IN SEARCH OF BETTER PROXIMITY

BY

NIRMAN KUMAR

DISSERTATION

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Urbana-Champaign, 2014

Urbana, Illinois

Doctoral Committee:

Associate Professor Sariel Har-Peled, Chair
Professor Jeff Erickson
Associate Professor Mahesh Viswanathan
Professor David M. Mount, University of Maryland

Abstract

Given a set of points in a metric space, a fundamental problem is to preprocess these points for answering nearest-neighbor queries on them. Proximity search is the problem of answering more general queries that need the first, second, or further closest neighbors of a query point, possibly in spaces where a separation function is defined which may be more general than a metric. In this thesis, we look at several proximity search problems. Our goal is to better understand, when proximity search is easy, i.e., there is a data-structure requiring near-linear space and allowing logarithmic query time. We study three problems:

(i) Answering nearest-neighbor queries in a metric space when the query is restricted to a subspace of low doubling dimension. We show that even though the points lie in a high dimensional ambient space, the problem is inherently low dimensional.

(ii) Answering kth nearest-neighbor queries in Euclidean space. We provide a sub-linear space data-structure for this problem. We also extend this to the case when the data points are replaced by disjoint balls (of arbitrary radii), and the distance of a query point to a ball is the distance to the ball as a set.

(iii) We consider more general distance functions and proximity search queries on them. This translates to the abstract problem of computing the lower envelope of a set of functions, for a query point. For this abstract problem, we provide a set of sufficient conditions that allow efficient data-structures for computation of the lower envelope. We apply this to several problems of interest. Among new results, we provide approximate weighted Voronoi diagrams in low dimensional Euclidean space.

To my family.

Acknowledgements

First and foremost, I would like to thank my advisor, Sariel Har-Peled. He showed confidence in me right from the start. When I was first admitted to grad school in UIUC, Sariel had graciously accepted me as a student. After I had finished my masters, and had spent some time in industry, Sariel again agreed to take me as a student. His confidence in me has been a great motivation for me. He has taught me how to do research, write and present my research better, and to cultivate taste in research problems. Sariel funded my research right from the start of my PhD to the very end, and allowed me to travel to conferences, even though I was not presenting – this has been very helpful for my research career. His jokes have been enlightening and levitating, especially when research seemed hard and dragging down. I thank him whole-heartedly for taking me as his student and mentoring me.

I would like to thank all the faculty at UIUC, from whom I learned much of Computer Science and math, simply by listening to them. I would like to thank Profs. Sarita Adve, Gul Agha, Jozsef Balogh, Bruce Berndt, Roy Campbell, Chandra Chekuri, Jeff Erickson, Zoltan Furedi, Sariel Har-Peled, Ely Kerman, Alexandra Kolla, Jose Meseguer, Manoj Prabhakaran, Hal Schenck, Paul Schupp, Mahesh Viswanathan and Doug West.

I would like to thank Chandra Chekuri and Jeff Erickson for the excellent advice they offered me along the way, and the excellent teaching they offered. I hope it has helped me improve as a teacher.

I would like to thank all my teachers, for instilling the love of academic pursuits, by their teaching.

I thank all my office mates, past and present, for being great friends and co-workers and I will miss the time spent with them. So, thanks to Naman Agarwal, Shashank Agrawal, Hsien-Chih Chang, Alina Ene, Kyle Fox, Shalmoli Gupta, Sungjin Im, Nitish Korula, Vivek Madan, Hemanta Maji, Ben Moseley, Amir Nayyeri, Kent Quanrud, Ben Raichel, Daniel Schreiber and Chao Xu.

I would like to thank my parents and my elder brother for all the love and support along the way, and for being my first teachers. I can still remember the time when math seemed arcane and hard (it still does, but slightly lesser and for different reasons), and they taught me patiently till I understood a few things, saw the beauty and simplicity of numbers and geometry, and found a new love in Mathematics.

I thank all my friends for being there for me whenever I needed them.

I thank my wife, Neha, for being a perfect companion, and making Urbana-Champaign a fun place to live in. All these years, I never felt Urbana-Champaign was even slightly boring, though having lived here for a while, sometimes it got very boring when she was not around. I thank her for her love, patience and support.

Thanks to my daughter Nysa, for being a constant source of pleasure in life. Her sense of wonder at the simplest things, and of late her many questions, reaffirm my belief that the world is a very interesting place, even though I am conditioned to think that I understand it very well. But simply thinking why she finds something very interesting, has sometimes given me inspiration to look at things differently, as well as to think how I can present things that I think are obvious to me, in a new and interesting manner.

Finally, I would like to thank the great Masters, who inspire us all by their exemplary work, to dedicate serious hours of study and thought to research, which at the face of it seems pointless, as can be experienced when we try to explain what we do to others, but when we do it, we know the tremendous satisfaction it gives us.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Proximity Search and Generalizations

Proximity search is a generalization of nearest-neighbor search. The problem is fundamental in several areas of Computer Science. Knuth, see [Knu98], attributes an early solution for geometric proximity search to McNutt, which he named as the ***post office problem***. In another early appearance, it was called ***best-match file searching*** [BK73]. In information-retrieval literature, it is known as the problem of building an index for similarity search [HS03]. In the information theory literature it appears as the problem of building a vector quantization encoder [LBG80, GN98]. In statistics and learning theory it has been termed as fast nearest-neighbor classifier [DGL96].

In the nearest-neighbor problem, we are given a set P of n points in a metric space, typically $\mathbb{R}^d$ equipped with the usual ℓ_2 metric, and we need to preprocess them, so that given a query point q, the closest point among the data points can be found efficiently. There is a straightforward algorithm to solve this problem in linear time per query, that iterates through each of the data points and returns the closest such point to q. However, some preprocessing can improve the query time.

In the plane, i.e., $d = 2$, for instance, we can construct the Voronoi diagram of the points, that can be stored using $O(n)$ space along with a point location data-structure that can locate the cell in the Voronoi diagram containing a query point q. The point location data-structure also takes $O(n)$ storage space, and allows a $O(\log n)$ query time to locate a query point. Therefore, if each cell in the Voronoi diagram is annotated with the nearest neighbor among the data points, we have a nearest neighbor data-structure that takes $O(n)$ space and affords $O(\log n)$ query time. This result is surprising, and is still one of the classical cornerstones of Computational Geometry and books on Computational Geometry written even today present this result in sufficient detail, which it deserves. The same framework extends to higher dimensions, but the space usage of the data-structure is very high – specifically, the space required to store the Voronoi diagram is $\Theta(n^{\lceil d/2 \rceil})$ in the worst case. Clarkson [Cla88] showed a data-structure with query time $O(\log n)$ and space usage $O(n^{\lceil d/2 \rceil + \delta})$, where $\delta > 0$ is a pre-specified constant, and the $O(\cdot)$ notation here hides constants c^d

for small constants like $c = 2$, that are exponential in the dimension d. Matoušek and Agarwal showed that one can trade off the space used and the resulting query time [AM93a]. Mesier [Mei93] provided a data-structure with query time $O(d^5 \log n)$, which achieved polynomial dependence on the dimension d, where the space used is $O(n^{d+\delta})$, which is the square of the usage by the data-structure of Clarkson. Even though these results are theoretically impressive and interesting, they are not too good for practical purposes if the dimension is larger than 2. Even data sets of moderate size will require too much space, as the space usage is quadratic or worse. Fortunately, in real world applications of this problem, it is often sufficient to work with an approximate answer to this problem. We now define an approximate version of the nearest neighbor problem in $\mathbb{R}^d$.

In the approximate version of the problem, we are required to return a data point such that its distance from $\mathtt{q}$ is at most $(1 + \varepsilon)$ times the distance of $\mathtt{q}$ to the closest data point. The parameter ε is the accuracy requirement of the problem and the space requirement of the data-structure depends on ε. Considerable amount of research work has been done on this problem too, see [Cla06]. This problem was first considered by Bern [Ber93], who showed that the fixed approximation factor of $O(d^{1/2})$ can be achieved in a $O(n)$ sized quad-tree based data-structure with $O(\log n)$ query time. Arya and Mount [AM93b] showed that the approximation factor can be brought down to $(1 + \varepsilon)$ if $\varepsilon > 0$ is specified at the time of preprocessing. They used a randomized data-structure of near-linear size that achieves poly-logarithmic query time in expectation. Finally, the seminal work of Arya *et al.* [AMN$^+$98] showed that the approximation parameter ε can even be specified as part of the query. Their data-structure, known as a BBDtree, thus has space requirement $O(n)$ which is independent of ε. The query time is $O((1/\varepsilon^d) \log n)$. Mount and Arya implemented this algorithm and made its optimized code available as a library [AM98]. Many researchers have proposed improvements to this data-structure, mainly to move the dependence of the query time on ε from a multiplicative factor to an additive one, see [Bes96, Dun99, Cha02, Cha06]. The best one of these improves the query time to $O(\log n + 1/\varepsilon^{d-1})$. Achieving even better dependence on ε is also a major researched problem. Clarkson [Cla94] and Chan [Cha98] showed that query time of $O((1/\varepsilon^{(d-1)/2}) \log n)$ can be achieved by increasing the size of the data-structure to roughly $O((1/\varepsilon^{(d-1)/2})n)$. Using a conceptually different approach, Har-Peled [Har01] achieved the best query time till date of $O(\log(n/\varepsilon))$ using a data-structure of size roughly $O((1/\varepsilon^d)n \log^2 n)$. Har-Peled defined an analogue of classical Voronoi diagrams, known as approximate Voronoi diagrams (**AVD**), which is a partition of space into regions, of near-linear total complexity, and an associated data point in each region, which is a valid $(1+\varepsilon)$-ANN for any point in the region. The space bound was improved by Sabharwal et al. [SSS06] by a logarithmic factor. Arya and Malamatos [AM02] simplified these results and reduced the space complexity further to $O((1/\varepsilon^d)n)$. Finally, the space complexity was

reduced to $O((1/\varepsilon^{d-1})n)$ by Arya, Malamatos, and Mount [AMM02]. They also showed that $(1+\varepsilon)$-ANN queries can be answered in $O(\log n + (1/\varepsilon^{(d-1)/2}))$ time, see also [AMM09]. All of these data-structures rely on computing an AVD of the point set and then building a point location data-structure for it.

ANN for high dimensional spaces. This thesis deals with problems in spaces of fixed, constant dimension d. Since the space of our data-structures and query times depend exponentially on the dimension d, if the dimension d is larger than $\log n$, all the above mentioned algorithms are even worse than the "brute-force" algorithm, which computes the distance from the query to every point in P, and answers queries in $O(dn)$ time. Arguably, the algorithms and data-structures, as well as the ones proposed and studied in this thesis, work only for low dimensions d – a rough ballpark figure is 20, for which the Arya and Mount library code works with excellent performance. For the sake of completeness however, we also mention some known results on the extremely important problem of ANN search in high dimensions d.

Kleinberg [Kle97] showed that by combining one dimensional random projections in a novel way, a $(1+\varepsilon)$-ANN query can be answered in $O(n + d\log^3 n)$ time using roughly $O(dn)$ space. Later, Indyk and Motwani [IM98] and Kushilevitz et al. [KOR00] showed that polynomial space $n^{O(1/\varepsilon^2)}$ can be used to answer queries in time polynomial in d, $\log n$, and $1/\varepsilon$. The best result along these lines is that of Ailon and Chazelle [AC09], who showed that queries can be answered in time $O(d\log d + 1/\varepsilon^3 \log^2 n)$ using the same space. Techniques used in the above solutions include random projection (for example, dimension reduction and locality sensitive hashing [DIIM04]) and divide-and-conquer.

General metric spaces. The ANN problem has been studied for more general metric spaces, than Euclidean spaces. Data structures for proximity searching in metric spaces have been known for some time, see for example, [Bri95, FS82, Yia93]. Clarkson [Cla99] and later Karger and Ruhl [KR02] designed models to capture the sphere packing and local growth properties of low dimensional Euclidean spaces and studied their relation to the ANN problem. Much of the recent work has focused on metric spaces of low doubling dimension [Ass83]. The ***doubling dimension*** of a metric space is the minimum value τ such that every ball in the space can be covered by 2^τ balls of half the radius. This model was applied to various proximity problems by Krauthgamer, Lee, and co-authors [GKL03, KL04, KL05, KL06]. The results have been extended by Har-Peled and Mendel [HM06] and others [BKL06, CG06]. A thorough survey on metric space dimensions related to nearest-neighbor searching can be found in Clarksons paper [Cla06]. The results described in these papers on doubling spaces apply in the ***black box model***, in which points of the space can only be accessed through a black box that computes the distance between any two points in constant time. The model thus relies only on the barest set of assumptions, and so it is possible to obtain the conceptually

simplest and most general algorithms. In this model, Har-Peled and Mendel [HM06] have shown that, given a set P of n points in a metric space of doubling dimension τ, $(1+\varepsilon)$-ANN queries can be answered in time $O(\log n) + 1/\varepsilon^{O(\tau)}$ using a data-structure of linear space. Cole and Gottlieb [CG06] presented a dynamic data-structure with similar query time and size. Their data-structure can also support point insertion and deletion in $O(\log n)$ time. (These asymptotic bounds hide multiplicative factors that depend on the doubling dimension, except for the space bounds of Cole and Gottlieb [CG06], which are truly $O(n)$, irrespective of the dimension.)

1.1.1 Abstract proximity search

We now define the proximity search problem more abstractly, including its approximate version. Proximity search can be formulated as the following abstract problem. Given two sets $\mathcal{Q}$ and $\mathcal{X}$, a ***separation function*** $\psi : \mathcal{Q} \times \mathcal{X} \to \mathbb{R}^+$, and a set of n ***data points*** $\mathcal{D} = \{\mathrm{d}_1, \ldots, \mathrm{d}_n\} \subseteq \mathcal{X}$, one is required to pre-process the data points such that given a ***query point*** $\mathrm{q} \in \mathcal{Q}$ one can output the element $\mathrm{d} \in \{\mathrm{d}_1, \ldots, \mathrm{d}_n\}$ such that $\psi(\mathrm{q}, \mathrm{d})$ is the minimum of the numbers $\psi(\mathrm{q}, \mathrm{d}_1), \ldots, \psi(\mathrm{q}, \mathrm{d}_n)$. Yet another generalization of this problem is similar to a rank query. We are additionally given an integer k with $1 \leq k \leq n$ and we must return the element $\mathrm{d} \in \{\mathrm{d}_1, \ldots, \mathrm{d}_n\}$ such that $\psi(\mathrm{q}, \mathrm{d})$ has rank k in the list of numbers $\psi(\mathrm{q}, \mathrm{d}_1), \ldots, \psi(\mathrm{q}, \mathrm{d}_n)$. For $k = 1$ this problem is known as the nearest-neighbor problem (**NN** ***problem***) and for $k > 1$ as the k-nearest neighbor problem (k-**NN** problem). The computational model assumes that computing the separation function $\psi(\mathrm{q}, \mathrm{d})$ given any $\mathrm{q} \in \mathcal{Q}$ and $\mathrm{d} \in \mathcal{X}$ requires $O(1)$ time. In general, the space $\mathcal{Q}$ can be infinite, and there is no hope of storing precomputed answers for all possible query points. On the other hand, a simple $O(n)$ time algorithm exists, that computes $\psi(\mathrm{q}, \mathrm{d}_i)$ for $i = 1, 2, \ldots, n$ and does a rank selection query. In the most general case, not much can be done beyond this trivial algorithm, as a simple adversarial argument can show. However, by exploiting the structure of the spaces $\mathcal{Q}$ and $\mathcal{X}$, and the separation function ψ, one can hope to design faster query algorithms. In typical applications however, it is sufficient to work with an approximate answer to a proximity query. For an approximate query for some k where $1 \leq k \leq n$, one is additionally specified a real $\varepsilon > 0$ and one needs to return an object $\mathrm{d} \in \{\mathrm{d}_1, \ldots, \mathrm{d}_n\}$ such that,

$$\psi_k(\mathrm{q}, \mathcal{X})/(1+\varepsilon) \leq \psi(\mathrm{q}, \mathrm{d}) \leq (1+\varepsilon)\psi_k(\mathrm{q}, \mathcal{X})$$

where $\psi_k(\mathrm{q}, \mathcal{X})$ is the kth smallest number among the numbers, $\psi(\mathrm{q}, \mathrm{d}_1), \ldots, \psi(\mathrm{q}, \mathrm{d}_n)$. When, $\varepsilon \leq 1$, as is often the case where approximations are used, a slightly weaker condition is sometimes required,

$$(1-\varepsilon)\psi_k(\mathrm{q}, \mathcal{X}) \leq \psi(\mathrm{q}, \mathrm{d}) \leq (1+\varepsilon)\psi_k(\mathrm{q}, \mathcal{X}).$$

This problem is the approximate nearest-neighbor problem (**ANN** ***problem***) for $k = 1$ and the k-approximate nearest-neighbor problem (k-**ANN** ***problem***) for $k > 1$.

We now describe the problems that have been studied in this thesis.

1.2 Low Dimensional Queries

A natural setting for the ANN problem is the case when the data (or the queries) come from a low dimensional subspace that lies inside a high dimensional ambient space. Such cases are interesting as it is widely believed that in practice, real world data usually lies on a low dimensional manifold (or is close to lying on such a manifold). Such low-dimensionality arises from the way the data is being acquired, inherent dependency between parameters, aggregation of data that leads to concentration of mass phenomena, etc.

Indyk and Naor [IN07] showed that if the data is in high dimensional Euclidean space, but lies on a manifold with low doubling dimension, then one can do a dimension reduction into constant dimension (i.e., similar in spirit to the Johnson Lindenstrauss lemma [JL84]), such that a $(1 + \varepsilon)$-ANN to a query point (the query point might lie anywhere in the ambient space) is preserved with constant probability. Using an appropriate data-structure on the embedded space and repeating this process sufficient number of times, results in a data-structure that can answer ANN queries in polylog time (ignoring the dependency on ε).

1.2.1 Our Results

We study the "reverse" problem. Here the given set P of points, lies (possibly) in a high dimensional space, and we would like to pre-process it for ANN queries, where the queries come from a manifold of low doubling dimension. The question arises naturally when the given data is formed by merging together a large number of data sets, while the ANN queries come from a single data set.

In particular, the question here is whether this problem is low or high dimensional in nature. Note that direct dimension reduction as done by Indyk and Naor would not work in this case. Indeed, imagine the data lies densely on a slightly deformed sphere in high dimensions, and the query is the center of the sphere. Clearly, a random dimension reduction via projection into constant dimension would not preserve the $(1 + \varepsilon)$-ANN.

Given the set P lying in a general metric space $\mathcal{X}$ (which is not necessarily Euclidean and is conceptually high dimensional), and a subspace $\mathcal{M}$ having low doubling dimension τ, we show how to pre-process P such that given any query point in $\mathcal{M}$ we can quickly answer $(1 + \varepsilon)$-ANN queries on P. In particular, we get a data-structure of (roughly) linear size that can answer $(1 + \varepsilon)$-ANN queries in (roughly) logarithmic time.

However, we do require access to the query subspace in a non black box fashion.

