, Cambridge, 2018., pages xiv+284. An introduction with applications in data science, With a foreword by Sara van de Geer. DOI (34, 41)

A. Construct Partition Tree

In this section, we discuss our construction of a Partition Tree (Definition 2.7) and prove Lemma 4.1 which we restate below:

LEMMA 4.1. (Restated) *Let $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$ and $\delta \in (0,1)$. Then, Algorithm 8 when given X, δ and n, runs in time $\widetilde{O}(nd\log(1/\delta))$ and constructs, $\mathcal{T}$, satisfying:*

$$\forall \mathcal{T}' = \left(Z, \{\mathcal{T}_C\}_{C\in C_{\text{low}}}, \mathcal{T}_{\text{rep}}, C_{\text{low}}, C_{\text{high}}, C_{\text{rep}}, r_{\text{apx}}\right) \in \mathcal{T} :$$
$$\text{CC}(Z, 1000n^2 r_{\text{apx}}) \sqsubseteq C_{\text{high}} \sqsubseteq \text{CC}(Z, r_{\text{apx}}) \sqsubseteq \text{CC}(Z, r_{\text{med}}) \sqsubseteq \text{CC}\left(Z, \frac{r_{\text{apx}}}{10n}\right) \sqsubseteq C_{\text{low}} \sqsubseteq \text{CC}\left(Z, \frac{r_{\text{apx}}}{1000n^3}\right)$$

with probability at least $1-\delta$. Furthermore, as a consequence we have for all $n \geqslant 3$:

$$\text{Size}(\mathcal{T}) \leqslant Cn\log n,\ \forall C \in C_{\text{low}} \cup \{C_{\text{rep}}\} : |C| \leqslant \frac{|Z|}{2} \text{ and } r_{\text{med}} \leqslant r_{\text{apx}} \leqslant n r_{\text{med}}.$$

The algorithm works by recursively constructing the nodes of the tree starting from the root and then partitioning the point set for that node to construct its children and so on. In Appendix A.1 we show how to partition points at a single node and in Appendix A.2, we use this to construct the full tree.

A.1 Partitioning at a Single Node

In this subsection, we describe the partitioning procedure at a single node. This will then be incorporated into a recursive procedure to construct the entire tree. To begin, recall Definitions 2.5 and 2.6 and the definition of r_{med} from Section 2. While it is possible to compute $r_{\text{med}}(X)$ in time $\widetilde{O}(n^2 d)$, obtaining a crude estimate is sufficient for our purposes [7]. Furthermore, we will also not require computing $\text{CC}(X, r)$ exactly but appropriate refinements and coarsenings suffice. We first restate a simple lemma from [7]:

LEMMA A.1. *Given $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$ and $\delta \in (0,1)$, there is a randomized algorithm,* CompRmed, *that computes in time $O(nd \log 1/\delta)$ and outputs an estimate r_{apx} satisfying:*

$$\mathbb{P}\left\{r_{\text{apx}} \geqslant r_{\text{med}}(X)\right\} = 1 \text{ and } \mathbb{P}\left\{r_{\text{apx}} \leqslant n r_{\text{med}}(X)\right\} \geqslant 1 - \delta.$$

PROOF. Let $C \in \text{CC}(X, r_{\text{med}}(X))$ be such that $|C| \geqslant n/2$. Picking a point, x, uniformly at random from X picks a point in C with probability at least $1/2$. Now, we compute distances $\{\|x_i - x\|\}_{i=1}^n$ and output their median, r_{apx}. Conditioned on $x \in C$, we have by the triangle inequality, that $r_{\text{apx}} \leqslant n r_{\text{med}}(X)$ which proves the second claim with probability at least $1/2$. For the first, note that x belongs to a connected component in $\text{CC}(X, r_{\text{apx}})$ of size at least $n/2$. This establishes the first claim of the lemma. By repeating this procedure $\Omega(\log 1/\delta)$ times and taking the minimum of the returned estimates establishes the lemma by an application of Hoeffding's inequality. ■

LEMMA A.2. *Let $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$, $r > 0$ and $\delta \in (0,1)$. Then, there is a randomized algorithm,* ConstructPartition *that outputs a partitioning of X, C, satisfying:*

$$\text{CC}(X, 1000n^2 r) \sqsubseteq C \sqsubseteq \text{CC}(X, r)$$

with probability at least $1 - \delta$. Furthermore, the algorithm runs in time $O(nd \log(n/\delta))$.

PROOF. The randomized algorithm is detailed in the following pseudocode.

By the definition of Algorithm 7, we see that for every $(x, y) \in E$, we must have $\|x - y\| \leqslant v$. Therefore, we obtain C refines $\text{CC}(X, 1000n^2 r)$. We now show that $\text{CC}(X, r)$ refines C. To do this, we will need the following claim:

CLAIM A.3. *For $x, y \in \mathbb{R}^d$ and $g \sim \mathcal{N}(0, I)$, we have:*

$$\mathbb{P}\left\{|\langle g, x - y\rangle| \leqslant \tau\right\} \geqslant \frac{999}{1000} \text{ if } \|x - y\| \leqslant r$$
$$\mathbb{P}\left\{|\langle g, x - y\rangle| \leqslant \tau\right\} \leqslant \frac{1}{100n^2} \text{ if } \|x - y\| \geqslant v.$$

Input: Point set $X = \{x_i\}_{i=1}^n \subset \mathbb{R}^d$, Resolution r,
Failure Probability δ

$K \leftarrow 10\log(n/\delta), \tau \leftarrow 10r, v \leftarrow 1000n^2 r$
$V \leftarrow X, E \leftarrow \phi$
for $k = 1 : K$ **do**
  $g_k \sim \mathcal{N}(0, I)$
  For all $i \in [n]$, let $v_i^{(k)} = \langle x_i, g_k \rangle$
  Let $x_1^{(k)}, \ldots, x_n^{(k)}$ be an ordering of the x_i increasing in $v_i^{(k)}$
  $i \leftarrow 1$
  while $i < n$ **do**
    $i_n = \max\{i < j \leqslant n : v_j^{(k)} \leqslant v_i^{(k)} + \tau\}$
    Add $(x_i^{(k)}, x_j^{(k)})$ to E for $j \in \{i+1, \ldots, i_n\}$ if $\|x^{(k)} - x^{(k)}\| \leqslant v$
    $i \leftarrow \max(i+1, i_n)$
$C \leftarrow \text{ConnectedComponents}(V, E)$
return C

Algorithm 7. ConstructPartition(X, r, δ)

Proof. The first inequality follows from the observation that $\langle g, x - y \rangle$ is a Gaussian with variance $\|x - y\|^2$ and the definition of τ. The second follows from the same fact and the fact that pdf of the standard normal distribution takes maximum value $1/\sqrt{2\pi}$. ◆

Now, fix an outer iteration k. From Claim A.3, we have with probability at least $1/100$, that $|v_i^{(k)} - v_j^{(k)}| > \tau$ for all $\|x_i - x_j\| \geqslant v$. Let $x_i, x_j \in X$ with $\|x_i - x_j\| \leqslant r$. Then with probability at least 0.999, we have that $|v_i^{(k)} - v_j^{(k)}| \leqslant \tau$. For the rest of the argument, we condition on the previous two events and we assume, without loss of generality, that $v_i^{(k)} \leqslant v_j^{(k)}$. Note that since $v_j^{(k)} - v_i^{(k)} \leqslant \tau$, one of the following cases must occur:

Case 1: $(x_i, x_j) \in E$,

Case 2: $(z, x_i), (z, x_j) \in E$ for some $z \in X$ or

Case 3: $(z, x_i), (z, w), (w, x_j) \in E$ for some $z, w \in X$.

In all three cases, we see that x_i and x_j are in the same connected component with respect to E. Note that x_i, x_j are in the same connected component if our good event occurs for at least 1 of the K outer iterations of the algorithm. Therefore, the probability that x_i, x_j are in the same connected component over all K runs is at least $1 - \delta/n^2$. By a union bound, this establishes

that with probability at least $1-\delta$, x_i, x_j are in the same connected component in E for all $\|x_i - x_j\| \leqslant r$.

The runtime of the algorithm follows from the fact that we add at most $O(n\log(n/\delta))$ edges to E over the entire run of the algorithm and we do at most $O(nd)$ amount of work in each run of the while loop. ■

A.2 Constructing the Partition Tree - Proof of Lemma 4.1

We will now use the results of Appendix A.1 to construct the whole tree, proving Lemma 4.1.