Our construction uses ideas developed for handling the low dimensional case, i.e., where both the data and query come from the same space of low doubling dimension. Initially, we embed P and $\mathcal{M}$ into a space with low doubling dimension that (roughly) preserves distances between points in $\mathcal{M}$ and points in P. We can use the embedded space to answer constant factor ANN queries. Getting a better approximation requires some further ideas. In particular, we build a data-structure over $\mathcal{M}$ that is somewhat similar to approximate Voronoi diagrams [Har01]. By sprinkling points carefully on the subspace $\mathcal{M}$ and using the net-tree data-structure of Har-Peled and Mendel [HM06], we can answer $(1+\varepsilon)$-ANN queries in time $O(\varepsilon^{-O(\tau)} + 2^{O(\tau)} \log n)$.

To get a better query time requires some further work. In particular, we borrow ideas from the simplified construction of Arya and Malamatos [AM02] (see also [AMM09]). Naively, this requires us to use well separated pairs decomposition (i.e., WSPD) [CK95] for P. Unfortunately, no such small WSPD (i.e., of linear size) exists for data in high dimensions. To overcome this problem, we build the WSPD in the embedded space. Next, we use this to guide us in the construction of the ANN data-structure. This improved construction requires space $n\varepsilon^{-O(\tau)}$ and queries can be answered in time $2^{O(\tau)} \log n$.

We also present an algorithm for a weaker model similar to the black box model. Every time an ANN query is issued, the algorithm computes a region around the query point such that the returned point is a valid ANN for all the points in this region. Furthermore, the algorithm caches such regions, and whenever a query arrives it first checks if the query point is already contained in one of the regions computed, and if so it answers the ANN query immediately; otherwise it finds an ANN and also computes a region where the found ANN is a valid ANN, and then adds it to the set of regions. Significantly, for this algorithm we need no pre-specified knowledge about the query subspace. The resulting algorithm computes on the fly, an AVD on the query subspace. In particular, we show that if the queries come from a subspace with doubling dimension τ, then the algorithm would create at most $n/\varepsilon^{O(\tau)}$ regions overall. A limitation of this new algorithm is that we currently do not know how to efficiently perform a point location query in a set of such regions, without assuming further knowledge about the subspace. Interestingly, the new algorithm can be interpreted as learning the underlying manifold $\mathcal{M}$ the queries come from.

These results first appeared in the conference paper [HK11], and later as a journal paper [HK13a]. In this thesis, they are presented in Chapter 3.

1.3 k-ANN Queries in $\mathbb{R}^d$

Generalizing the ANN problem in $\mathbb{R}^d$, we study the k-ANN problem in $\mathbb{R}^d$. Here we are also given an integer k with $1 \leq k \leq n$, and we are required to find a point $\mathsf{p} \in \mathsf{P}$, whose distance from q is within a multiplicative factor of $1 \pm \varepsilon$ to the kth NN distance of q to P.

A data-structure of Arya *et al.* [AMM05], was already known for the case when ε is specified at the time of preprocessing (but k can be specified during query time). They present a data structure parametrized by a parameter γ with $2 \leq \gamma \leq 1/\varepsilon$ and achieving a natural space-time trade off with the variance in γ. At one end of the spectrum where $\gamma = 2$ they get a data-structure with space requirement $O(n \log(1/\varepsilon))$ and query time $O(\log n + 1/\varepsilon^d)$ for this problem.

1.3.1 Our Results

We study the problem and also a few applications. Our results are the following.

(i) For the case where k is specified with the query, we improve upon the result of Arya *et al.* [AMM05], by showing that the problem can be solved in roughly the same space-time bounds as facilitated by the method of Arya *et al.* [AMM05], *even* if ε is specified only during query. This makes the algorithm more practical than the algorithm of Arya *et al.*.

(ii) Similar to AVD, one can define the concept of AVD for the k-ANN problem. The case $k = 1$ is the regular approximate Voronoi diagram [Har01, AM02, AMM09]. The case $k = n$ is the furthest-neighbor Voronoi diagram. It is not hard to see that it has a constant size approximation (see [Har99], although it was probably known before). We show that this AVD has complexity $\widetilde{O}(n/k)$.

(iii) We show that certain kinds of problems, which relate to density estimation, can be solved more efficiently than known before, using our data-structure. An example is to compute the function, which for a given query point q returns the sum of the distances of q to its k nearest-neighbors in P.

These results first appeared in the conference paper [HK12]. In this thesis, they are presented in Chapter 4.

1.4 k-ANN Queries for Balls in $\mathbb{R}^d$

Many of the results for k-ANN queries can be extended to the following generalization of the problem. Here, instead of points as our input, we have balls, that are disjoint from each other, but the distance of a query point q to a ball is the distance to its boundary if it lies outside the ball, or 0 if it lies inside the ball.

1.4.1 Our Results

We show that the results for points can be extended to the case for balls. More specifically, we show that one can construct an AVD for this problem of complexity $\widetilde{O}(n/k)$.

These results are presented in Chapter 5.

1.5 Approximating Lower Envelopes

The algorithms for proximity search generally rely heavily on the triangle inequality. One possible generalization is to consider separation function as some "distance" where the triangle inequality holds only approximately, for example, squared distances. However, these variations can typically be handled by small modifications in the algorithms.

Such general problems can be posed more cleanly in the language of minimization diagrams. The problem of approximating minimization diagrams, also known as lower envelopes, is a far reaching generalization of proximity search. We would like to understand, for what classes of functions, can we approximate their minimization diagrams with the desirable space and query time bounds?

1.5.1 Our Results

Our contributions to this problem are the following: We provide a set of sufficient conditions, and show that if these conditions are satisfied by a class of functions, then for any set of n functions from the class, one can pre-process them into a data-structure with the desirable space bound of $O(n\text{polylog}(n))$ and allowing logarithmic query time for approximating the lower envelope.

Of main interest are applications of this result to concrete classes of functions. We describe here at a very high and intuitive level a few applications given in this thesis.

(i) We consider the problem of approximate multiplicative weighted Voronoi diagrams, and show that our framework applies to the class of functions defining multiplicative weighted Voronoi diagrams.

(ii) We consider the scaling distance function defined by a certain class of objects. In particular, the distance defined by Minkowski metrics of symmetric convex bodies with constant boundary complexity can be handled by our framework, and we get a data-structure for answering ANN queries for such a metric.

(iii) For a set of points the furthest-neighbor distance can be approximated easily. We consider the problem, where we are given a set of point sets, and we want to find the point set, whose furthest-neighbor

distance is the smallest, approximately. This problem, the nearest furthest-neighbor problem arises naturally when trying to model uncertainty. We show how to solve this problem approximately using our framework.

These results first appeared in the conference paper [HK13b]. In this thesis, they are presented in Chapter 6.

Chapter 2

Preliminaries

This chapter discusses some defintions and data-structures which are used in the later chapters. The discussion here is not a survey of the material. They serve as short introductions, and include the appropriate references.

2.1 Notation and basic definitions

Let $\mathcal{X}$ denote a metric space. We denote the distance between two points $\mathsf{p}, \mathsf{q} \in \mathcal{X}$ by $\mathsf{d}(\mathsf{p}, \mathsf{q})$. If the metric space is the Euclidean space $\mathbb{R}^d$ for some d, sometimes we also use $\|\mathsf{p} - \mathsf{q}\|$ to denote the distance.

The following definition captures the range of distances for a set of points.

Definition 2.1.1 For a point set P, the ***spread*** is the ratio $\frac{\max_{\mathsf{p},v \in \mathsf{P}} \mathsf{d}(\mathsf{p},v)}{\min_{\mathsf{p},v \in \mathsf{P}, \mathsf{p} \neq v} \mathsf{d}(\mathsf{p},v)}$.

2.2 Model of computation

For all our algorithms and data-structures our model of computation is the real RAM model, where one can store and manipulate arbitrarily large real numbers in constant time. As we also use grids, we need to be able to compute in constant time $\lg(x)$ (i.e., $\log_2 x$), 2^x and $\lfloor x \rfloor$. For quadtrees, see below, we also need the following operation to be done in constant time. It is equivalent to the assumption that for 2 given points we can construct their quadtree in constant time, see Har-Peled's book [Har11] for details.

Definition 2.2.1 (Bit index) Let $\alpha, \beta \in [0,1)$ be two real numbers. When written in base two, where for definiteness we assume an infinite sequence of trailing 1's does not occur, suppose $\alpha = 0.\alpha_1\alpha_2\ldots$ and $\beta = 0.\beta_1\beta_2\ldots$. Let $\mathsf{bit}_\Delta(\alpha, \beta)$ be the index of the first bit after the period where they differ.

Assumption 2.2.2 *Given two numbers $\alpha, \beta \in [0,1)$ where $\alpha \neq \beta$. Then we can compute $\mathsf{bit}_\Delta(\alpha, \beta)$ in $O(1)$ time.*

2.3 Quadtrees

The quadtree is a data-structure used for point location in $\mathbb{R}^d$. A quadtree is imposed on a finite region of space, say the unit cube $[0,1]^d$. Conceptually, it is a division of this region into a hierarchy of cells. At any level of the tree the cells are of the same size, which is a power of two. All cells of the same size are interior disjoint and cover the given area. The natural tree structure, based on the partial order of the containment relation on such a collection of cells, is the quadtree. Given a finite point set, if one considers such an infinite hierarchy of cells and clips it at the right level, it can be seen as a multi-grid representation of the point set, where all distances between the sets of points have a corresponding level in the quadtree, whose cell size matches the distance up to a constant factor. If the spread of distances is large, see Definition 2.1.1, a quadtree for the set of points can be very deep. To go around this problem, a quadtree can be compressed quite easily and one can have the desired multi-grid representation. Such a compressed quadtree takes $O(n)$ space for storage. Every node in a compressed quadtree has a region of space associated with it. For a quadtree, such a region is always a canonical cube, i.e., a cube whose sidelength is a power of two and which lies inside the unit cube. But for a compressed quadtree, one can have a region which is the set difference of two such cubes, associated with it. For a node ν of a quadtree let $\square_\nu$ be the region associated with it. We state several results for quadtrees. The user is referred to the book [Har11] for details and proofs.

Theorem 2.3.1 ([Har11]) *Given a set P of n points in the unit cube $[0,1]^d$, one can construct a compressed quadtree of P in $O(n \log n)$ deterministic time.*

In the above construction, one is required to use the assumption about bit index, see Definition 2.2.1. Occasionally, we also use the following very useful result.

Theorem 2.3.2 ([Har11]) *Given a list $\mathcal{C}$ of n canonical cubes (i.e., whose sides are powers of two), all lying inside the unit cube, one can construct a minimal compressed quadtree $\mathcal{T}$ such that for any cell $\mathsf{c} \in \mathcal{C}$, there exists a node $\nu \in \mathcal{T}$, such that $\square_\nu = \mathsf{c}$. The construction time is $O(n \log n)$.*

Finally, we have the following result, which justifies the quadtree as a point location data-structure.

Theorem 2.3.3 ([Har11]) *Given a compressed quadtree $\mathcal{T}$ of size n, one can preprocess it in time $O(n \log n)$, such that given a query point q, one can return the lowest node in $\mathcal{T}$ whose region contains q in $O(\log n)$ time.*

2.4 Nets in a Metric Space

Given a set of points P in a metric space $\mathcal{X}$, an r-net for P, is a set of points $\mathsf{Q} \subseteq \mathcal{X}$, such that for any different $\mathsf{p}, v \in \mathsf{Q}$ we have that $\mathsf{d}(\mathsf{p}, v) > r$ and at the same time $\mathsf{Q} \subseteq \bigcup_{\mathsf{p} \in \mathsf{Q}} \mathsf{ball}(\mathsf{p}, r)$, i.e., the balls of radius r around points in Q cover P. Such a set of points, represents well the set of points P at scale r. We note that it is possible to compute such a set of points, which is also a subset of P, using a greedy algorithm.

2.5 Grids

Grids are an extremely useful construct used to discretize a continuous region of space. For a real positive number x and a point $\mathsf{p} = (\mathsf{p}_1, \ldots, \mathsf{p}_d) \in \mathbb{R}^d$, define $\mathsf{G}_x(\mathsf{p})$ to be the grid point $(\lfloor \mathsf{p}_1/x \rfloor\, x, \ldots, \lfloor \mathsf{p}_d/x \rfloor\, x)$. The number x is the ***width*** or ***sidelength*** of the ***grid*** G_x. The mapping G_x partitions $\mathbb{R}^d$ into cubes that are called grid ***cells***.

Definition 2.5.1 A cube is a ***canonical cube*** if it is contained inside the unit cube $U = [0,1]^d$, it is a cell in a grid G_r, and r is a power of two (i.e., it might correspond to a node in a quadtree having $[0,1]^d$ as its root cell). We will refer to such a grid G_r as a ***canonical grid***. Note that all the cells corresponding to nodes of a compressed quadtree are canonical.

We will often approximate sets in $\mathbb{R}^d$ by the set of grid cells that intersect it. The union of these cells contains the set under concern, and is hopefully not too large if the width of the grid is small. The following captures this, where the sidelength of the grid is chosen in proportion to the diameter of the set being approximated.

Definition 2.5.2 Given a set $\mathsf{b} \subseteq \mathbb{R}^d$, and a parameter $\delta > 0$, let $\mathsf{G}_{\approx}(\mathsf{b}, \delta)$ denote the set of canonical grid cells of sidelength $2^{\lfloor \log_2 \delta \mathrm{diam}(\mathsf{b})/\sqrt{d} \rfloor}$, that intersect b, where $\mathrm{diam}(\mathsf{b}) = \max_{\mathsf{p},\mathsf{u} \in \mathsf{b}} \|\mathsf{u} - \mathsf{p}\|$ denotes the ***diameter*** of b. Clearly, the diameter of any grid cell of $\mathsf{G}_{\approx}(\mathsf{b}, \delta)$, is at most $\delta \mathrm{diam}(\mathsf{b})$. Let $\mathsf{G}_{\approx}(\mathsf{b}) = \mathsf{G}_{\approx}(\mathsf{b}, 1)$. It is easy to verify that $|\mathsf{G}_{\approx}(\mathsf{b})| = O(1)$. The set $\mathsf{G}_{\approx}(\mathsf{b})$ is the ***grid approximation*** to b.

Sometimes it is easier to use an absolute value for the width of the grid, i.e., not defining it in terms of the set being approximated.

Definition 2.5.3 To approximate a set $X \subseteq [0,1]^d$, up to distance r, consider the set $\mathsf{G}_{\approx r}(X)$ of all the canonical grid cells of G_ℓ, that have a non-empty intersection with X, where $\ell = 2^{\lfloor \log_2 (r/\sqrt{d}) \rfloor}$. Let $\cup \mathsf{G}_{\approx r}(X) = \bigcup_{\square \in \mathsf{G}_{\approx r}(X)} \square$, denote the union of cubes of $\mathsf{G}_{\approx r}(X)$. Each cube of $\mathsf{G}_{\approx r}(X)$ has diameter at most r.

Observe that $X \subseteq \cup \mathsf{G}_{\approx r}(X) \subseteq X \oplus \mathsf{ball}(0,r)$, where $\oplus$ denotes the Minkowski sum, and $\mathsf{ball}(0,r)$ is the ball of radius r centered at the origin.

Definition 2.5.4 For a ball b of radius r, and a parameter ψ, let $\boxplus(\mathsf{b}, \psi)$ denote the set of all the canonical cells intersecting b, when considering the canonical grid with sidelength $2^{\lfloor \log_2 \psi \rfloor}$.

Clearly, $|\boxplus(\mathsf{b}, \psi)| = O\big((r/\psi)^d\big)$, for $\psi \leq r$.

2.6 Well Separated Pairs Decomposition

Given a set of points P in a metric space $\mathcal{X}$, they define $n(n-1)/2$ pairwise distances between different points. By rescaling we can assume that the minimum distance is 1. In this case, the spread, see Definition 2.1.1 gives the maximum distance among the points. However, in certain applications, we need a good idea of all the different possible distances. In general all the $n(n-1)/2$ distances can be different. However, we can say something interesting if we bucketize these distances and for a bucket we mandate that all distances vary by at most a multiplicative factor of $(1+\varepsilon)$. Clearly, a possible bucketization is to put every distance in its own bucket. However, that is still $\Omega(n^2)$ buckets. For algorithmic applications, we would want the smallest number of buckets. This is not always possible, but in Euclidean spaces it is possible. The definition of a well-separated pair decomposition (see below), enables us to give further structure for such buckets. These buckets represent distances of points between small clusters of points.

For a point set P, a ***pair decomposition*** of P is a set of pairs $\mathcal{W} = \Big\{ \{A_1, B_1\}, \ldots, \{A_s, B_s\} \Big\}$, such that (I) $A_i, B_i \subset \mathsf{P}$ for every i, (II) $A_i \cap B_i = \emptyset$ for every i, and (III) $\cup_{i=1}^{s} A_i \otimes B_i = \mathsf{P} \otimes \mathsf{P}$. Here $X \otimes Y = \Big\{ \{x,y\} \;\Big|\; x \in X, y \in Y, \text{ and } x \neq y \Big\}$.

A pair $\mathsf{Q} \subseteq \mathsf{P}$ and $\mathsf{R} \subseteq \mathsf{P}$ is $(1/\varepsilon)$***-separated*** if $\max(\mathrm{diam}(\mathsf{Q}), \mathrm{diam}(\mathsf{R})) \leq \varepsilon \cdot \mathsf{d}(\mathsf{Q}, \mathsf{R})$, where $\mathsf{d}(\mathsf{Q}, \mathsf{R}) = \min_{\mathsf{p} \in \mathsf{Q}, v \in \mathsf{R}} \mathsf{d}(\mathsf{p}, v)$.

Definition 2.6.1 For a point set P, a ***well-separated pair decomposition*** (**WSPD**) of P with parameter $1/\varepsilon$ is a pair decomposition of P with a set of pairs $\mathcal{W} = \{\{A_1, B_1\}, \ldots, \{A_s, B_s\}\}$, such that, for any i, the sets A_i and B_i are ε^{-1}-separated [CK95].

We have the following result, which makes this concept very useful for Euclidean spaces.

Theorem 2.6.2 ([CK95, Har11]) *Given a set P of n points in $\mathbb{R}^d$ under the Euclidean metric, and ε with $0 < \varepsilon \leq 1$, one can construct a ε^{-1}-WSPD of size $n\varepsilon^{-d}$, and the construction time is $O(n \log n + n\varepsilon^{-d})$.*

2.7 Approximate Voronoi Diagrams(AVD)

The approximate version of a regular Voronoi diagram is an approximate Voronoi diagram(AVD). For a set P of n points in $\mathbb{R}^d$, a $(1+\varepsilon)$-AVD is a partition of $\mathbb{R}^d$ into regions which are interior disjoint, and for each region there is an associated point $\mathsf{p} \in \mathsf{P}$, such that for any point q in the region, we have that $\mathsf{d}(\mathsf{q},\mathsf{p}) \leq (1+\varepsilon)\mathsf{d}(\mathsf{q},\mathsf{P})$. A point location data-structure for an AVD, immediately gives us a data-structure for the approximate nearest-neighbor problem. AVD's were first defined by Har-Peled [Har01], see also the book [Har11] and the journal version of the paper [HIM12]. Har-Peled showed that a $(1+\varepsilon)$-AVD can be constructed such that the total complexity of the Voronoi diagram is $O(n\varepsilon^{-d}\log^2 n)$. Here the constant in the $O(\cdot)$ expression are exponential in d. However, this is surprising, as the complexity is near-linear in n, while that of an exact Voronoi diagram is $\Theta(n^{\lceil d/2 \rceil})$, which even for $d = 3$ is quadratic in n. Sabharwal [SSS02] showed that an AVD of complexity $O(n \log n \varepsilon^{-d})$ can be constructed, thus improving the bound by a logarithmic factor. The best result was achieved by Arya and Malamatos, who gave a conceptually very simple construction of AVD's which achieve optimal complexity as a function of n. Their main result implies the following, which we state in slightly less general form than stated in the paper;

Theorem 2.7.1 *Let P be a set of n points in $\mathbb{R}^d$, and $0 < \varepsilon \leq 1/2$. One can construct a $(1+\varepsilon)$-AVD for P with $O(n\varepsilon^{-d})$ regions, where each region is the difference of two cubes. Moreover, for any query point, its $(1+\varepsilon)$-ANN can be returned in $O(\log(n/\varepsilon))$ time.*

In the next section, we describe at an intuitive level, their construction. For details, one can look at their paper [AM02].