We will first assume that the functions CompRmed and ConstructPartition run successfully (that is, they satisfy the conclusions of Lemmas A.1 and A.2 respectively) in every recursive call of Algorithm 8 and then finally bound the probability of this event. Note that when CompRmed runs successfully, r_{apx} always satisfies $r_{\text{med}} \leqslant r_{\text{apx}} \leqslant nr_{\text{med}}$ for every recursive call of Algorithm 8 (Lemma A.1). Together with the correctness of ConstructPartition (Lemma A.2), we get that:

$$\text{CC}(Z, 1000n^2 r_{\text{apx}}) \sqsubseteq C_{\text{high}} \sqsubseteq \text{CC}(Z, r_{\text{apx}}) \sqsubseteq \text{CC}(Z, r_{\text{med}}) \sqsubseteq \text{CC}\left(Z, \frac{r_{\text{apx}}}{10n}\right) \sqsubseteq C_{\text{low}} \sqsubseteq \text{CC}\left(Z, \frac{r_{\text{apx}}}{1000n^3}\right)$$

for every node $\mathcal{T}' = \left(Z, \{\mathcal{T}_C\}_{C\in C_{\text{low}}}, \mathcal{T}_{\text{rep}}, C_{\text{low}}, C_{\text{high}}, C_{\text{rep}}, r_{\text{apx}}\right) \in \mathcal{T}$. Furthermore, for any such $\mathcal{T}'$, we get from the fact that $\text{CC}(Z, r_{\text{med}}) \sqsubseteq C_{\text{low}}$, that all $C \in C_{\text{low}}$ satisfy $|C| \leqslant |Z|/2$. In addition, the definition of r_{med} and the fact that $C_{\text{high}} \sqsubseteq \text{CC}(Z, r_{\text{med}})$ yield $|C_{\text{rep}}| \leqslant |Z|/2$. To bound, first note that $|C_{\text{rep}}| \leqslant |C_{\text{low}}|$ from the fact that $C_{\text{high}} \sqsubseteq C_{\text{low}}$ and the construction of C_{rep}. To bound Size$(\mathcal{T})$, define $B(n)$ as follows:

$$B(n) = n + \max_{\substack{n_1,\ldots,n_k,k \\ \sum_{i=1}^k n_i = n \\ k\leqslant n/2, \forall i\in[k]: 1\leqslant n_i\leqslant n/2}} B(k) + \sum_{i=1}^{k} B(n_i) \text{ and } B(1) = 1.$$

From the definition of $B(n)$, we see that $B(n)$ is monotonic in n and from this, we get that $B(n)$ is an upper bound on Size$(\mathcal{T})$. We now recall the following claim from [7]:

CLAIM A.4 ([7]). *For all $n \geqslant 3$, $B(n) \leqslant Cn\log n$.*

The above claim establishes the bound on Size$(\mathcal{T})$.

Finally, we bound the probability that any execution of CompRmed and ConstructPartition fail. We start by bounding the probability that any of the first $5B(n)$ runs of CompRmed and ConstructPartition fail. From the definition of $\delta^\dagger$, the probability that any of the $5B(n)$ runs of CompRmed and ConstructPartition fail is at most $1-\delta$ by the union bound. However, the preceding argument shows that the algorithm terminates with fewer than $B(n)$ recursive calls if none of the executions of CompRmed and ConstructPartition fail. Therefore, the probability that any of the executions of CompRmed and ConstructPartition fail in the running of the algorithm is at most $1-\delta$. This yields the previously derived conclusions with probability at least $1-\delta$. ■

Input: Point set $Z = \{x_i\}_{i=1}^m \subset \mathbb{R}^d$, Failure Probability δ, Total Number of points n

1: **if** $|X| = 1$ **then**
2: **return** $(X, \phi, \phi, \phi, \phi, \phi)$
3: $\delta^\dagger \leftarrow \frac{c_{\text{prob}}\delta}{n^2}$
4: $r_{\text{apx}} \leftarrow \text{CompRmed}(Z, \delta^\dagger)$
5: $C_{\text{low}} \leftarrow \text{ConstructPartition}(Z, r_{\text{apx}}/(1000n^3), \delta^\dagger)$, $C_{\text{high}} \leftarrow \text{ConstructPartition}(Z, r_{\text{apx}}, \delta^\dagger)$
6: For $C \in C_{\text{low}}$, let $\mathcal{T}_C \leftarrow \text{ConstructPartitionTree}(C, n, \delta)$
7: For $C \in C_{\text{high}}$, pick representative $x \in C$ and add it to C_{rep}
8: $\mathcal{T}_{\text{rep}} \leftarrow \text{ConstructPartitionTree}(C_{\text{rep}}, n, \delta)$
9: **return** $\left(Z, \{\mathcal{T}_C\}_{C \in C_{\text{low}}}, \mathcal{T}_{\text{rep}}, C_{\text{low}}, C_{\text{high}}, C_{\text{rep}}, r_{\text{apx}}\right)$