2.7.1 The Arya and Malamatos Construction

Arya and Malamatos roughly do the following to construct a $(1+\varepsilon)$-AVD. First, they compute a ε/c-WSPD for some large enough constant c. Next, for each pair (A_i, B_i) of the WSPD, compute approximately their distance D_i. To this end, take any two points, a, b with $a \in A_i$, $b \in B_i$ and let $\mathsf{d}(a,b) = D_i$: this guarantees by property of a WSPD that it holds approximately for any $a' \in A_i, b' \in B_i$ that $\mathsf{d}(a',b') \approx D_i$. Now, consider $\mathsf{ball}\big(a, 2^j D_i/4\big)$ for $j = 0, 1, \ldots,$ so that the final radius of the ball around a is roughly D_i/ε and similarly balls of these radii around b. We tile $\mathsf{ball}\big(a, 2^j D_i/4\big)$ by quadtree cells of size roughly $\varepsilon 2^j D_i$. The union of all cells, arising from all such balls around a and b are thrown into a quadtree for point location, i.e., given a query q we would need to output the "innermost" or smallest cell containing it, or report that q is outside the union. Also, for each of these cells, we need to choose an arbitrary point (the representative of the cell) and a ANN among P for the representative point, which can be computed, say, by the algorithm of

Arya et. al. [AMN+98]. For a query, once we locate the smallest cell containing it we can simply report the ANN stored for its representative point. To see why it works at an intuitive level, notice that in the region, which is the union of the balls given, except for the smallest one (i.e., from the union we remove the region occupied by the smallest balls), the AVD intuitively does not have any "challenge" in deciding between any point of A_i and any point of B_i. This is true because the smallest distance to $A_i \cup B_i$ is roughly at least $D_i/4$ (or $2^j D_i/4$, if the query lies in a further off region), while the smallest cell containing the query point is roughly ε times this number. As such, by Lipschitzness of distance, the reported ANN is correct (as far as competition between A_i, B_i is concerned). For a more global argument, one simply sees that all pairs of points occur in some such pair, as such competition between any pair of points is resolved correctly for the query point.

2.8 Doubling Dimension

The real space of d dimensions has a natural concept of dimension, coming from the underlying affine space $\mathbb{R}^d$. When the space is endowed with a norm, such as the usual ℓ_2 Euclidean norm, the dimension d plays a key role in several metrical concepts. However, going further it ties up with the natural volume (Lebesgue) measure on the space through packing bounds. Thus, for example one has the following phenomena: The number of dilates of an object by a factor $1/2$ that are disjoint and contained within the object is at most 2^d. This follows by a simple volume packing bound. Many algorithms in Computational Geometry, and most particularly approximation algorithms make use of such packing bounds to obtain bounds on the size of ε-nets in the space. Intuitively, the following principle is a powerful workhorse of approximation algorithms. Suppose one knows that the solution to a problem has size at least $\Omega(R)$. If to find an approximate solution, one needs to explore a space of diameter $O(R)$, one can restrict oneself to examining the space at a scale of resolution roughly εR, where $(1 + \varepsilon)$ is the degree of approximation desired. By the packing bounds mentioned above, this leads to the examination of a constant sized set of points. In trying to extend such a phenomena to general metric spaces, we are immediately faced with several problems. Firstly, there is no intrinsic notion of dimension and secondly, there is no notion of volume. However, the following definition captures both of these concepts with surprising efficacy.

Definition 2.8.1 A metric space $\mathcal{X}$ is said to be of doubling dimension τ if every ball of radius R in the space can be covered by 2^τ balls of radius $R/2$.

Note that in the above definition we do not require the covering balls to be disjoint. The strength of the above definition is that it can even apply to discrete or finite metric spaces. A few simple properties are the

following, which attest to the strength of this definition, and point to the fact that the definition is really talking about the intrinsic dimension.

Lemma 2.8.2 *For any finite metric space on n points, the doubling dimension is bounded by* $\log n$.

If one recalls the JL lemma, one sees that heuristically, it is a good idea to think of a n point metric space, that is Euclidean, as being of dimension roughly $O(\log n)$. The doubling dimension thus ties up nicely with this. We also have,

Lemma 2.8.3 *The dimension of the Euclidean space $\mathbb{R}^d$ is $\Theta(d)$.*

Thus, the doubling dimension ties up nicely with the usual concept of dimension for Euclidean spaces, as well. The following result is elementary.

Lemma 2.8.4 *Let $\mathcal{M}$ be a metric space of doubling dimension τ and $\mathsf{P} \subseteq \mathcal{M}$ be a point set with spread λ. Then $|\mathsf{P}| \leq \lambda^{O(\tau)}$.*

2.9 Net-trees

The net-tree [HM06] is a data-structure that defines hierarchical nets in finite metric spaces. Formally, a net-tree is defined as follows: Let $\mathsf{P} \subseteq \mathcal{M}$ be a finite subset. A net-tree of P is a tree T, whose set of leaves is P. Denote by P_v the set of leaves in the subtree rooted at a vertex $v \in T$. With each vertex v is associated a point $\mathrm{rep}_v \in \mathsf{P}_v$. Internal vertices have at least two children. Each vertex v has a level $l(v) \in \mathbb{Z} \cup \{-\infty\}$. The levels satisfy $l(v) < l(\overline{p}(v))$, where $\overline{p}(v)$ is the parent of v in T. The levels of the leaves are $-\infty$. Let γ be some large enough constant, say $\gamma = 11$. The following properties are satisfied: (I) For every vertex $v \in T$, $\mathsf{ball}\Big(\mathrm{rep}_v, \frac{2\gamma}{\gamma-1}\gamma^{l(v)}\Big) \supseteq \mathsf{P}_v$, (II) For every vertex $v \in T$ that is not the root, $\mathsf{ball}\Big(\mathrm{rep}_v, \frac{\gamma-5}{2(\gamma-1)}\gamma^{l(\overline{p}(v))-1}\Big) \cap \mathsf{P} \subseteq \mathsf{P}_v$, (III) For every internal vertex $u \in T$, there exists a child $v \in T$ of u such that $\mathrm{rep}_u = \mathrm{rep}_v$. The following construction result by Har-Peled and Mendel shows that net-trees can actually be constructed efficiently by a randomized algorithm.

Theorem 2.9.1 ([HM06]) *Given a set P of n points with doubling dimension τ, one can construct a net-tree for P in $2^{O(\tau)} n \log n$ expected time.*

Notice that in the above result, we use the intrinsic doubling dimension of the point set P. The algorithms thus only depend on the point set, and do not use properties of any ambient space in which the point set may lie. The following results, also from [HM06], shows that net-trees can be used to construct data-structures for answering ANN queries, and to construct WSPD's in metric spaces of low doubling dimension τ, efficiently.

Theorem 2.9.2 ([HM06]) *Given a set* P *of* n *points with doubling dimension* τ*, lying inside a metric space* $\mathcal{X}$*, one can construct a data-structure for answering* ANN *queries, where the quality parameter* ε *is provided together with the query. The query time is* $2^{O(\tau)} \log n + \varepsilon^{-O(\tau)}$*, the expected preprocessing time is* $2^{O(\tau)} n \log n$*, and the space used is* $2^{O(\tau)} n$*.*

Notice that, in the above result, the ambient space $\mathcal{X}$ does not necessarily have bounded doubling dimension. The following result regarding computation of WSPD's in spaces $\mathcal{X}$ of doubling dimension τ, generalizes a result of Talwar [Tal04], in which the number of WSPD pairs depended on the spread of the point set.

Theorem 2.9.3 ([HM06]) *For* $1 \geq \varepsilon > 0$*, one can construct an* ε^{-1}*-*WSPD *of size* $n\varepsilon^{-O(\tau)}$*, and the expected construction time is* $2^{O(\tau)} n \log n + n\varepsilon^{-O(\tau)}$*. Furthermore, the pairs of the* WSPD *correspond to* $(\mathsf{P}_u, \mathsf{P}_v)$*, where* u, v *are vertices of a net-tree of* P*, and for any pair* $(\mathsf{P}_u, \mathsf{P}_v)$ *in the* WSPD*, we have* $\operatorname{diam}(\mathsf{P}_u), \operatorname{diam}(\mathsf{P}_v) \leq \varepsilon \mathsf{d}(u, v)$*.*

Chapter 3

Low Dimensional Queries

3.1 Problem and Model

The Problem. We look at the ANN problem in the following setting. Given a set P of n data points in a metric space $\mathcal{X}$, and a set $\mathcal{M} \subseteq \mathcal{X}$ of (hopefully low) doubling dimension τ, and $\varepsilon > 0$, we want to preprocess the points of P, such that given a query point $\mathsf{q} \in \mathcal{M}$ one can efficiently find a $(1+\varepsilon)$-ANN of q in P.

Model. We are given a metric space $\mathcal{X}$ and a subset $\mathcal{M} \subseteq \mathcal{X}$ of doubling dimension τ. We assume that the distance between any pair of points can be computed in constant time in a black-box fashion. Specifically, for any $\mathsf{p}, q \in \mathcal{X}$ we denote by $\mathsf{d}(\mathsf{p}, q)$ the distance between p and q. We also assume that one can build nets on $\mathcal{M}$. Specifically, given a point $\mathsf{p} \in \mathcal{M}$ and a radius $r > 0$, we assume we can compute 2^τ points $\mathsf{p}_i \in \mathcal{M}$, such that $\mathsf{ball}(\mathsf{p}, r) \cap \mathcal{M} \subseteq \bigcup \mathsf{ball}(\mathsf{p}_i, r/2)$. By applying this recursively we can compute an ***r-net*** N for any $\mathsf{ball}(\mathsf{p}, R)$ centered at p; that is, for any point $v \in \mathsf{ball}(\mathsf{p}, R)$ there exists a point $u \in N$ such that $\mathsf{d}(v, u) \leq r$. Let **compNet**(p, R, r) denote this algorithm for computing an r-net. The size of N is $(R/r)^{O(\tau)}$, and we assume this also bounds the time it takes to compute it. For example, in Euclidean space $\mathbb{R}^d$, let p be the origin and consider the tiling of space by a grid of cubes of diameter r. One can compute an r-net, by simply enumerating all the vertices of the grid cells that intersect the cube $\left[-R, R\right]^d$ surrounding $\mathsf{ball}(\mathsf{p}, R) = \mathsf{ball}(0, R)$.

Finally, given any point $\mathsf{p} \in \mathcal{X}$ we assume that one can compute, in $O(1)$ time, a point $\alpha(\mathsf{p}) \in \mathcal{M}$ such that $\alpha(\mathsf{p})$ is the closest point in $\mathcal{M}$ to p. (Alternatively, $\alpha(\mathsf{p})$ might be specified for each point of P in advance.)

3.1.1 Warm-up exercise: Affine Subspace

We first consider the case where our query subspace is an affine subspace embedded in d dimensional Euclidean space. Thus let $\mathcal{X} = \mathbb{R}^d$ with the usual Euclidean metric. Suppose our query subspace $\mathcal{M}$ is an affine subspace of dimension k where $k \ll d$. We are also given n data points $\mathsf{P} = \{\mathsf{p}_1, \mathsf{p}_2, \ldots, \mathsf{p}_n\}$. We want

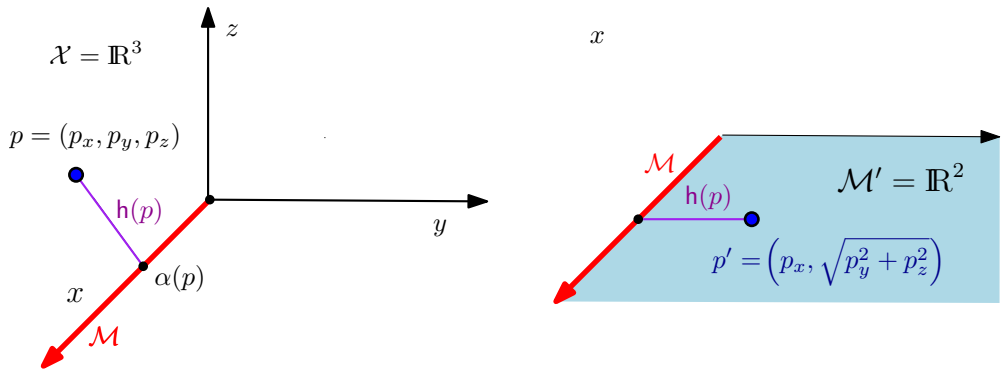


Figure 3.1: An example of embedding of space into two dimensions where $\mathcal{M}$ is the x-axis.

to preprocess P such that given a $\mathsf{q} \in \mathcal{M}$ we can quickly find a point $\mathsf{p}_i \in \mathsf{P}$ which is a $(1+\varepsilon)$-ANN of q in P.

We choose an orthonormal system of coordinates for $\mathcal{M}$. Denote the projection of a point p to $\mathcal{M}$ as $\alpha(\mathsf{p})$. Denote the coordinates of a point $\alpha(\mathsf{p}) \in \mathcal{M}$ in the chosen coordinate system as $(\mathsf{p}^1, \mathsf{p}^2, \ldots, \mathsf{p}^k)$. Let $\mathsf{h}(\mathsf{p})$ denote the distance of any $\mathsf{p} \in \mathbb{R}^d$ from the subspace $\mathcal{M}$. Notice that $\mathsf{h}(\mathsf{p}) = \|\alpha(\mathsf{p}) - \mathsf{p}\|$, and consider the following embedding.

Definition 3.1.1 For the point $\mathsf{p} \in \mathbb{R}^d$, the embedded point is $\mathsf{p}' = \left(\mathsf{p}^1, \mathsf{p}^2, \ldots, \mathsf{p}^k, \mathsf{h}(\mathsf{p})\right) \in \mathbb{R}^{k+1}$.

An example of the above embedding is shown in Figure 3.1. It is easy to see that for $x \in \mathcal{M}$ and $y \in \mathbb{R}^d$, by the Pythagorean theorem, we have $\|y - x\|^2 = \|\alpha(y) - x\|^2 + \|y - \alpha(y)\|^2 = \|\alpha(y) - x\|^2 + \mathsf{h}(y)^2 = \|y' - x'\|^2$. So, $\|y - x\| = \|y' - x'\|$. That is, the above embedding preserves the distances between points on $\mathcal{M}$ and any point in $\mathbb{R}^d$.

As such, given a query point $\mathsf{q} \in \mathcal{M}$, let p'_i be its $(1+\varepsilon)$-ANN in $\mathsf{P}' \subseteq \mathbb{R}^{k+1}$. Then the original point $\mathsf{p}_i \in \mathsf{P}$ (that generated p'_i) is a $(1+\varepsilon)$-ANN of q in the original space $\mathbb{R}^d$.

But this is easy to do using known data-structures for ANN [AMN+98], or the data-structures for approximate Voronoi diagram [Har01, AM02].

Thus, we have n points in $\mathbb{R}^{k+1}$ to preprocess and, without loss of generality, we can assume that p'_i are all distinct. Now given $\varepsilon \leq 1/2$, we can preprocess the points $\{\mathsf{p}'_1, \ldots, \mathsf{p}'_n\}$ and construct an approximate Voronoi diagram consisting of $O\left(n\varepsilon^{-(k+1)} \log \varepsilon^{-1}\right)$ regions [AM02]. Each such region is the difference of two cubes. Given a point $\mathsf{q}' \in \mathbb{R}^{k+1}$ we can find a $(1+\varepsilon)$-ANN in $O(\log(n/\varepsilon))$ time, using this data-structure.

3.1.2 An Embedding

Here, we show how to embed the points of P (and all of $\mathcal{X}$) into another metric space $\mathcal{M}'$ with finite doubling dimension, such that the distances between P and $\mathcal{M}$ are roughly preserved.

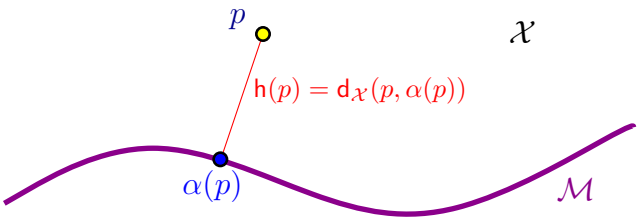


Figure 3.2: The quantities $\alpha(\mathsf{p})$ and $\mathsf{h}(\mathsf{p})$.

For a point $\mathsf{p} \in \mathcal{X}$, let $\alpha(\mathsf{p})$ denote the closest point in $\mathcal{M}$ to p (for the sake of simplicity of exposition we assume this point is unique). The ***height*** of a point $\mathsf{p} \in \mathcal{X}$ is the distance between p and $\alpha(\mathsf{p})$; namely, $\mathsf{h}(\mathsf{p}) = \mathsf{d}_{\mathcal{X}}(\mathsf{p}, \alpha(\mathsf{p}))$. For a set $\mathsf{B} \subseteq \mathcal{X}$, let $\alpha(\mathsf{B})$ denote the set $\left\{ \alpha(x) \;\middle|\; x \in \mathsf{B} \right\}$. An example is shown in Figure 3.2.

Definition 3.1.2 (*$\mathcal{M}'$ embedding*.) Consider the embedding of $\mathcal{X}$ into $\mathcal{M}' = \mathcal{M} \times \mathbb{R}^+$ induced by the distances of points of $\mathcal{X}$ from $\mathcal{M}$. Formally, for a point $\mathsf{p} \in \mathcal{X}$, the embedding is defined as

$$\mathsf{p}' = \Big(\alpha(\mathsf{p}),\, \mathsf{h}(\mathsf{p}) \Big) \in \mathcal{M}'.$$

The distance between any two points $\mathsf{p}' = (\alpha(\mathsf{p}), \mathsf{h}(\mathsf{p}))$ and $v' = (\alpha(v), \mathsf{h}(v))$ of $\mathcal{M}'$ is defined as

$$\mathsf{d}_{\mathcal{M}'}(\mathsf{p}', v') = \mathsf{d}_{\mathcal{X}}\Big(\alpha(\mathsf{p}), \alpha(v) \Big) + |\mathsf{h}(\mathsf{p}) - \mathsf{h}(v)| \,.$$

It is easy to verify that $\mathsf{d}_{\mathcal{M}'}(\cdot, \cdot)$ complies with the triangle inequality. For the sake of simplicity of exposition, we assume that for any two distinct points p and v in our (finite) input point set P it holds that $\mathsf{p}' \neq v'$ (that is, $\mathsf{d}_{\mathcal{M}'}(\mathsf{p}', v') \neq 0$). This can be easily guaranteed by introducing symbolic perturbations.

Lemma 3.1.3 *The following holds: (A) For any two points $x, y \in \mathcal{M}$, we have $\mathsf{d}_{\mathcal{M}'}(x', y') = \mathsf{d}_{\mathcal{X}}(x, y)$.*

(B) For any point $x \in \mathcal{M}$ and $y \in \mathcal{X}$, we have $\mathsf{d}_{\mathcal{X}}(x, y) \leq \mathsf{d}_{\mathcal{M}'}(x', y') \leq 3\mathsf{d}_{\mathcal{X}}(x, y)$.

(C) The space $\mathcal{M}'$ has doubling dimension at most $2\tau + 2$, where τ is the doubling dimension of $\mathcal{M}$.

Proof: (A) Clearly, for $x, y \in \mathcal{M}$, we have $x' = (x, 0)$ and $y' = (y, 0)$. As such, $\mathsf{d}_{\mathcal{M}'}(x', y') = \mathsf{d}_{\mathcal{X}}(x, y) + |0 - 0| = \mathsf{d}_{\mathcal{X}}(x, y)$.

(B) Let $x \in \mathcal{M}$ and $y \in \mathcal{X}$. We have $x' = (x, 0)$ and $y' = \Big(\alpha(y), \mathsf{d}_{\mathcal{X}}\big(y, \alpha(y)\big) \Big)$. As such,

$$\mathsf{d}_{\mathcal{M}'}(x', y') = \mathsf{d}_{\mathcal{X}}(\alpha(x), \alpha(y)) + |0 - \mathsf{h}(y)| = \mathsf{d}_{\mathcal{X}}(x, \alpha(y)) + \mathsf{d}_{\mathcal{X}}(\alpha(y), y) \geq \mathsf{d}_{\mathcal{X}}(x, y) \,,$$

by the triangle inequality. On the other hand, because $\mathsf{d}_{\mathcal{X}}(y, \alpha(y)) = \mathsf{d}_{\mathcal{X}}(y, \mathcal{M}) \leq \mathsf{d}_{\mathcal{X}}(x, y)$, we have

$$\begin{aligned}
\mathsf{d}_{\mathcal{M}'}(x', y') &= \mathsf{d}_{\mathcal{X}}\Big(\alpha(x), \alpha(y)\Big) + |\mathsf{h}(x) - \mathsf{h}(y)| = \mathsf{d}_{\mathcal{X}}\Big(x, \alpha(y)\Big) + \mathsf{h}(y) \\
&= \mathsf{d}_{\mathcal{X}}(x, \alpha(y)) + \mathsf{d}_{\mathcal{X}}(y, \alpha(y)) \leq \Big(\mathsf{d}_{\mathcal{X}}(x, y) + \mathsf{d}_{\mathcal{X}}(y, \alpha(y))\Big) + \mathsf{d}_{\mathcal{X}}(y, \alpha(y)) \\
&= \mathsf{d}_{\mathcal{X}}(x, y) + 2\mathsf{d}_{\mathcal{X}}(y, \alpha(y)) \leq 3\mathsf{d}_{\mathcal{X}}(x, y)\,,
\end{aligned}$$

by the triangle inequality.

(C) Consider a point $(\mathsf{p}, \psi) \in \mathcal{M} \times \mathbb{R}^+ = \mathcal{M}'$ and the ball $\mathsf{b} = \mathsf{ball}_{\mathcal{M}'}((\mathsf{p}, \psi), r) \subseteq \mathcal{M}'$ of radius r centered at (p, ψ). Consider the projection of b into $\mathcal{M}$; that is $\mathsf{P}_{\mathcal{M}} = \Big\{v \;\Big|\; (v, h) \in \mathsf{b}\Big\}$. Similarly, let $\mathsf{P}_{\mathbb{R}} = \Big\{h \;\Big|\; (v, h) \in \mathsf{b}\Big\}$.

Clearly, $\mathsf{ball}_{\mathcal{M}'}\big((\mathsf{p}, \psi), r\big) \subseteq \mathsf{P}_{\mathcal{M}} \times \mathsf{P}_{\mathbb{R}}$, and $\mathsf{P}_{\mathcal{M}}$ is contained in $\mathsf{ball}_{\mathcal{M}}(\mathsf{p}, r) = \mathsf{ball}_{\mathcal{X}}(\mathsf{p}, r) \cap \mathcal{M}$. Since the doubling dimension of $\mathcal{M}$ is τ, this ball can be covered by $2^{2\tau}$ balls of the form $\mathsf{ball}_{\mathcal{M}}(\mathsf{p}_i, r/4)$ with centers $\mathsf{p}_i \in \mathcal{M}$.

Also since $\mathsf{P}_{\mathbb{R}} \subseteq \mathbb{R}$ is contained in the interval $\big[\psi - r, \psi + r\big]$ having length $2r$, it can be covered by at most 4 intervals $I_1, \ldots, I_4$ of length $r/2$ each, centered at values $x_1, \ldots, x_4$, respectively. (Intuitively, each of the intervals I_j, is a "ball" of radius $r/4$.) Then,

$$\begin{aligned}
\mathsf{ball}_{\mathcal{M}'}\Big((\mathsf{p}, \psi), r\Big) \quad &\subseteq \quad \mathsf{P}_{\mathcal{M}} \times \mathsf{P}_{\mathbb{R}} \quad \subseteq \quad \left(\bigcup_i \mathsf{ball}_{\mathcal{M}}(\mathsf{p}_i, r/4)\right) \times \left(\bigcup_{j=1}^{4} I_j\right) \\
&\subseteq \quad \bigcup_{j=1}^{4} \bigcup_i (\mathsf{ball}_{\mathcal{M}}(\mathsf{p}_i, r/4) \times I_j) \quad \subseteq \quad \bigcup_{j=1}^{4} \bigcup_i \mathsf{ball}_{\mathcal{M}'}\Big((\mathsf{p}_i, x_j), r/2\Big)\,,
\end{aligned}$$

since the set $\mathsf{ball}_{\mathcal{M}}\big(\mathsf{p}_i, r/4\big) \times I_j$ is contained in $\mathsf{ball}_{\mathcal{M}'}\big((\mathsf{p}_i, x_j), r/2\big)$. We conclude that $\mathsf{ball}_{\mathcal{M}'}((\mathsf{p}, \psi), r)$ can be covered using at most $2^{2\tau+2}$ balls of half the radius. ■

3.2 A Constant Factor ANN Algorithm

In this section we present a 6-ANN algorithm. We refine this to a $(1+\varepsilon)$-ANN in the next section.

Preprocessing. In the preprocessing stage, we map the points of P into the metric space $\mathcal{M}'$ of Lemma 3.1.3. Build a net-tree for the point set $\mathsf{P}' = \Big\{\mathsf{p}' \;\Big|\; \mathsf{p} \in \mathsf{P}\Big\}$ in $\mathcal{M}'$ and preprocess it for ANN queries using the net-tree data-structure (augmented for nearest-neighbor queries) of Har-Peled and Mendel [HM06]. Let $\mathcal{D}$ denote the resulting data-structure.

Answering a query. Given $\mathsf{q} \in \mathcal{M}$, we compute a 2-ANN to $\mathsf{q}' \in \mathcal{M}'$ using $\mathcal{D}$. Let this be the point y'. Return $\mathsf{d}_{\mathcal{X}}(\mathsf{q}, y)$, where y is the original point in P corresponding to y'.

Correctness. Let n_{q} be the nearest neighbor of q in P and let y be the point returned. As $\mathsf{q} \in \mathcal{M}$ we have by Lemma 3.1.3 (B) that $\mathsf{d}_{\mathcal{X}}(\mathsf{q}, y) \leq \mathsf{d}_{\mathcal{M}'}(\mathsf{q}', y')$ and $\mathsf{d}_{\mathcal{M}'}\left(\mathsf{q}', \mathsf{n}'_{\mathsf{q}}\right) \leq 3\mathsf{d}_{\mathcal{X}}\left(\mathsf{q}, \mathsf{n}_{\mathsf{q}}\right)$. As y' is a 2-ANN for q' it follows that,

$$\mathsf{d}_{\mathcal{X}}(\mathsf{q}, y) \leq \mathsf{d}_{\mathcal{M}'}(\mathsf{q}', y') \leq 2\mathsf{d}_{\mathcal{M}'}\left(\mathsf{q}', \mathsf{n}'_{\mathsf{q}}\right) \leq 6\mathsf{d}_{\mathcal{X}}\left(\mathsf{q}, \mathsf{n}_{\mathsf{q}}\right).$$

We thus proved the following.

Lemma 3.2.1 *Given a set $\mathsf{P} \subseteq \mathcal{X}$ of n points and a subspace $\mathcal{M}$ of doubling dimension τ, one can build a data-structure in $2^{O(\tau)} n \log n$ expected time, such that given a query point $\mathsf{q} \in \mathcal{M}$, one can return a 6-ANN to q in P in $2^{O(\tau)} \log n$ query time. The space used by this data-structure is $2^{O(\tau)} n$.*

Proof: Since the doubling dimension of $\mathcal{M}'$ is at most $2\tau + 2$, building the net-tree and preprocessing it for ANN queries takes $2^{O(\tau)} n \log n$ expected time, and the space used is $2^{O(\tau)} n$ [HM06]. The 2-ANN query for a point q takes time $2^{O(\tau)} \log n$. ■

3.3 Answering $(1+\varepsilon)$-ANN

Once we have a constant factor approximation to the nearest-neighbor in P it is not too hard to boost it into $(1+\varepsilon)$-ANN. To this end we need to understand what the net-tree [HM06] provides us with. See Har-Peled and Mendel [HM06] (see also Section 2.9) for a precise definition of the net-tree. Roughly speaking, the nodes at a given level l, define an γ^l-net for Q. This means that one can compute an r-net for any desired r by looking at nodes whose levels define the right resolution. Thus r-nets derived from the net-tree have a corresponding set of nodes in the net-tree. Suppose one needs to find an r-net for the points of Q inside a ball $\mathsf{ball}_{\mathcal{M}}(\mathsf{p}, R)$. One computes an ANN $y \in \mathsf{Q}$ of the center p. This determines a leaf node l of the net-tree. One then seeks out a vertex v of the net-tree on the l to root path, such that $l \in \mathsf{Q}_v$ and the v associated ball radius is roughly R. By adding appropriate pointers, one can perform this hopping up the tree in logarithmic time. Now, exploring the top of the subtree rooted at v, and collecting the representative points of the vertices in that traversal, one can compute an r-net for the points in $\mathsf{Q} \cap \mathsf{ball}_{\mathcal{M}}(\mathsf{p}, R)$. In particular, using the ANN data-structure of Har-Peled and Mendel [HM06] this operation is readily supported.

Lemma 3.3.1 ([HM06]) *Given a net-tree for a set $\mathsf{Q} \subseteq \mathcal{M}$ of n points in a metric space with doubling dimension τ, and given a point $\mathsf{p} \in \mathcal{M}$ and radius $r \leq R$, one can compute an r-net $N \subseteq \mathsf{Q}$ of $\mathsf{Q} \cap \mathsf{ball}_{\mathcal{M}}(\mathsf{p}, R)$, such that the following properties hold:*

(A) *For any point* $v \in \mathsf{Q} \cap \mathsf{ball}_{\mathcal{M}}(\mathsf{p}, R)$ *there exists a point* $u \in N$ *such that* $\mathsf{d}_{\mathcal{M}}(v, u) \leq r$.

(B) $|N| = (R/r)^{O(\tau)}$.

(C) *Each point* $z \in N$ *corresponds to a node* $v(z)$ *in the net-tree. Let* $\mathsf{Q}_{v(z)}$ *denote the subset of points of* Q *stored in the subtree of* $v(z)$. *The union* $\bigcup_{z \in N} \mathsf{Q}_{v(z)}$ *covers* $\mathsf{Q} \cap \mathsf{ball}_{\mathcal{M}}(\mathsf{p}, R)$.

(D) *For any* $z \in N$, *the diameter of the point set* $\mathsf{Q}_{v(z)}$ *is bounded by* r.

(E) *The time to compute* N *is* $2^{O(\tau)} \log n + O(|N|)$.

Construction. For every point $\mathsf{p} \in \mathsf{P}$ we compute an $r(\mathsf{p})$-net $U(\mathsf{p})$ for $\mathsf{ball}_{\mathcal{M}}(\alpha(\mathsf{p}), R(\mathsf{p}))$, where $r(\mathsf{p}) = \varepsilon \mathsf{h}(\mathsf{p}) / (20 c_1)$ and $R(\mathsf{p}) = c_1 \mathsf{h}(\mathsf{p}) / \varepsilon$. Here c_1 is some sufficiently large constant. This net is computed using the algorithm **compNet**, see Subsection 3.1. This takes $1/\varepsilon^{O(\tau)}$ time to compute for each point of P.

For each point u of the net $U(\mathsf{p}) \subseteq \mathcal{M}$ store the original point p it arises from, and the distance to the original point p. We will refer to $\mathsf{s}(u) = \mathsf{d}_{\mathcal{X}}(u, \mathsf{p})$ as the ***reach*** of u.

Let $\mathsf{Q} \subseteq \mathcal{M}$ be union of all these nets. Clearly, we have that $|\mathsf{Q}| = n/\varepsilon^{O(\tau)}$. Build a net-tree $\mathcal{T}$ for the points of Q. We compute in a bottom-up fashion for each node v of the net-tree $\mathcal{T}$ the point with the smallest reach stored in Q_v.

Answering a query. Given a query point $\mathsf{q} \in \mathcal{M}$, compute using the algorithm of Lemma 3.2.1 a 6-ANN to q in P. Let Δ be the distance from q to this ANN. Let $R = 20\Delta$, and $r' = \varepsilon\Delta/20$. Using $\mathcal{T}$ and Lemma 3.3.1, compute an r'-net N of $\mathsf{ball}_{\mathcal{M}}(\mathsf{q}, R) \cap \mathsf{Q}$.

Next, for each point $\mathsf{p} \in N$ consider its corresponding node $v(\mathsf{p}) \in \mathcal{T}$. Each such node stores a point of minimum reach in $\mathsf{Q}_{v(\mathsf{p})}$. We compute the distance to each such minimum-reach point and return the nearest-neighbor found as the ANN.

Theorem 3.3.2 *Given a set* $\mathsf{P} \subseteq \mathcal{X}$ *of* n *points and a subspace* $\mathcal{M}$ *of doubling dimension* τ, *and a parameter* $\varepsilon > 0$, *one can build a data-structure in* $n\varepsilon^{-O(\tau)} \log n$ *expected time, such that given a query point* $\mathsf{q} \in \mathcal{M}$, *one can return a* $(1+\varepsilon)$-*ANN to* q *in* P. *The query time is* $2^{O(\tau)} \log n + \varepsilon^{-O(\tau)}$. *This data-structure uses* $n\varepsilon^{-O(\tau)}$ *space.*

Proof: We only need to prove the bound on the quality of the approximation. Consider the nearest-neighbor n_{q} to q in P.

(A) If there is a point $z \in U(\mathsf{n}_{\mathsf{q}}) \subseteq \mathsf{Q}$ within distance r' from q then there is a net point u of N that contains z in its subtree of $\mathcal{T}$. Let w_y be the point of minimum reach in $\mathsf{Q}_{v(u)}$, and let $y \in \mathsf{P}$ be the corresponding original point. Now, we have

$$\mathsf{d}_{\mathcal{X}}(\mathsf{q}, y) \leq \mathsf{d}_{\mathcal{X}}(\mathsf{q}, w_y) + \mathsf{d}_{\mathcal{X}}(w_y, y) \leq \mathsf{d}_{\mathcal{X}}(\mathsf{q}, w_y) + \mathsf{d}_{\mathcal{X}}\big(z, \mathsf{n}_{\mathsf{q}}\big)$$

as the point w_y has reach $\mathsf{d}_{\mathcal{X}}(w_y, y)$, w_y is the point of minimal reach among all the points of $\mathsf{Q}_{v(u)}$, $z \in \mathsf{Q}_{v(u)}$, and $\mathsf{d}_{\mathcal{X}}\big(z, \mathsf{n}_{\mathsf{q}}\big)$ is the reach of z and thus an upper bound on $\mathsf{d}_{\mathcal{X}}(w_y, y)$. By the triangle inequality, we have

$$\begin{aligned}\mathsf{d}_{\mathcal{X}}(\mathsf{q}, y) &\leq \mathsf{d}_{\mathcal{X}}(\mathsf{q}, w_y) + \mathsf{d}_{\mathcal{X}}\big(\mathsf{q}, \mathsf{n}_{\mathsf{q}}\big) + \mathsf{d}_{\mathcal{X}}(z, \mathsf{q}) \\ &\leq \Big(\mathsf{d}_{\mathcal{X}}(\mathsf{q}, z) + \mathsf{d}_{\mathcal{X}}(z, w_y)\Big) + \mathsf{d}_{\mathcal{X}}\big(\mathsf{q}, \mathsf{n}_{\mathsf{q}}\big) + \mathsf{d}_{\mathcal{X}}(z, \mathsf{q}) \\ &\leq \mathsf{d}_{\mathcal{X}}\big(\mathsf{q}, \mathsf{n}_{\mathsf{q}}\big) + 3r',\end{aligned}$$

as $z, w_y \in \mathsf{Q}_{v(u)}$, the diameter of $\mathsf{Q}_{v(u)}$ is at most r', and by assumption $\mathsf{d}_{\mathcal{X}}(z, \mathsf{q}) \leq r'$. So we have,

$$\mathsf{d}_{\mathcal{X}}(\mathsf{q}, y) \leq \mathsf{d}_{\mathcal{X}}\big(\mathsf{q}, \mathsf{n}_{\mathsf{q}}\big) + 3\varepsilon\Delta/20 \leq (1+\varepsilon)\mathsf{d}_{\mathcal{X}}\big(\mathsf{q}, \mathsf{n}_{\mathsf{q}}\big).$$

(B) Otherwise, it must be that, $\mathsf{d}_{\mathcal{X}}\big(\mathsf{q}, U(\mathsf{n}_{\mathsf{q}})\big) > r'$. Observe that it must be that $r(\mathsf{n}_{\mathsf{q}}) < r'$ as $\mathsf{h}\big(\mathsf{n}_{\mathsf{q}}\big) \leq \Delta$. It must be therefore that the query point is outside the region covered by the net $U(\mathsf{n}_{\mathsf{q}})$. As such, we have

$$R(\mathsf{n}_{\mathsf{q}}) = \frac{c_1 \mathsf{h}\big(\mathsf{n}_{\mathsf{q}}\big)}{\varepsilon} < \mathsf{d}_{\mathcal{X}}\big(\alpha(\mathsf{n}_{\mathsf{q}}), \mathsf{q}\big) \leq \mathsf{d}_{\mathcal{X}}\big(\mathsf{q}, \mathsf{n}_{\mathsf{q}}\big) + \mathsf{d}_{\mathcal{X}}\big(\mathsf{n}_{\mathsf{q}}, \alpha(\mathsf{n}_{\mathsf{q}})\big) \leq 2\mathsf{d}_{\mathcal{X}}\big(\mathsf{n}_{\mathsf{q}}, \mathsf{q}\big) \leq 2\Delta,$$

which means $\mathsf{h}\big(\mathsf{n}_{\mathsf{q}}\big) \leq 2\varepsilon\Delta/c_1$. Namely, the height of the point n_{q} is insignificant in comparison to its distance from q (and conceptually can be considered to be zero). In particular, consider the net point $u \in N$ that contains in its subtree the point $z \in U(\mathsf{n}_{\mathsf{q}})$ closest to $\alpha(\mathsf{n}_{\mathsf{q}})$, i.e., $\mathsf{d}_{\mathcal{M}}\big(\alpha(\mathsf{n}_{\mathsf{q}}), z\big) \leq r(\mathsf{n}_{\mathsf{q}})$. The point of smallest reach in this subtree provides a $(1+\varepsilon)$-ANN as an easy but tedious argument similar to the one above shows. ■

3.4 Answering $(1+\varepsilon)$-ANN faster

In this section, we extend the approach used in the above construction to get a data-structure which is similar in spirit to an AVD of P on $\mathcal{M}$. Specifically, we spread a set of points $\mathcal{C}$ on $\mathcal{M}$, and we associate a point of P with each one of them. Now, answering 2-ANN on $\mathcal{C}$, and returning the point of P associated with this point, results in the desired $(1+\varepsilon)$-ANN.

algBuildANN$(\mathsf{P}, \mathcal{M})$.