Algorithm 8. ConstructPartitionTree(X, n, δ)

B. Miscellaneous Results

In this section, we develop some standard tools needed for our constructions. In Appendix B.1, we recall some basic facts about the Ellipsoid algorithm for convex optimization [11, 3] and analyze it when it's instantiated with a weak oracle as in our terminal embedding construction.

B.1 Ellipsoid Algorithm Preliminaries

We recall some basic facts regarding the operation of the Ellipsoid algorithm for convex optimization. We will instead use a weaker version where your goal is simply to output a feasible point in a convex set. In what follows, our goal is to find a point in a closed convex set, $K \subset \mathbb{R}^d$ and we are given a starting x and $R \geqslant 0$ such that for all $K \subseteq \mathbb{B}(x, R)$. Furthermore, the algorithm assumes access to an oracle O which when given a point $x \in \mathbb{R}^d$ either:

1. Outputs $v \neq 0$ such that $\forall y \in K, \langle y - x, v\rangle \geqslant 0$ or
2. Outputs FAIL.

Given access to such an oracle the Ellipsoid algorithm proceeds as follows:

Input: Initialization $x \in \mathbb{R}^d$, Initial Distance $R > 0$, Separating Oracle $\mathcal{O}$

$x^{(0)} \leftarrow x, A^{(0)} \leftarrow R^2 \cdot I, t \leftarrow 0$

while $v^{(t)} = \mathcal{O}(x^{(t)}) \neq \text{FAIL}$ **do**

$u^{(t+1)} \leftarrow \frac{A^{(t)} v^{(t)}}{\sqrt{(v^{(t)})^\top A^{(t)} v^{(t)}}}$

$x^{(t+1)} \leftarrow x^{(t)} + \frac{1}{(d+1)} u^{(t+1)}$

$A^{(t+1)} \leftarrow \frac{d^2}{d^2-1}\left(A^{(t)} - \frac{2}{d+1} u^{(t)} (u^{(t)})^\top\right)$

$t \leftarrow t + 1$

return $x^{(t)}$

Algorithm 9. Ellipsoid$(x, R, \mathcal{O})$

We recall some classical facts regarding the operation of Algorithm 9 where $\mathcal{E}(x, A)$ denotes the ellipsoid $\{y \in \mathbb{R}^d : (y-x)^\top A^{-1}(y-x) \leqslant 1\}$.

LEMMA B.1 ([11, 3]). *Let $x \in \mathbb{R}^d$, $A \succ 0$. Then for any $v \neq 0$, let:*

$$u = \frac{Av}{\sqrt{v^\top A v}}, \qquad \widetilde{x} = x + \frac{1}{d+1} u \quad \text{and} \quad \widetilde{A} = \frac{d^2}{d^2-1}\left(A - \frac{2}{d+1} u u^\top\right).$$

Then, we have:

$$\mathcal{E}(x, A) \cap \{y : \langle y - x, v\rangle \geqslant 0\} \subseteq \mathcal{E}(\widetilde{x}, \widetilde{A}) \quad \text{and} \quad \text{Vol}(\mathcal{E}(\widetilde{x}, \widetilde{A})) \leqslant \exp\left(-\frac{1}{2(d+1)}\right) \text{Vol}(\mathcal{E}(x, A)).$$

We now use Lemma B.1 to establish a slightly weaker guarantee for the Algorithm 9 corresponding to our weaker oracle.

LEMMA 3.8. (Restated) *Suppose $\varepsilon > 0$ and $K \subset \mathbb{R}^d$ be a closed convex set such that there exists $x^* \in K$ such that for all $y \in \mathbb{B}(x^*, \varepsilon)$, $y \in K$. Furthermore, let $x \in \mathbb{R}^d$ and $R > 0$ be such that $K \subset \mathbb{B}(x, R)$. Suppose further that $\mathcal{O}$ satisfies for any input z, $\mathcal{O}$:*

1. *Outputs $v \neq 0$ such that for all $y \in K$, we have $\langle v, y - x\rangle \geqslant 0$ or*
2. *Outputs* FAIL.