$\mathsf{P}' = \left\{ x' \;\middle|\; x \in \mathsf{P} \right\} \subseteq \mathcal{M}'$

Compute a 8-WSPD $\mathcal{W} = \{\{A'_1, B'_1\}, \ldots, \{A'_s, B'_s\}\}$ of P'

for $\{A'_i, B'_i\} \in \mathcal{W}$ **do**

  Choose points $a'_i \in A'_i$ and $b'_i \in B'_i$.

  $t_i = \mathsf{d}_{\mathcal{M}'}(a'_i, b'_i), \quad T_i = t_i + \mathsf{h}_{\max}(A'_i) + \mathsf{h}_{\max}(B'_i)$

  $R_i = c_2 T_i/\varepsilon, \quad r_i = \varepsilon T_i / c_2$

  $N_i = \textbf{compNet}(\alpha(a_i), R_i, r_i) \cup \textbf{compNet}(\alpha(b_i), R_i, r_i).$

$\mathcal{C} = N_1 \cup \ldots \cup N_s$

$\mathcal{N}_{\mathcal{C}} \leftarrow$ Net-tree for $\mathcal{C}$ [HM06]

for $\mathsf{p} \in \mathcal{C}$ **do**

  Compute $\mathsf{nn}(\mathsf{p}, \mathsf{P})$ and store it with p

Figure 3.3: Preprocessing the subspace $\mathcal{M}$ to answer $(1+\varepsilon)$-ANN queries on P. Here c_2 is a sufficiently large constant.

algANN ($\mathsf{q} \in \mathcal{M}$)

  $\mathsf{p} \leftarrow$ 2-ANN of q in $\mathcal{C}$

  (Use net-tree $\mathcal{N}_{\mathcal{C}}$ [HM06] to compute p.)

  $y \leftarrow$ the point in P associated with p.

  return y

Figure 3.4: Computing a $(1 + O(\varepsilon))$-ANN in P for a query point $\mathsf{q} \in \mathcal{M}$.

3.4.1 The construction

For a set $Z' \subseteq \mathsf{P}'$ let

$$\mathsf{h}_{\max}(Z') = \max_{(\mathsf{p},h) \in Z'} h.$$

The preprocessing stage is presented in Figure 3.3, and the algorithm for finding the $(1+\varepsilon)$-ANN for a given query is presented in Figure 3.4.

3.4.2 Analysis

Suppose the data-structure returned y and the actual nearest neighbor of q is n_{q}. If $y = \mathsf{n}_{\mathsf{q}}$ then the algorithm returned the exact nearest-neighbor to q and we are done. Otherwise, by our general position assumption, we can assume that $y' \neq \mathsf{n}'_{\mathsf{q}}$. Note that there is a WSPD pair $\{A', B'\} \in \mathcal{W}$ that separates y' from n'_{q} in $\mathcal{M}'$; namely, $y' \in A'$ and $\mathsf{n}'_{\mathsf{q}} \in B'$. Let

$$t = \mathsf{d}_{\mathcal{M}'}(a', b'),$$

where a' and b' are the representative points of A' and B', respectively. Let a and b be the points of P corresponding to a' and b', respectively. Now, let

$$T = \mathsf{h}_{\max}(A') + \mathsf{h}_{\max}(B') + t, \qquad R = c_2 T/\varepsilon \qquad \text{and} \qquad r = \varepsilon T/c_2.$$

Observation 3.4.1 *By the definition of a 8-WSPD and the triangle inequality, for any $x' \in A'$ and $y' \in B'$, we have that $\mathsf{d}_{\mathcal{M}'}(x', y') \leq \operatorname{diam}(A') + \operatorname{diam}(B') + \mathsf{d}_{\mathcal{M}'}(a', b') \leq (5/4)\, t$.*

We study the two possible cases, $\mathsf{q} \notin \mathsf{ball}_{\mathcal{M}}(\alpha(a), R) \cup \mathsf{ball}_{\mathcal{M}}(\alpha(b), R)$ (Lemma 3.4.2) and $\mathsf{q} \in \mathsf{ball}_{\mathcal{M}}(\alpha(a), R) \cup \mathsf{ball}_{\mathcal{M}}(\alpha(b), R)$ (Lemma 3.4.3).

Lemma 3.4.2 *If $\mathsf{q} \notin \mathsf{ball}_{\mathcal{M}}(\alpha(a), R) \cup \mathsf{ball}_{\mathcal{M}}(\alpha(b), R)$ then the algorithm from Figure 3.4 returns a $(1+\varepsilon)$-ANN in P to the query point q (assuming c_2 is sufficient large). Restated informally – if q is far from both y and n_{q} (compared to the distance between them) then the ANN computed is correct.*

Proof: We have $\mathsf{d}_{\mathcal{X}}\big(\alpha(\mathsf{n}_{\mathsf{q}}), \alpha(y)\big) \leq \mathsf{d}_{\mathcal{M}'}\big(\mathsf{n}'_{\mathsf{q}}, y'\big) \leq 5/4t$ by Observation 3.4.1. So, by the triangle inequality, we have $\mathsf{d}_{\mathcal{X}}\big(\mathsf{n}_{\mathsf{q}}, y\big) \leq \mathsf{h}\big(\mathsf{n}_{\mathsf{q}}\big) + \mathsf{d}_{\mathcal{X}}\big(\alpha(\mathsf{n}_{\mathsf{q}}), \alpha(y)\big) + \mathsf{h}(y) \leq \mathsf{h}_{\max}(A') + (5/4)t + \mathsf{h}_{\max}(B') \leq (5/4)T$.

Since $\mathsf{n}'_{\mathsf{q}}, b' \in B'$, we have $\mathsf{d}_{\mathcal{X}}\big(\alpha(\mathsf{n}_{\mathsf{q}}), \alpha(b)\big) \leq \mathsf{d}_{\mathcal{M}'}\big(\mathsf{n}'_{\mathsf{q}}, b'\big) \leq \operatorname{diam}(B') \leq t/8 \leq T/8$. Therefore,

$$\begin{aligned}\mathsf{d}_{\mathcal{X}}\big(\mathsf{q}, \alpha(\mathsf{n}_{\mathsf{q}})\big) &\geq \mathsf{d}_{\mathcal{X}}(\mathsf{q}, \alpha(b)) - \mathsf{d}_{\mathcal{X}}\big(\alpha(\mathsf{n}_{\mathsf{q}}), \alpha(b)\big) \geq R - \operatorname{diam}(B') = c_2 \frac{T}{\varepsilon} - \operatorname{diam}(B') \\ &\geq T\left(\frac{c_2}{\varepsilon} - \frac{1}{8}\right) \geq \frac{c_2 T}{2\varepsilon},\end{aligned}$$

assuming $\varepsilon \leq 1$ and $c_2 \geq 1$. Now, $\mathsf{d}_{\mathcal{X}}\big(\mathsf{q}, \mathsf{n}_{\mathsf{q}}\big) \geq \mathsf{d}_{\mathcal{X}}\big(\mathsf{n}_{\mathsf{q}}, \mathcal{M}\big) = \mathsf{d}_{\mathcal{X}}\big(\mathsf{n}_{\mathsf{q}}, \alpha(\mathsf{n}_{\mathsf{q}})\big)$, and thus by the triangle inequality, we have

$$\mathsf{d}_{\mathcal{X}}\big(\mathsf{q}, \mathsf{n}_{\mathsf{q}}\big) \geq \frac{\mathsf{d}_{\mathcal{X}}\big(\mathsf{q}, \mathsf{n}_{\mathsf{q}}\big) + \mathsf{d}_{\mathcal{X}}\big(\mathsf{n}_{\mathsf{q}}, \alpha(\mathsf{n}_{\mathsf{q}})\big)}{2} \geq \frac{\mathsf{d}_{\mathcal{X}}\big(\mathsf{q}, \alpha(\mathsf{n}_{\mathsf{q}})\big)}{2} \geq \frac{c_2 T}{4\varepsilon}.$$

This implies that $\mathsf{d}_{\mathcal{X}}(\mathsf{q}, y) \leq \mathsf{d}_{\mathcal{X}}\big(\mathsf{q}, \mathsf{n}_{\mathsf{q}}\big) + \mathsf{d}_{\mathcal{X}}\big(\mathsf{n}_{\mathsf{q}}, y\big) \leq \mathsf{d}_{\mathcal{X}}\big(\mathsf{q}, \mathsf{n}_{\mathsf{q}}\big) + (5/4)T \leq (1+\varepsilon)\mathsf{d}_{\mathcal{X}}\big(\mathsf{q}, \mathsf{n}_{\mathsf{q}}\big)$, assuming $c_2 \geq 5$. ■

Lemma 3.4.3 *If $\mathsf{q} \in \mathsf{ball}_{\mathcal{M}}(\alpha(a), R) \cup \mathsf{ball}_{\mathcal{M}}(\alpha(b), R)$ then the algorithm returns a $(1+\varepsilon)$-ANN in P to the query point q.*

Proof: Since the algorithm covered the set $\mathsf{ball}_{\mathcal{M}}(\alpha(a), R) \cup \mathsf{ball}_{\mathcal{M}}(\alpha(b), R)$ with a net of radius $r = \varepsilon T/c_2$, it follows that $\mathsf{d}_{\mathcal{X}}(\mathsf{q}, \mathcal{C}) \leq r$. Let $\overline{\mathsf{c}}$ be the point in the 2-ANN search to q in $\mathcal{N}_{\mathcal{C}}$. We have $\mathsf{d}_{\mathcal{X}}(\mathsf{q}, \overline{\mathsf{c}}) \leq 2r$. Now, the algorithm returned the nearest neighbor to $\overline{\mathsf{c}}$ as the ANN; that is, y is the nearest neighbor of $\overline{\mathsf{c}}$ in P.

Now,

$$\begin{aligned}\mathsf{d}_{\mathcal{X}}(\mathsf{q}, y) &\leq \mathsf{d}_{\mathcal{X}}(\overline{\mathsf{c}}, y) + \mathsf{d}_{\mathcal{X}}(\mathsf{q}, \overline{\mathsf{c}}) \leq \mathsf{d}_{\mathcal{X}}(\overline{\mathsf{c}}, y) + 2r \leq \mathsf{d}_{\mathcal{X}}\big(\overline{\mathsf{c}}, \mathsf{n}_{\mathsf{q}}\big) + 2r \\ &\leq \mathsf{d}_{\mathcal{X}}\big(\mathsf{q}, \mathsf{n}_{\mathsf{q}}\big) + \mathsf{d}_{\mathcal{X}}(\overline{\mathsf{c}}, \mathsf{q}) + 2r \leq \mathsf{d}_{\mathcal{X}}\big(\mathsf{q}, \mathsf{n}_{\mathsf{q}}\big) + 4r = \mathsf{d}_{\mathcal{X}}\big(\mathsf{q}, \mathsf{n}_{\mathsf{q}}\big) + 4\frac{\varepsilon T}{c_2},\end{aligned}$$

by the triangle inequality. Therefore, if $\mathsf{d}_{\mathcal{X}}(\mathsf{q}, y) \geq T/40$ then,

$$\mathsf{d}_{\mathcal{X}}\big(\mathsf{q}, \mathsf{n}_{\mathsf{q}}\big) \geq \mathsf{d}_{\mathcal{X}}(\mathsf{q}, y) - 4\frac{\varepsilon T}{c_2} \geq (1 - \varepsilon/2)\mathsf{d}_{\mathcal{X}}(\mathsf{q}, y)\,,$$

assuming $c_2 \geq 320$. Since $1/(1 - \varepsilon/2) \leq 1 + \varepsilon$, we have that $\mathsf{d}_{\mathcal{X}}(\mathsf{q}, y) \leq (1 + \varepsilon)\mathsf{d}_{\mathcal{X}}\big(\mathsf{q}, \mathsf{n}_{\mathsf{q}}\big)$.

Similarly, if $\mathsf{d}_{\mathcal{X}}\big(\mathsf{q}, \mathsf{n}_{\mathsf{q}}\big) \geq T/40$ then,

$$\mathsf{d}_{\mathcal{X}}(\mathsf{q}, y) \leq \mathsf{d}_{\mathcal{X}}\big(\mathsf{q}, \mathsf{n}_{\mathsf{q}}\big) + 4\frac{\varepsilon T}{c_2} \leq (1 + \varepsilon)\mathsf{d}_{\mathcal{X}}\big(\mathsf{q}, \mathsf{n}_{\mathsf{q}}\big)\,,$$

assuming $c_2 \geq 160$.

We prove by contradiction that the case $\mathsf{d}_{\mathcal{X}}\big(\mathsf{q}, \mathsf{n}_{\mathsf{q}}\big) \leq T/40$ and $\mathsf{d}_{\mathcal{X}}(\mathsf{q}, y) \leq T/40$ is impossible. That is, intuitively, T is roughly the distance between n_{q} to y, and there is no point that can be close to both n_{q} and y. Indeed, under those assumptions, $\mathsf{h}\big(\mathsf{n}_{\mathsf{q}}\big) \leq \mathsf{d}_{\mathcal{X}}\big(\mathsf{q}, \mathsf{n}_{\mathsf{q}}\big) \leq T/40$ and $\mathsf{h}(y) \leq \mathsf{d}_{\mathcal{X}}(\mathsf{q}, y) \leq T/40$. Observe that

$$\mathsf{h}_{\max}(A') \leq \mathsf{h}(y) + \operatorname{diam}(A') \leq T/40 + \frac{t}{8} \leq \frac{3T}{20}.$$

and similarly $\mathsf{h}_{\max}(B') \leq 3T/20$. This implies that

$$\begin{aligned}(3/4)t &= t\left(1 - \frac{1}{8} - \frac{1}{8}\right) \\ &\leq \mathsf{d}_{\mathcal{M}'}(a', b') - \operatorname{diam}(A') - \operatorname{diam}(B') \leq \mathsf{d}_{\mathcal{M}'}\big(\mathsf{n}'_{\mathsf{q}}, y'\big) \\ &= \big|\mathsf{h}\big(\mathsf{n}_{\mathsf{q}}\big) - \mathsf{h}(y)\big| + \mathsf{d}_{\mathcal{X}}\big(\alpha(\mathsf{n}_{\mathsf{q}}), \alpha(y)\big) \leq T/40 + \mathsf{d}_{\mathcal{X}}\big(\alpha(\mathsf{n}_{\mathsf{q}}), \mathsf{n}_{\mathsf{q}}\big) + \mathsf{d}_{\mathcal{X}}\big(\mathsf{n}_{\mathsf{q}}, y\big) + \mathsf{d}_{\mathcal{X}}(y, \alpha(y)) \\ &\leq T/40 + \mathsf{h}\big(\mathsf{n}_{\mathsf{q}}\big) + (\mathsf{d}_{\mathcal{X}}\big(\mathsf{n}_{\mathsf{q}}, \mathsf{q}\big) + \mathsf{d}_{\mathcal{X}}(\mathsf{q}, y)) + \mathsf{h}(y) \\ &\leq T/40 + 3T/20 + T/40 + T/40 + 3T/20 \leq 3T/8\end{aligned}$$

This implies that $t \leq T/2$ and thus $T = t + \mathsf{h}_{\max}(A') + \mathsf{h}_{\max}(B') \leq T/2 + 3T/20 + 3T/20 = (4/5)T$. This implies that $T \leq 0$. We conclude that $\mathsf{d}_{\mathcal{M}'}(a', b') = t \leq T \leq 0$. This implies that $a' = b'$, which is impossible, as no two points of P get mapped to the same point in $\mathcal{M}'$. (And of course, no point can appear in both sides of a pair in the WSPD.) ∎

The preprocessing time of the above algorithm is dominated by the task of computing for each point of $\mathcal{C}$ its nearest neighbor in P. Observe that the algorithm would work even if we only use $(1 + O(\varepsilon))$-ANN. Using Theorem 3.3.2 to answer these queries, we get the following result.

Theorem 3.4.4 *Given a set of $\mathsf{P} \subseteq \mathcal{X}$ of n points, and a subspace $\mathcal{M}$ of doubling dimension τ, one can construct a data-structure requiring space $n\varepsilon^{-O(\tau)}$, such that given a query point $\mathsf{q} \in \mathcal{M}$ one can find a $(1+\varepsilon)$-ANN to q in P. The query time is $2^{O(\tau)} \log(n/\varepsilon)$, and the preprocessing time to build this data-structure is $n\varepsilon^{-O(\tau)} \log n$.*

3.5 Online ANN

The algorithms of Section 3.3 and Section 3.4 require that the subspace of the query points is known, in that we can compute the closest point $\alpha(\mathsf{p})$ on $\mathcal{M}$ given a $\mathsf{p} \in \mathcal{X}$, and that we can find a net for a ball on $\mathcal{M}$ using **compNet**, see Subsection 3.1. In this section we show that if we are able to efficiently answer membership queries in regions that are the difference of two balls, then we do not need such explicit access to $\mathcal{M}$. We construct an AVD on $\mathcal{M}$ in an online manner as the query points arrive. When a new query point arrives, we test for membership among the existing regions of the AVD. If a region contains the point we immediately output its associated ANN that is already stored with the region. Otherwise we use an appropriate algorithm to find a nearest neighbor for the query point and add a new region to the AVD.

Here we present our algorithm to compute the AVD in this online setting and prove that when the query points come from a subspace of low doubling dimension, the number of regions created is linear.

3.5.1 Online AVD Construction and ANN Queries

The algorithm **algBuildAVD**$(\mathsf{P}, \mathcal{R}, \mathsf{q})$ is presented in Figure 3.5. The algorithm maintains a set of regions $\mathcal{R}$ that represent the partially constructed AVD. Given a query point q it returns an ANN from P and if needed adds a region C_{q} to $\mathcal{R}$.

Remark 3.5.1 *In order to compute the $(1 + \varepsilon/10)$-ANN y_1 of q in P and the $(1 + \varepsilon/10)$-ANN of q in $\mathsf{P} \setminus \mathsf{ball}_{\mathcal{X}}(y_1, \varepsilon r_1/3)$ (see Figure 3.5) one can use any ANN algorithm or simple brute-force nearest-neighbor search.*

The quantity D' is a 2-approximation to the diameter D of P, and can be precomputed in $O(n)$ time. Let p be an arbitrary fixed point of P.

The regions created by the algorithm in Figure 3.5 are the difference of two balls. An example region when the balls $\mathsf{ball}_{\mathcal{X}}(\mathsf{q}, \varepsilon r_2/5)$ and $\mathsf{ball}_{\mathcal{X}}(y_1, 5\rho_1/4\varepsilon)$ intersect, is shown in Figure 3.6. The intuition as to

algBuildAVD$(\mathsf{P}, \mathcal{R}, \mathsf{q})$.

// p is an arbitrary fixed point in P.
// D′ is a 2-approximation to diam(P).
if $\mathsf{d}_{\mathcal{X}}(\mathsf{q}, \mathsf{p}) \geq 4\mathsf{D}'/\varepsilon$ **then return** p.
if $\exists C \in \mathcal{R}$ with $\mathsf{q} \in C$ **then**
    return the point associated with C.
Compute $(1+\varepsilon/10)$-ANN y_1 of q in P.// See Remark 3.5.1.
$r_1 \leftarrow \mathsf{d}_{\mathcal{X}}(\mathsf{q}, y_1)$.
if there is no point in $\mathsf{P} \setminus \mathsf{ball}_{\mathcal{X}}(y_1, \varepsilon r_1/3)$ **then**
    $C_{\mathsf{q}} \leftarrow \mathsf{ball}_{\mathcal{X}}(\mathsf{q}, \mathsf{D}'/4)$.
else
    $f_1 \leftarrow$ furthest point from y_1 in $\mathsf{P} \cap \mathsf{ball}_{\mathcal{X}}(y_1, \varepsilon r_1/3)$.
    $\rho_1 \leftarrow \mathsf{d}_{\mathcal{X}}(y_1, f_1)$. // $\rho_1 \leq \varepsilon r_1/3$.
    $y_2 \leftarrow (1+\varepsilon/10)$-ANN of q in $\mathsf{P} \setminus \mathsf{ball}_{\mathcal{X}}(y_1, \varepsilon r_1/3)$. // See Remark 3.5.1.
    $r_2 \leftarrow \mathsf{d}_{\mathcal{X}}(\mathsf{q}, y_2)$.
    $C_{\mathsf{q}} \leftarrow \mathsf{ball}_{\mathcal{X}}(\mathsf{q}, \varepsilon r_2/5) \setminus \mathsf{ball}_{\mathcal{X}}(y_1, 5\rho_1/4\varepsilon)$.
Associate y_1 with C_{q}.
$\mathcal{R} \leftarrow \mathcal{R} \cup C_{\mathsf{q}}$.
return y_1 as the ANN for q.

Figure 3.5: Answering $(1+\varepsilon)$-ANN and constructing AVD.

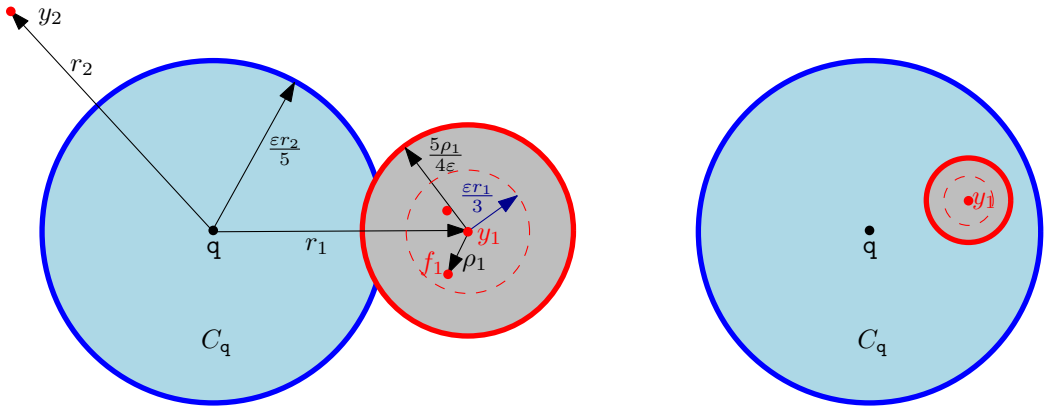


Figure 3.6: Examples of a computed AVD region C_{q}.

why y_1 is a valid ANN inside this region is as follows. Since the distance of q to y_1 is r_1, the points inside $\mathsf{ball}_{\mathcal{X}}(y_1, \varepsilon r_1/3)$ are all roughly the same distance from q as q is far enough from y_1. The next distance of interest, $\mathsf{d}_{\mathcal{X}}(\mathsf{q}, y_2) = r_2$, is the distance to a ANN of points outside this ball. As long as we are inside $\mathsf{ball}_{\mathcal{X}}(\mathsf{q}, \varepsilon r_2/5)$ and far enough from y_1, i.e., $\mathsf{d}_{\mathcal{X}}(\mathsf{q}, y_1) > 5\rho_1/4\varepsilon$, the points outside $\mathsf{ball}_{\mathcal{X}}(y_1, \varepsilon r_1/3)$ are too far and cannot be a $(1+\varepsilon)$-ANN. But if we get too close to y_1 we can no longer be certain that y_1 is a valid $(1+\varepsilon)$-ANN, as it is no more true that distances to points inside $\mathsf{ball}_{\mathcal{X}}(y_1, \varepsilon r_1/3)$ look all roughly the same. In other words, there may be points much closer than y_1, when we are close enough to y_1. Thus in a small enough neighborhood around y_1 we need to zoom in and possibly create a new region.

3.5.2 Correctness

Lemma 3.5.2 *If* $\mathsf{d}_{\mathcal{X}}(\mathsf{q},\mathsf{p}) \geq 2\mathsf{D}' + 2\mathsf{D}'/\varepsilon$ *then* p *is a valid* $(1+\varepsilon)$*-ANN.*

Proof: Since D' is a 2-approximation to the diameter of P, so $2\mathsf{D}' \geq \mathsf{D} = \mathrm{diam}(\mathsf{P})$. This means $\mathsf{d}_{\mathcal{X}}(\mathsf{q},\mathsf{p}) \geq \mathsf{D} + \mathsf{D}/\varepsilon$. Let $\mathsf{n}_{\mathsf{q}} \in \mathsf{P}$ be the closest point to q. By the triangle inequality,

$$\mathsf{D} + \mathsf{D}/\varepsilon \leq \mathsf{d}_{\mathcal{X}}(\mathsf{q},\mathsf{p}) \leq \mathsf{d}_{\mathcal{X}}\big(\mathsf{q},\mathsf{n}_{\mathsf{q}}\big) + \mathsf{d}_{\mathcal{X}}\big(\mathsf{n}_{\mathsf{q}},\mathsf{p}\big) \leq \mathsf{d}_{\mathcal{X}}\big(\mathsf{q},\mathsf{n}_{\mathsf{q}}\big) + \mathsf{D}.$$

As such $\mathsf{D} \leq \varepsilon \mathsf{d}_{\mathcal{X}}\big(\mathsf{q},\mathsf{n}_{\mathsf{q}}\big)$. We conclude $\mathsf{d}_{\mathcal{X}}(\mathsf{q},\mathsf{p}) \leq \mathsf{d}_{\mathcal{X}}\big(\mathsf{q},\mathsf{n}_{\mathsf{q}}\big) + \mathsf{d}_{\mathcal{X}}\big(\mathsf{n}_{\mathsf{q}},\mathsf{p}\big) \leq (1+\varepsilon)\mathsf{d}_{\mathcal{X}}\big(\mathsf{q},\mathsf{n}_{\mathsf{q}}\big)$. ■

Lemma 3.5.3 *If there is no region in* $\mathcal{R}$ *containing* q *then the algorithm outputs a valid* $(1+\varepsilon/10)$*-ANN.*

Proof: We output y_1 which is a $(1+\varepsilon/10)$-ANN of q. ■

Lemma 3.5.4 *The* $(1+\varepsilon/10)$*-ANN* y_1 *found in the algorithm is a* $(1+\varepsilon)$*-ANN for any point* $\overline{\mathsf{q}} \in C_{\mathsf{q}}$.

Proof: Let $r_1 = \mathsf{d}_{\mathcal{X}}(\mathsf{q},y_1)$ and $r_2 = \mathsf{d}_{\mathcal{X}}(\mathsf{q},y_2)$. There are two possibilities.

If the region C_{q} is the ball $\mathsf{ball}_{\mathcal{X}}(\mathsf{q},\mathsf{D}'/4)$ constructed when there is no point in $\mathsf{P} \setminus \mathsf{ball}_{\mathcal{X}}(y_1,\varepsilon r_1/3)$, then $\mathsf{D} = \mathrm{diam}(\mathsf{P}) \leq 2\varepsilon r_1/3$. As such,

$$\mathsf{d}_{\mathcal{X}}(\mathsf{q},\mathsf{P}) \geq \frac{\mathsf{d}_{\mathcal{X}}(\mathsf{q},y_1)}{1+\varepsilon/10} = \frac{r_1}{1+\varepsilon/10} \geq \frac{3\mathsf{D}}{2\varepsilon(1+\varepsilon/10)} = \frac{3\mathsf{D}}{2\varepsilon+\varepsilon^2/5} \geq (4/3)\frac{\mathsf{D}}{\varepsilon}.$$

It is not hard to see that in this case, y_1 is a valid $(1+\varepsilon)$-ANN for any point inside $\mathsf{ball}_{\mathcal{X}}(\mathsf{q},\mathsf{D}'/4) \subseteq \mathsf{ball}_{\mathcal{X}}(\mathsf{q},\mathsf{D}/4)$, as $\mathsf{d}_{\mathcal{X}}(\mathsf{ball}_{\mathcal{X}}(\mathsf{q},\mathsf{D}/4),\mathsf{P}) \geq \mathsf{D}/\varepsilon$, for ε sufficiently small.

Otherwise, if the set $\mathsf{P} \setminus \mathsf{ball}_{\mathcal{X}}(y_1,\varepsilon r_1/3)$ is nonempty then let y_2 be a $(1+\varepsilon/10)$-ANN of q in $\mathsf{P} \setminus \mathsf{ball}_{\mathcal{X}}(y_1,\varepsilon r_1/3)$ and let $r_2 = \mathsf{d}_{\mathcal{X}}(\mathsf{q},y_2)$. We break the analysis into two cases.

(i) If $r_2 \leq 2r_1$, then let $\overline{\mathsf{q}}$ be any point in C_{q} and let $\mathsf{n}_{\overline{\mathsf{q}}} \in \mathsf{P}$ be its nearest neighbor. If $\mathsf{n}_{\overline{\mathsf{q}}} = y_1$ there is nothing to show. Otherwise $\mathsf{d}_{\mathcal{X}}(\mathsf{q},\overline{\mathsf{q}}) \leq \varepsilon r_2/5$ and by the triangle inequality we have

$$\begin{aligned}
\mathsf{d}_{\mathcal{X}}\big(\overline{\mathsf{q}},\mathsf{n}_{\overline{\mathsf{q}}}\big) &\geq \mathsf{d}_{\mathcal{X}}\big(\mathsf{q},\mathsf{n}_{\overline{\mathsf{q}}}\big) - \mathsf{d}_{\mathcal{X}}(\mathsf{q},\overline{\mathsf{q}}) \geq \mathsf{d}_{\mathcal{X}}\big(\mathsf{q},\mathsf{n}_{\overline{\mathsf{q}}}\big) - \varepsilon r_2/5 \\
&\geq \mathsf{d}_{\mathcal{X}}(\mathsf{q},y_1)/(1+\varepsilon/10) - \varepsilon 2r_1/5 \\
&\geq (1-\varepsilon/2)r_1,
\end{aligned}$$

as $\mathsf{d}_{\mathcal{X}}\big(\mathsf{q},\mathsf{n}_{\overline{\mathsf{q}}}\big) \geq \mathsf{d}_{\mathcal{X}}(\mathsf{q},\mathsf{P}) \geq \mathsf{d}_{\mathcal{X}}(\mathsf{q},y_1)/(1+\varepsilon/10)$ and $r_1 = \mathsf{d}_{\mathcal{X}}(\mathsf{q},y_1)$. Again, by the triangle inequality

and the above, we have

$$\begin{aligned}\mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, y_1) &\leq \mathsf{d}_{\mathcal{X}}(\mathsf{q}, y_1) + \mathsf{d}_{\mathcal{X}}(\mathsf{q}, \overline{\mathsf{q}}) \leq \mathsf{d}_{\mathcal{X}}(\mathsf{q}, y_1) + 2\varepsilon r_1/5 = (1 + 2\varepsilon/5) r_1 \\ &\leq \frac{1 + 2\varepsilon/5}{1 - \varepsilon/2} \mathsf{d}_{\mathcal{X}}\left(\overline{\mathsf{q}}, \mathsf{n}_{\overline{\mathsf{q}}}\right) \leq (1 + \varepsilon) \mathsf{d}_{\mathcal{X}}\left(\overline{\mathsf{q}}, \mathsf{n}_{\overline{\mathsf{q}}}\right),\end{aligned}$$

for $\varepsilon \leq 1/5$.

(ii) If $r_2 > 2r_1$ then let f_1 be the furthest point from y_1 inside $\mathsf{ball}_{\mathcal{X}}(y_1, \varepsilon r_1/3)$ and let $\rho_1 = \mathsf{d}_{\mathcal{X}}(y_1, f_1)$. Let $\overline{\mathsf{q}}$ be any point in C_{q} and as before let $\mathsf{n}_{\overline{\mathsf{q}}} \in \mathsf{P}$ be its nearest neighbor. We claim that the nearest neighbor of $\overline{\mathsf{q}}$ in P lies in $\mathsf{ball}_{\mathcal{X}}(y_1, \rho_1)$. To see this, let z be any point in $\mathsf{P} \setminus \mathsf{ball}_{\mathcal{X}}(y_1, \rho_1)$. Noting that the distance from q to the closest point in P outside $\mathsf{ball}_{\mathcal{X}}(y_1, \rho_1)$ is at least $r_2/(1 + \varepsilon/10)$ and by triangle inequality we have,

$$\begin{aligned}\mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, z) &\geq \mathsf{d}_{\mathcal{X}}(\mathsf{q}, z) - \mathsf{d}_{\mathcal{X}}(\mathsf{q}, \overline{\mathsf{q}}) \geq \mathsf{d}_{\mathcal{X}}(\mathsf{q}, z) - \varepsilon r_2/5 \\ &\geq r_2/(1 + \varepsilon/10) - \varepsilon r_2/5 > (1 - 3\varepsilon/10) r_2.\end{aligned}$$

On the other hand, as $r_1 = \mathsf{d}_{\mathcal{X}}(\mathsf{q}, y_1)$ and $r_1 < r_2/2$, we have

$$\begin{aligned}\mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, y_1) &\leq \mathsf{d}_{\mathcal{X}}(\mathsf{q}, y_1) + \mathsf{d}_{\mathcal{X}}(\mathsf{q}, \overline{\mathsf{q}}) \leq \mathsf{d}_{\mathcal{X}}(\mathsf{q}, y_1) + \varepsilon r_2/5 = r_1 + \varepsilon r_2/5 < r_2/2 + \varepsilon r_2/5 \\ &\leq (1 - 3\varepsilon/10) r_2 < \mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, z),\end{aligned}$$

by the above. As such, no point in $\mathsf{P} \setminus \mathsf{ball}_{\mathcal{X}}(y_1, \rho_1)$ can be the nearest neighbor of $\overline{\mathsf{q}}$ for $\varepsilon < 1$. As such $\mathsf{n}_{\overline{\mathsf{q}}} \in \mathsf{ball}_{\mathcal{X}}(y_1, \rho_1)$. Now,

$$\mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, y_1) \leq \mathsf{d}_{\mathcal{X}}\left(\overline{\mathsf{q}}, \mathsf{n}_{\overline{\mathsf{q}}}\right) + \mathsf{d}_{\mathcal{X}}\left(\mathsf{n}_{\overline{\mathsf{q}}}, y_1\right) \leq \mathsf{d}_{\mathcal{X}}\left(\overline{\mathsf{q}}, \mathsf{n}_{\overline{\mathsf{q}}}\right) + \rho_1. \tag{3.1}$$

Now $\overline{\mathsf{q}} \in C_{\mathsf{q}} = \mathsf{ball}_{\mathcal{X}}(\mathsf{q}, \varepsilon r_2/5) \setminus \mathsf{ball}_{\mathcal{X}}(y_1, 5\rho_1/4\varepsilon)$, and thus $\mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, y_1) > 5\rho_1/4\varepsilon$. Thus,

$$\mathsf{d}_{\mathcal{X}}\left(\overline{\mathsf{q}}, \mathsf{n}_{\overline{\mathsf{q}}}\right) \geq \mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, y_1) - \mathsf{d}_{\mathcal{X}}\left(y_1, \mathsf{n}_{\overline{\mathsf{q}}}\right) \geq \mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, y_1) - \rho_1 \geq \left(\frac{5}{4\varepsilon} - 1\right) \rho_1. \tag{3.2}$$

Therefore from (3.1) and (3.2), we have

$$\mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, y_1) \leq \mathsf{d}_{\mathcal{X}}\left(\overline{\mathsf{q}}, \mathsf{n}_{\overline{\mathsf{q}}}\right) + \rho_1 \leq \left(1 + \frac{1}{5/4\varepsilon - 1}\right) \mathsf{d}_{\mathcal{X}}\left(\overline{\mathsf{q}}, \mathsf{n}_{\overline{\mathsf{q}}}\right) = \left(1 + \frac{4\varepsilon}{5 - 4\varepsilon}\right) \mathsf{d}_{\mathcal{X}}\left(\overline{\mathsf{q}}, \mathsf{n}_{\overline{\mathsf{q}}}\right)$$
$$\leq (1 + \varepsilon) \mathsf{d}_{\mathcal{X}}\left(\overline{\mathsf{q}}, \mathsf{n}_{\overline{\mathsf{q}}}\right).$$

for $\varepsilon \leq 1/4$.

■

3.5.3 Bounding the number of regions created

The online algorithm presented in Figure 3.5 is valid for any general metric space $\mathcal{X}$, without any restriction on the subspace of query points. However, when the query points are restricted to lie in a subspace $\mathcal{M}$ of low doubling dimension τ, then one can show that at most $n\varepsilon^{-O(\tau)}$ regions are created overall, where $n = |\mathsf{P}|$. There are two types of regions created. The ***outer*** regions are created when $\mathsf{P} \setminus \mathsf{ball}_{\mathcal{X}}(y_1, \varepsilon r_1/3)$ is empty and the ***inner*** regions are created when this condition does not hold. An example of an inner region is shown in Figure 3.6.

Bounding the number of outer regions

First we show that there are at most $\varepsilon^{-O(\tau)}$ outer regions created.

Lemma 3.5.5 *When all the queries to the algorithm come from a subspace of doubling dimension τ, then at most $\varepsilon^{-O(\tau)}$ outer regions are created overall.*

Proof: Any two query points creating distinct outer regions occur at a distance of at least $\mathsf{D}'/4$ from each other. However all of them occur inside a ball of radius $4\mathsf{D}'/\varepsilon$ around p. Thus the spread of the set containing all these query points is bounded by $(4\mathsf{D}'/\varepsilon)\ /\ (\mathsf{D}'/4) = O(1/\varepsilon)$. As such, there are at most $\varepsilon^{-O(\tau)}$ such points. ■

Bounding the number of inner regions

We now consider the inner regions created by the algorithm. Consider the mapped point set P' in the space $\mathcal{M}'$, see Section 3.1.2. Fix a c-WSPD $\left\{\{A'_1, B'_1\}, \ldots, \{A'_s, B'_s\}\right\}$ of P' where c is a constant to be specified shortly and $s = c^{O(\tau)}n$ is the number of pairs. Let $A_i, B_i \subseteq \mathsf{P}$ denote the corresponding "unmapped" points corresponding to A'_i, B'_i, that is, $A_i = \{\mathsf{p} \in \mathsf{P} \mid \mathsf{p}' \in A'_i\}$ and $B_i = \{\mathsf{p} \in \mathsf{P} \mid \mathsf{p}' \in B'_i\}$. If a query point q creates a new inner region we shall assign it to a set $\mathcal{U}_i$ associated with the pair $\{A'_i, B'_i\}$, if the pair of points y'_1, y'_2 of the algorithm satisfy $y'_1 \in A'_i$ and $y'_2 \in B'_i$. Similarly assign q to the set $\mathcal{V}_i$ if $y'_1 \in B'_i$ and $y'_2 \in A'_i$.