Then, Algorithm 9 when instantiated with x, R and $\mathcal{O}$, outputs $\hat{x}$ satisfying $\mathcal{O}(\hat{x}) =$ FAIL. Furthermore, the number of iterations of the algorithm is bounded by $O\left(d^2 \log \frac{R}{\varepsilon}\right)$ and hence the total computational complexity is bounded by $O\left(d^4 \log \frac{R}{\varepsilon}\right)$.

PROOF. Suppose Algorithm 9 ran for T iterates, $(x^{(t)}, A^{(t)})_{t=0}^T$. From Lemma B.1, we have for any $t \in \{0, \ldots, T-1\}$, $\text{Vol}(\mathcal{E}(x^{(t+1)}, A^{(t+1)})) \leqslant \exp\left(-\frac{1}{2(d+1)}\right) \text{Vol}(\mathcal{E}(x^{(t)}, A^{(t)}))$. Furthermore, Lemma B.1 along with our assumption on $\mathcal{O}$ also implies that for all $t \in \{0, \ldots, T\}$,

$K \subseteq \mathcal{E}(x^{(t)}, A^{(t)})$. From these facts, we have:

$$\exp\left(-\frac{T}{2(d+1)}\right)\mathrm{Vol}(\mathbb{B}(x,R)) = \exp\left(-\frac{T}{2(d+1)}\right)\mathrm{Vol}(\mathcal{E}(x,R^2\cdot I)) \geqslant \mathrm{Vol}(K) \geqslant \mathrm{Vol}(\mathbb{B}(x^*,\varepsilon)).$$

By taking rearranging the above inequality, taking logarithms on both sides and by using the homogeneity properties of Euclidean volume, we get the desired bound on the number of iterations. The bound on the computational complexity of the algorithm follows from the fact that each iteration takes time $O(d^2)$ to compute. ■

B.2 Miscellaneous Technical Results

Here, we present miscellaneous technical results needed in other parts of our proof.

LEMMA 3.4. (Restated) *Let $X = \{x_i\}_{i=1}^n$, $0 < \varepsilon < 1$ and $T = \left\{\frac{x-y}{\|x-y\|} : x \neq y \in X\right\} \bigcup \{0\}$. Furthermore, suppose $\Pi \in \mathbb{R}^{k\times d}$ has ε-convex hull distortion for X. Then, we have:*

$$\forall x, y \in \mathrm{Conv}(T) : |\langle \Pi x, \Pi y\rangle - \langle x, y\rangle| \leqslant 6\varepsilon.$$

PROOF. We have by the definition of the inner product:

$$\langle x, y\rangle = \frac{1}{4}\left(\|x+y\|^2 - \|x-y\|^2\right).$$

Now, let $x, y \in \mathrm{Conv}(T)$. If $x = y$, the result is true from the fact that Π has ε-convex hull distortion for X. Therefore, assume $x \neq y$. We have:

$$\begin{aligned}|\langle \Pi x, \Pi y\rangle - \langle x, y\rangle| &= \frac{1}{4}\left(|\|\Pi(x+y)\|^2 - \|\Pi(x-y)^2\| - \|x+y\|^2 + \|x-y\|^2|\right)\\ &\leqslant \frac{1}{4}\left(|\|\Pi(x+y)\|^2 - \|x+y\|^2| + |\|\Pi(x-y)^2\| - \|x-y\|^2|\right).\end{aligned}$$

For the first term, we have:

$$|\|\Pi(x+y)\|^2 - \|x+y\|^2| = 4\left(\left\|\Pi\left(\frac{x+y}{2}\right)\right\| + \left\|\frac{x+y}{2}\right\|\right)\left|\left\|\Pi\left(\frac{x+y}{2}\right)\right\| - \left\|\frac{x+y}{2}\right\|\right| \leqslant 12\varepsilon$$

where the final inequality follows from the fact that $\frac{x+y}{2} \in \mathrm{Conv}(T)$, the fact that $\|x\|, \|y\| \leqslant 1$ and the assumption that Π has ε-convex hull distortion for X. A similar inequality for the second term yields:

$$|\langle \Pi x, \Pi y\rangle - \langle x, y\rangle| \leqslant \frac{1}{4}(12\varepsilon + 12\varepsilon) = 6\varepsilon$$

concluding the proof of the lemma. ■