Thus, the query points that gave rise to new regions are now associated with pairs of the WSPD. Our analysis bounds the size of the sets $\mathcal{U}_i$ and $\mathcal{V}_i$ associated with a pair $\{A'_i, B'_i\}$, for $i = 1, \ldots, s$, thus bounding the total number of regions created.

Let $\mathcal{U}'_i = \left\{ \mathsf{q}' \;\middle|\; \mathsf{q} \in \mathcal{U}_i \right\} \subseteq \mathcal{M}'$ and $\mathcal{V}'_i = \left\{ \mathsf{q}' \;\middle|\; \mathsf{q} \in \mathcal{V}_i \right\}$, for $i = 1, \ldots, s$. For a pair $\{A'_i, B'_i\}$ of the WSPD we define the numbers $\mathsf{h}_{\max}(A'_i) = \max_{(u,h)\in A'_i} h$. Similarly let $\mathsf{h}_{\max}(B'_i) = \max_{(z,h)\in B'_i} h$. Also, let

$$\mathsf{l}_i = \max_{u'\in A'_i, z'\in B'_i} \mathsf{d}_{\mathcal{X}}(\alpha(u), \alpha(z)) \qquad \text{and} \qquad \mathsf{L}_i = \mathsf{l}_i + \mathsf{h}_{\max}(A'_i) + \mathsf{h}_{\max}(B'_i)\,.$$

The following sequence of lemmas establish our claim. The basic strategy is to show that the set $\mathcal{U}'_i$ has spread $O\big(1/\varepsilon^2\big)$. This holds analogously for $\mathcal{V}'_i$ and so we will only work with $\mathcal{U}'_i$. We will assume that c is a sufficiently large constant and ε is sufficiently small.

Lemma 3.5.6 *For any i, we have* $\operatorname{diam}_{\mathcal{M}'}(A'_i) \leq \mathsf{L}_i/c$ *and* $\operatorname{diam}_{\mathcal{M}'}(B'_i) \leq \mathsf{L}_i/c$.

Proof: By the construction of the WSPD, we have that $\operatorname{diam}_{\mathcal{M}'}(A'_i) \leq \mathsf{d}_{\mathcal{M}'}(A'_i, B'_i)/c$. Moreover, we have

$$\begin{aligned}\mathsf{d}_{\mathcal{M}'}(A'_i, B'_i) &= \min_{\mathsf{p}'\in A'_i, v'\in B'_i} \mathsf{d}_{\mathcal{M}'}(\mathsf{p}', v') = \min_{\mathsf{p}'\in A'_i, v'\in B'_i} \Big(\mathsf{d}_{\mathcal{X}}\Big(\alpha(\mathsf{p}), \alpha(v)\Big) + |\mathsf{h}(\mathsf{p}) - \mathsf{h}(v)|\Big) \\ &\leq \mathsf{l}_i + \min_{\mathsf{p}'\in A'_i, v'\in B'_i} \Big(\,|\mathsf{h}(\mathsf{p})| + |\mathsf{h}(v)|\Big) \leq \mathsf{l}_i + \mathsf{h}_{\max}(A'_i) + \mathsf{h}_{\max}(B'_i) = \mathsf{L}_i.\end{aligned}$$

This implies that $\operatorname{diam}_{\mathcal{M}'}(A'_i) \leq \mathsf{L}_i/c$, and similarly $\operatorname{diam}_{\mathcal{M}'}(B'_i) \leq \mathsf{L}_i/c$. ■

Lemma 3.5.7 *We have* $\mathsf{diam}(\mathcal{U}'_i) = O(\mathsf{L}_i/\varepsilon)$.

Proof: Let q be a (query) point in $\mathcal{U}_i$. By assumption we have $y'_1 \in A'_i$ and $y'_2 \in B'_i$. By the triangle inequality,

$$\begin{aligned}\mathsf{d}_{\mathcal{X}}(y_1, y_2) &\leq \mathsf{d}_{\mathcal{X}}(y_1, \alpha(y_1)) + \mathsf{d}_{\mathcal{X}}(\alpha(y_1), \alpha(y_2)) + \mathsf{d}_{\mathcal{X}}(\alpha(y_2), y_2) \leq \mathsf{h}_{\max}(A'_i) + \mathsf{l}_i + \mathsf{h}_{\max}(B'_i) \\ &\leq \mathsf{L}_i.\end{aligned}$$

On the other hand, since the point y_2 is outside $\mathsf{ball}_{\mathcal{X}}(y_1, \varepsilon r_1/3)$, we have that $\mathsf{d}_{\mathcal{X}}(y_1, y_2) > \varepsilon r_1/3$, where $r_1 = \mathsf{d}_{\mathcal{X}}(\mathsf{q}, y_1)$. This gives us $r_1 < (3/\varepsilon)\mathsf{d}_{\mathcal{X}}(y_1, y_2) < 3\mathsf{L}_i/\varepsilon$. By Lemma 3.1.3, $\mathsf{d}_{\mathcal{M}'}(\mathsf{q}', y'_1) \leq 3\mathsf{d}_{\mathcal{X}}(\mathsf{q}, y_1) = 3r_1 < 9\mathsf{L}_i/\varepsilon$. Also, we have,

$$\mathsf{d}_{\mathcal{M}'}(y'_1, y'_2) = \mathsf{d}_{\mathcal{X}}(\alpha(y_1), \alpha(y_2)) + |\mathsf{h}(y_1) - \mathsf{h}(y_2)| \leq \mathsf{l}_i + \mathsf{h}_{\max}(A'_i) + \mathsf{h}_{\max}(B'_i) = \mathsf{L}_i. \tag{3.3}$$

Let $\overline{\mathsf{q}}$ be any other point in $\mathcal{U}_i$, and let the points $\overline{y_1}$ and $\overline{y_2}$ be the points found by the algorithm such that

$\overline{y_1}' \in A_i'$ and $\overline{y_2}' \in B_i'$. Since y_1' is also in A_i', we have by Lemma 3.5.6 that $\mathsf{d}_{\mathcal{M}'}(y_1', \overline{y_1}') \leq \mathrm{diam}_{\mathcal{M}'}(A_i) \leq \mathsf{L}_i/c$. As such,

$$\begin{aligned}\mathsf{diam}(\mathcal{U}_i') &= \max_{\mathsf{q}', \overline{\mathsf{q}}' \in \mathcal{U}_i'} \mathsf{d}_{\mathcal{M}'}(\mathsf{q}', \overline{\mathsf{q}}') \leq \max_{\mathsf{q}', \overline{\mathsf{q}}' \in \mathcal{U}_i'} (\mathsf{d}_{\mathcal{M}'}(\mathsf{q}', y_1') + \mathsf{d}_{\mathcal{M}'}(y_1', \overline{y_1}') + \mathsf{d}_{\mathcal{M}'}(\overline{\mathsf{q}}', \overline{y_1}')) \\ &\leq 9\mathsf{L}_i/\varepsilon + \mathsf{L}_i/c + 9\mathsf{L}_i/\varepsilon = O(\mathsf{L}_i/\varepsilon)\,,\end{aligned}$$

for ε small enough. ■

Lemma 3.5.8 *For a query point* q*, the associated distances* r_2 *and* L_i *satisfy* $r_2 \geq \mathsf{L}_i/18$.

Proof: Let u' be the point with maximum height in A_i'; that is $\mathsf{h}(u) = \mathsf{h}_{\max}(A_i')$. By Lemma 3.5.6, we have $\mathsf{d}_{\mathcal{M}'}(u', y_1') \leq \mathsf{L}_i/c$. The definition of the distance in $\mathcal{M}'$, gives

$$\mathsf{h}_{\max}(A_i') - \mathsf{h}(y_1) \leq |\mathsf{h}_{\max}(A_i') - \mathsf{h}(y_1)| = |\mathsf{h}(u) - \mathsf{h}(y_1)| \leq \mathsf{d}_{\mathcal{M}'}(u', y_1') \leq \mathsf{L}_i/c,$$

and so $\mathsf{h}(y_1) \geq \mathsf{h}_{\max}(A_i') - \mathsf{L}_i/c$. Similarly we have, $\mathsf{h}(y_2) \geq \mathsf{h}_{\max}(B_i') - \mathsf{L}_i/c$. We have $r_1 = \mathsf{d}_{\mathcal{X}}(\mathsf{q}, y_1) \geq \mathsf{d}_{\mathcal{X}}(y_1, \mathcal{M}) = \mathsf{d}_{\mathcal{X}}(y_1, \alpha(y_1)) = \mathsf{h}(y_1)$ and similarly $r_2 = \mathsf{d}_{\mathcal{X}}(\mathsf{q}, y_2) \geq \mathsf{h}(y_2)$. Noting that, $r_2 \geq \mathsf{d}_{\mathcal{X}}(\mathsf{q}, \mathsf{P}) \geq r_1/(1+\varepsilon/10) \geq (10/11)r_1$ we get,

$$2.1r_2 = r_2 + \frac{11}{10}r_2 \geq r_2 + r_1 \geq \mathsf{h}(y_2) + \mathsf{h}(y_1) \geq \mathsf{h}_{\max}(A_i') + \mathsf{h}_{\max}(B_i') - \frac{2\mathsf{L}_i}{c}. \tag{3.4}$$

Let $z' \in A_i'$ and $w' \in B_i'$ be such that $\mathsf{d}_{\mathcal{X}}(\alpha(z), \alpha(w)) = \mathsf{l}_i$. Observing that $\mathsf{d}_{\mathcal{M}'}(\mathsf{q}', y_1') \leq 3\mathsf{d}_{\mathcal{X}}(\mathsf{q}, y_1) = 3r_1$ and similarly $\mathsf{d}_{\mathcal{M}'}(\mathsf{q}', y_2') \leq 3\mathsf{d}_{\mathcal{X}}(\mathsf{q}, y_2) = 3r_2$, we have by the triangle inequality that

$$\begin{aligned}&\mathsf{d}_{\mathcal{M}'}(\mathsf{q}', z') \leq \mathsf{d}_{\mathcal{M}'}(\mathsf{q}', y_1') + \mathsf{d}_{\mathcal{M}'}(y_1', z') \leq 3r_1 + \mathrm{diam}(A_i') \leq 3r_1 + \mathsf{L}_i/c, \\ \text{and} \qquad &\mathsf{d}_{\mathcal{M}'}(\mathsf{q}', w') \leq \mathsf{d}_{\mathcal{M}'}(\mathsf{q}', y_2') + \mathsf{d}_{\mathcal{M}'}(y_2', w') \leq 3r_2 + \mathrm{diam}(B_i') \leq 3r_2 + \mathsf{L}_i/c,\end{aligned}$$

by Lemma 3.5.6. By the triangle inequality, we have

$$\mathsf{l}_i \leq \mathsf{d}_{\mathcal{M}'}(z', w') \leq \mathsf{d}_{\mathcal{M}'}(z', \mathsf{q}') + \mathsf{d}_{\mathcal{M}'}(\mathsf{q}', w') \leq 3r_1 + 3r_2 + \frac{2\mathsf{L}_i}{c} \leq 6.3r_2 + \frac{2\mathsf{L}_i}{c},$$

as $r_1 \leq (11/10)r_2$. Thus we have,

$$6.3r_2 \geq \mathsf{l}_i - \frac{2\mathsf{L}_i}{c}. \tag{3.5}$$

By Eq. (3.4) and Eq. (3.5), we have, for $c \geq 8$, that

$$\begin{aligned} 9r_2 \geq 2.1r_2 + 6.3r_2 &\geq \left(\mathsf{h}_{\max}(A'_i) + \mathsf{h}_{\max}(B'_i) - \frac{2\mathsf{L}_i}{c}\right) + \left(\mathsf{l}_i - \frac{2\mathsf{L}_i}{c}\right) \\ &= \mathsf{h}_{\max}(A'_i) + \mathsf{h}_{\max}(B'_i) + \mathsf{l}_i - \frac{4\mathsf{L}_i}{c} \geq \mathsf{L}_i - \frac{\mathsf{L}_i}{2} = \frac{\mathsf{L}_i}{2}, \end{aligned}$$

which implies $\mathsf{L}_i \leq 18r_2$. ■

Suppose $\overline{\mathsf{q}}$ was added to $\mathcal{U}_i$ after q. We want to show that for $\mathsf{q}, \overline{\mathsf{q}} \in \mathcal{U}_i$ we must have $\mathsf{d}_{\mathcal{M}'}(\mathsf{q}', \overline{\mathsf{q}}') > \varepsilon r_2/5$ where $r_2 = \mathsf{d}_{\mathcal{X}}(\mathsf{q}, y_2)$. We establish this through a sequence of lemmas. The proof is essentially by contradiction, and the next four lemmas assume the contrary to derive a contradiction. Roughly speaking, the assumption that $\mathsf{d}_{\mathcal{M}'}(\mathsf{q}', \overline{\mathsf{q}}') = \mathsf{d}_{\mathcal{X}}(\mathsf{q}, \overline{\mathsf{q}}) \leq \varepsilon r_2/5$ places $\overline{\mathsf{q}}$ in the chipped off region of the crescent region C_{q}. It turns out that $\overline{\mathsf{q}}$ is far from both the approximate nearest neighbor of q, which is y_1 and the approximate nearest neighbor of q outside an environ of y_1, which is y_2. Under the assumption $\mathsf{q}, \overline{\mathsf{q}} \in \mathcal{U}_i$ we should however be able to find the corresponding approximate nearest neighbors for $\overline{\mathsf{q}}$ close to those of q. Enforcing the constraint that any approximate nearest neighbor of $\overline{\mathsf{q}}$ cannot be the second approximate nearest neighbor of q, which is y_2, leads to either counting discrepancies or geometric contradictions arising from the triangle inequality.

Lemma 3.5.9 *Let $\mathsf{q}, \overline{\mathsf{q}}$ be two points of $\mathcal{U}_i$, such that $\overline{\mathsf{q}}$ was added after q. If $\mathsf{d}_{\mathcal{X}}(\mathsf{q}, \overline{\mathsf{q}}) \leq \varepsilon r_2/5$, then (i) $\mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, y_1) \leq (5/4\varepsilon)\rho_1$, and (ii) $r_2 \geq (2/\varepsilon)r_1$.*

Proof: Since $\overline{\mathsf{q}}$ created a new region it lies outside $C_{\mathsf{q}} = \mathsf{ball}_{\mathcal{X}}(\mathsf{q}, \varepsilon r_2/5) \setminus \mathsf{ball}_{\mathcal{X}}(y_1, 5\rho_1/4\varepsilon)$. Since by assumption $\overline{\mathsf{q}} \in \mathsf{ball}_{\mathcal{X}}(\mathsf{q}, \varepsilon r_2/5)$, it must be the case that $\overline{\mathsf{q}} \in \mathsf{ball}_{\mathcal{X}}(y_1, 5\rho_1/4\varepsilon)$, as otherwise $\overline{\mathsf{q}} \in C_{\mathsf{q}}$. Thus, these two balls intersect, and

$$\frac{\varepsilon}{5} r_2 + \frac{5}{4\varepsilon}\rho_1 \geq \mathsf{d}_{\mathcal{X}}(\mathsf{q}, y_1) = r_1.$$

But $\rho_1 \leq \varepsilon r_1/3$ and so $r_1 \geq (3/\varepsilon)\rho_1$, implying

$$\frac{\varepsilon}{5} r_2 + \frac{5}{12} r_1 \geq \frac{\varepsilon}{5} r_2 + \frac{5}{12} \cdot \frac{3}{\varepsilon}\rho_1 = \frac{\varepsilon}{5} r_2 + \frac{5}{4\varepsilon}\rho_1 \geq r_1 \quad \Longrightarrow \quad r_2 \geq \frac{35}{12\varepsilon} r_1 \geq \frac{2}{\varepsilon} r_1.$$

■

Lemma 3.5.10 *Let $\mathsf{q}, \overline{\mathsf{q}}$ be two points in $\mathcal{U}_i$ such that $\overline{\mathsf{q}}$ was added after q. If $\mathsf{d}_{\mathcal{X}}(\mathsf{q}, \overline{\mathsf{q}}) \leq \varepsilon r_2/5$ then, for sufficiently small ε and sufficiently large c, we have that*

(A) $r_1 \leq \varepsilon \mathsf{L}_i$.

(B) $\mathsf{d}_{\mathcal{X}}(y_1, \overline{\mathsf{q}}) \leq 5r_1/12 \leq \varepsilon \mathsf{L}_i$.

(C) $\mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, B_i) \geq \mathsf{L}_i/120$.

Proof: (A) By Eq. (3.3) we have $\mathsf{d}_{\mathcal{M}'}(y_1', y_2') \leq \mathsf{L}_i$. Now, by Lemma 3.5.9, we have $r_2 \geq (2/\varepsilon) r_1$. As such, by the triangle inequality, and by Lemma 3.1.3, we have

$$\begin{aligned} \mathsf{L}_i \geq \mathsf{d}_{\mathcal{M}'}(y_1', y_2') &\geq \mathsf{d}_{\mathcal{M}'}(\mathsf{q}', y_2') - \mathsf{d}_{\mathcal{M}'}(\mathsf{q}', y_1') \geq \mathsf{d}_{\mathcal{X}}(\mathsf{q}, y_2) - 3\mathsf{d}_{\mathcal{X}}(\mathsf{q}, y_1) \\ &\geq r_2 - 3r_1 \geq 2r_1/\varepsilon - 3r_1 \geq r_1/\varepsilon, \end{aligned} \tag{3.6}$$

for $\varepsilon \leq 1/3$. Thus $\mathsf{L}_i \geq r_1/\varepsilon$.

(B) In terms of r_2, by Eq. (3.6), we have

$$\mathsf{d}_{\mathcal{M}'}(y_1', y_2') \geq r_2 - 3r_1 \geq r_2 - \frac{3\varepsilon r_2}{2} \geq \frac{r_2}{2} \geq \frac{\mathsf{L}_i}{36}, \tag{3.7}$$

since by Lemma 3.5.8 $r_2 \geq \mathsf{L}_i/18$ for $\varepsilon \leq 1/3$ and by Lemma 3.5.9 $r_1 \leq \varepsilon r_2/2$. Now $\overline{\mathsf{q}}$ lies inside $\mathsf{ball}_{\mathcal{X}}(y_1, (5/4\varepsilon)\rho_1)$ and as $\rho_1 \leq (\varepsilon/3) r_1$ (see Figure 3.6), we have

$$\mathsf{d}_{\mathcal{X}}(y_1, \overline{\mathsf{q}}) \leq (5/4\varepsilon)\rho_1 \leq (5/4\varepsilon)(\varepsilon/3) r_1 \leq 5r_1/12 \leq r_1 \leq \varepsilon \mathsf{L}_i,$$

by (A).

(C) Let z be an arbitrary point in B_i and notice that by Eq. (3.7) and the triangle inequality we have,

$$\begin{aligned} \mathsf{d}_{\mathcal{M}'}(\overline{\mathsf{q}}', z') &\geq \mathsf{d}_{\mathcal{M}'}(y_1', z') - \mathsf{d}_{\mathcal{M}'}(\overline{\mathsf{q}}', y_1') \geq \mathsf{d}_{\mathcal{M}'}(y_1', y_2') - \mathsf{d}_{\mathcal{M}'}(y_2', z') - \mathsf{d}_{\mathcal{M}'}(\overline{\mathsf{q}}', y_1') \\ &\geq \frac{\mathsf{L}_i}{36} - \operatorname{diam}(B_i') - 3\mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, y_1) \geq \frac{\mathsf{L}_i}{36} - \frac{\mathsf{L}_i}{c} - 3\varepsilon \mathsf{L}_i \geq \frac{\mathsf{L}_i}{40}, \end{aligned}$$

also using Lemma 3.1.3 (B) and Lemma 3.5.6 for sufficiently small ε and sufficiently large c. Now, Lemma 3.1.3 (B) implies that $\mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, z) \geq \mathsf{d}_{\mathcal{M}'}(\overline{\mathsf{q}}', z')/3 \geq \mathsf{L}_i/120$. ■

Lemma 3.5.11 *Let $\mathsf{q}, \overline{\mathsf{q}}$ be two points in $\mathcal{U}_i$ such that $\overline{\mathsf{q}}$ was added after q, and suppose $\mathsf{d}_{\mathcal{X}}(\mathsf{q}, \overline{\mathsf{q}}) \leq \varepsilon r_2/5$. Let $A_i^+ = A_i \cup \{f_1\}$, where f_1 is the furthest point from y_1 in the set $\mathsf{ball}_{\mathcal{X}}(y_1, \varepsilon r_1/3) \cap \mathsf{P}$. Then, for sufficiently small ε and sufficiently large c, we have $B_i \cap A_i^+ = \emptyset$. In particular, we have $\mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, B_i) > 2 \max_{u \in A_i^+} \mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, u)$.*

Proof: First, let u be any point in A_i. Then, by Lemma 3.1.3 (B), the triangle inequality, Lemma 3.5.6 and

Lemma 3.5.10 we have, for c sufficiently large and ε sufficiently small, that

$$\begin{aligned}\mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, u) &\leq \mathsf{d}_{\mathcal{M}'}(\overline{\mathsf{q}}', u') \leq \mathsf{d}_{\mathcal{M}'}(\overline{\mathsf{q}}', y_1') + \mathsf{d}_{\mathcal{M}'}(y_1', u') \leq 3\mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, y_1) + \operatorname{diam}_{\mathcal{M}'}(A_i') \\ &\leq 3\varepsilon \mathsf{L}_i + \frac{\mathsf{L}_i}{c} < \frac{\mathsf{L}_i}{240}.\end{aligned}$$

We also have by the triangle inequality,

$$\mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, f_1) \leq \mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, y_1) + \mathsf{d}_{\mathcal{X}}(y_1, f_1) \leq \varepsilon \mathsf{L}_i + \frac{\varepsilon}{3} r_1 \leq \varepsilon \mathsf{L}_i + \frac{\varepsilon^2}{3} \mathsf{L}_i < \frac{\mathsf{L}_i}{240},$$

since $\mathsf{d}_{\mathcal{X}}(y_1, f_1) \leq \varepsilon r_1/3$ and by Lemma 3.5.10. As such, for sufficiently large c and small ε, we have

$$\max_{u \in A_i^+} \mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, u) < \frac{\mathsf{L}_i}{240}. \tag{3.8}$$

On the other hand, for any $z \in B_i$, we have by Lemma 3.5.10 (C) that $\mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, z) \geq \mathsf{L}_i/120$. As such, by Eq. (3.8), we have

$$\mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, B_i) = \min_{z \in B_i} \mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, z) \geq \frac{\mathsf{L}_i}{120} = 2\frac{\mathsf{L}_i}{240} > 2 \max_{u \in A_i^+} \mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, u)\,.$$

We conclude that $B_i \cap A_i^+ = \emptyset$. ■

Remark 3.5.12 *A subtle (but minor) technicality is that we require $\rho_1 \neq 0$, where $\rho_1 = \mathsf{d}_{\mathcal{X}}(y_1, f_1)$. This can be enforced by replicating every point of P, and assigning infinitesimally small positive to the distance between a point and its copy. Clearly, for this modified point set this condition holds.*

Lemma 3.5.13 *Let $\mathsf{q}, \overline{\mathsf{q}}$ be two points in $\mathcal{U}_i$, such that $\overline{\mathsf{q}}$ was added after q. For a sufficiently small ε and a sufficiently large c, we have that $\mathsf{d}_{\mathcal{X}}(\mathsf{q}, \overline{\mathsf{q}}) > \varepsilon r_2/5$.*

Proof: We assume for the sake of contradiction that $\mathsf{d}_{\mathcal{X}}(\mathsf{q}, \overline{\mathsf{q}}) \leq \varepsilon r_2/5$. Let $\overline{y_1} \in A_i$ be the $(1+\varepsilon/10)$-ANN found by the algorithm for $\overline{\mathsf{q}}$, and let $\overline{y_2}$ be the $(1+\varepsilon/10)$-ANN of $\overline{\mathsf{q}}$ in $\mathsf{P} \setminus \mathsf{ball}_{\mathcal{X}}(\overline{y_1}, \varepsilon\overline{r_1}/3)$, where $\overline{r_1} = \mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, \overline{y_1})$. We have

$$\overline{r_1} = \mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, \overline{y_1}) \leq \left(1 + \frac{\varepsilon}{10}\right) \mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, \mathsf{P}) \leq \left(1 + \frac{\varepsilon}{10}\right) \mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, y_1) \leq \frac{5}{4\varepsilon}\left(1 + \frac{\varepsilon}{10}\right) \rho_1 < \frac{3}{2\varepsilon}\rho_1,$$

by Lemma 3.5.9 (i) and as $\overline{y_1}$ is a $(1+\varepsilon/10)$-ANN of $\overline{\mathsf{q}}$ in P. The strict inequality follows under the assumption

$\rho_1 > 0$, see Remark 3.5.12. As in Lemma 3.5.11 let $A_i^+ = A_i \cup \{f_1\}$. By Lemma 3.5.11, we have

$$\mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, B_i) > (1+\varepsilon/10) \max_{u \in A_i^+} \mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, u)\,,$$

as $\mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, B_i) > 2\max_{u\in A_i^+} \mathsf{d}_{\mathcal{X}}(\overline{\mathsf{q}}, u)$. If A_i^+ is not contained in $\mathsf{ball}_{\mathcal{X}}(\overline{y_1}, \varepsilon\overline{r_1}/3)$, then there is a point in $A_i^+ \setminus \mathsf{ball}_{\mathcal{X}}(\overline{y_1}, \varepsilon\overline{r_1}/3)$ that is, by a factor of $(1+\varepsilon/10)$, closer to $\overline{\mathsf{q}}$ than B_i. But this implies that $\overline{y_2} \notin B_i$, and this is a contradiction to the definition of $\overline{\mathsf{q}}$ ($\overline{\mathsf{q}}$ by definition has $\overline{y_1} \in A_i$ and $\overline{y_2} \in B_i$). Thus, A_i^+ is contained in $\mathsf{ball}_{\mathcal{X}}(\overline{y_1}, \varepsilon\overline{r_1}/3)$.

As such, we have $y_1 \in A_i \subseteq A_i^+ \subseteq \mathsf{ball}_{\mathcal{X}}(\overline{y_1}, \varepsilon\overline{r_1}/3)$ (and, by definition $f_1 \in A_i^+$, and thus f_1 also belongs to this ball). We conclude

$$\rho_1 = \mathsf{d}_{\mathcal{X}}(y_1, f_1) \leq \mathsf{d}_{\mathcal{X}}(y_1, \overline{y_1}) + \mathsf{d}_{\mathcal{X}}(\overline{y_1}, f_1) \leq 2\frac{\varepsilon\overline{r_1}}{3} < \frac{2\varepsilon}{3} \cdot \frac{3}{2\varepsilon}\rho_1 = \rho_1,$$

for ε sufficiently small. This is a contradiction. ■

Lemma 3.5.14 *Let $\mathsf{q}, \overline{\mathsf{q}}$ be two points in $\mathcal{U}_i$, such that $\overline{\mathsf{q}}$ was added after q. Then for sufficiently small ε and sufficiently large c we have, $\mathsf{d}_{\mathcal{M}'}(\mathsf{q}', \overline{\mathsf{q}}') = \mathsf{d}_{\mathcal{X}}(\mathsf{q}, \overline{\mathsf{q}}) > \varepsilon r_2/5 = \Omega(\varepsilon \mathsf{L}_i)$.*

Proof: Since $\mathsf{q}, \overline{\mathsf{q}} \in \mathcal{M}$ it follows from Lemma 3.1.3 that $\mathsf{d}_{\mathcal{M}'}(\mathsf{q}', \overline{\mathsf{q}}') = \mathsf{d}_{\mathcal{X}}(\mathsf{q}, \overline{\mathsf{q}})$. By Lemma 3.5.13, we have $\mathsf{d}_{\mathcal{X}}(\mathsf{q}, \overline{\mathsf{q}}) > \varepsilon r_2/5$. From Lemma 3.5.8 it follows that $\varepsilon r_2/5 = \Omega(\varepsilon \mathsf{L}_i)$. ■

Lemma 3.5.15 *We have that* $\max(|\mathcal{U}_i|, |\mathcal{V}_i|) = \varepsilon^{-O(\tau)}$.

Proof: From Lemma 3.5.7 and Lemma 3.5.14 it follows that the spread of the set $\mathcal{U}_i'$ is bounded by

$$O\left(\frac{\mathsf{L}_i/\varepsilon}{\varepsilon \mathsf{L}_i}\right) = O\left(\frac{1}{\varepsilon^2}\right).$$

Since $\mathcal{U}_i' \subseteq \mathcal{M}'$ which is a space of doubling dimension $O(\tau)$ it follows that $|\mathcal{U}_i'| = \varepsilon^{-O(\tau)}$. The same argument works for $\mathcal{V}_i'$. For any $\mathsf{q} \in \mathcal{M}, \mathsf{q}' = (\mathsf{q}, 0)$ and it is easy to see that the mapping $\mathsf{q} \to \mathsf{q}'$ is bijective. As such $|\mathcal{U}_i'| = |\mathcal{U}_i|$, and similarly $|\mathcal{V}_i'| = |\mathcal{V}_i|$, and the claimed bounds follow. ■

The next lemma bounds the number of regions created.

Lemma 3.5.16 *The number of regions created by the algorithm is $n/\varepsilon^{O(\tau)}$.*

Proof: As shown in Lemma 3.5.5 the number of outer regions created is bounded by $\varepsilon^{-O(\tau)}$. Consider an inner region C_{q}. For this point q the algorithm found a valid y_1 and y_2. Now from the definition of a WSPD

there is some i such that $y'_1 \in A'_i, y'_2 \in B'_i$ or $y'_1 \in B'_i, y'_2 \in A'_i$. In other words there is some i such that $\mathsf{q} \in \mathcal{U}_i$ or $\mathsf{q} \in \mathcal{V}_i$. As shown in Lemma 3.5.15 the size of each of these is bounded by $\varepsilon^{-O(\tau)}$. Since the total number of such sets is $2m$ where $m = nc^{O(\tau)}$ is the number of pairs of the WSPD, it follows that the total number of inner regions created is bounded by $(c/\varepsilon)^{O(\tau)} n \leq n\varepsilon^{-O(\tau)}$, for ε sufficiently small. ■

3.5.4 The result

We summarize the result of this section.

Theorem 3.5.17 *The online algorithm presented in Figure 3.5 always returns a $(1+\varepsilon)$-ANN. If the query points are constrained to lie on a subspace of doubling dimension τ, then the maximum number of regions created for the online AVD by the algorithm throughout its execution is $n/\varepsilon^{O(\tau)}$.*

3.6 Conclusions

In this chapter, we considered the ANN problem when the data points can come from an arbitrary metric space (not necessarily an Euclidean space) but the query points come from a subspace of low doubling dimension. We demonstrated that this problem is inherently low dimensional by providing fast ANN data-structures obtained by combining and extending ideas that were previously used to solve ANN for spaces with low doubling dimensions.

Interestingly, one can extend Assouad's type embedding to an embedding that $(1+\varepsilon)$-preserves distances from P to $\mathcal{M}$ (see [HM06] for an example of a similar embedding into the ℓ_∞ norm). This extension requires some work and is not completely obvious. The target dimension is roughly $1/\varepsilon^{O(\tau)}$ in this case. If one restricts oneself to the case where both P and $\mathcal{M}$ are in Euclidean space, then it seems one should be able to extend the embedding of Gottlieb and Krauthgamer [GK11] to get a similar result, with the target dimension having only polynomial dependency on τ. However, computing either embedding efficiently seems quite challenging. Furthermore, even if the embedded points are given, the target dimension in both cases is quite large, and yields results that are significantly weaker than the ones presented here.

The on the fly construction of AVD without any knowledge of the query subspace (Section 3.5) seems like a natural candidate for a practical algorithm for ANN. Such an implementation would require an efficient way to perform point-location in the generated regions. We leave the problem of developing such a data-structure as an open question for further research. In particular, there might be a middle ground between our two ANN data-structures that yields an efficient and practical ANN data-structure while having very limited access to the query subspace.

Chapter 4

Sublinear Space Data-Structures for k-ANN in $\mathbb{R}^d$

4.1 Introduction

A more general problem than the approximate nearest neighbor problem is the k-nearest neighbors problem where given a point set P, one is interested in finding the k points in P nearest to the query point q. This is widely used in pattern recognition, where points in P are labeled and the majority label among the k-nearest neighbors is used to label the query point for some appropriate value of k. In this chapter, we are interested in the more restricted problem of approximating the distance to the kth-nearest neighbor and finding a data point achieving the approximation. We call this problem the $(1+\varepsilon, k)$-***approximate nearest neighbor*** ($(1+\varepsilon, k)$-ANN) problem. This problem is widely used for density estimation in statistics, with $k \approx \sqrt{n}$ [Sil86]. It is also used in meshing (with $k = 2d$), or to compute the local feature size of a point set in $\mathbb{R}^d$ [Rup95]. The problem also has applications in non-linear dimensionality reduction; finding low dimensional structures in data – more specifically low dimensional submanifolds embedded in Euclidean spaces. Algorithms like ISOMAP, LLE, Hessian-LLE, SDE and others, use the k-nearest neighbor as a subroutine [Ten98, BSLT00, MS94, WS06].

Density estimation. Given distributions $\mu_1, \ldots, \mu_k$ defined over $\mathbb{R}^d$, and a query point q, we want to compute the *a posteriori* probabilities of q being generated by one of these distributions. This approach is used in unsupervised learning as a way to classify a new point. Naturally, in most cases, the distributions are given implicitly; that is, one is given a large number of points sampled from each distribution. So, let μ be such a distribution, and P be a set of n samples. To estimate the density of μ at q, a standard Monte Carlo technique is to consider a ball B centered at q, and count the number of points of P inside B. Specifically, one possible approach that is used in practice [DHS01], is to find the smallest ball centered at q that contains k points of P and use this to estimate the density of μ. The right value of k has to be chosen carefully – if it is too small, then the estimate is unstable (unreliable), and if it is too large, it either requires the set P to be larger, or the estimate is too "smoothed" out to be useful (values of k that are used in practice are

$\widetilde{O}(\sqrt{n})$), see Duda *et al.* [DHS01] for more details. To do such density estimation, one needs to be able to answer, approximate or exact, k-nearest neighbor queries.

Sometimes one is interested not only in the radius of this ball centered at the query point, but also in the distribution of the points inside this ball. The average distance of a point inside the ball to its center, can be estimated by the sum of distances of the sample points inside the ball to the center. Similarly, the variance of this distance can be estimated by the sum of squared distances of the sample points inside the ball to the center of the ball. As mentioned, density estimation is used in manifold learning and surface reconstruction. For example, Guibas *et al.* [GMM11] recently used a similar density estimate to do manifold reconstruction.

Answering exact k-nearest neighbor queries. Given a point set $\mathsf{P} \subseteq \mathbb{R}^d$, computing the partition of space into regions, such that the k nearest neighbors do not change, is equivalent to computing the ***kth order Voronoi diagram***. Via standard lifting, this is equivalent to computing the first k levels in an arrangement of hyperplanes in $\mathbb{R}^{d+1}$ [Aur91]. More precisely, if we are interested in the kth-nearest neighbor, we need to compute the $(k-1)$-level in this arrangement.

The complexity of the ($\leq k$) levels of a hyperplane arrangement in $\mathbb{R}^{d+1}$ is $\Theta(n^{\lfloor (d+1)/2 \rfloor}(k+1)^{\lceil (d+1)/2 \rceil})$ [CS89]. The exact complexity of the kth-level is not completely understood and achieving tight bounds on its complexity is one of the long-standing open problems in discrete geometry [Mat02]. In particular, via an averaging argument, in the worst case, the complexity of the kth-level is $\Omega\big(n^{\lfloor (d+1)/2 \rfloor}(k+1)^{\lceil (d+1)/2 \rceil - 1}\big)$. As such, the complexity of kth-order Voronoi diagram is $\Omega(nk)$ in two dimensions, and $\Omega(n^2 k)$ in three dimensions.

Thus, to provide a data-structure for answering k-nearest neighbor queries exactly and quickly (i.e., logarithmic query time) in $\mathbb{R}^d$, requires computing the k-level of an arrangement of hyperplanes in $\mathbb{R}^{d+1}$. The space complexity of this structure is prohibitive even in two dimensions (this also effects the preprocessing time). Furthermore, naturally, the complexity of this structure increases as k increases. On the other end of the spectrum one can use partition-trees and parametric search to answer such queries using linear space and query time (roughly) $O\big(n^{1-1/(d+1)}\big)$ [Mat92, Cha10]. One can get intermediate results using standard space/time tradeoffs [AE99].

Known results on approximate k-order Voronoi diagram. Similar to AVD, one can define a AVD for the k-nearest neighbor. The case $k = 1$ is the regular approximate Voronoi diagram [Har01, AM02, AMM09]. The case $k = n$ is the furthest neighbor Voronoi diagram. It is not hard to see that it has a constant size approximation (see [Har99], although it was probably known before). The results in this chapter can be interpreted as bridging between these two extremes.

4.2 Preliminaries

4.2.1 Problem definition

Given a set P of n points in $\mathbb{R}^d$ and a number k, $1 \leq k \leq n$, consider a point q and order the points of P by their distance from q; that is,

$$\|\mathsf{u}_1 - \mathsf{q}\| \leq \|\mathsf{u}_2 - \mathsf{q}\| \leq \cdots \leq \|\mathsf{u}_n - \mathsf{q}\|,$$

where $\mathsf{P} = \{\mathsf{u}_1, \mathsf{u}_2, \ldots, \mathsf{u}_n\}$. The point $\mathsf{u}_k = \mathsf{nn}_k(\mathsf{q}, \mathsf{P})$ is the ***kth-nearest neighbor*** of q and $\mathsf{d}_k(\mathsf{q}, \mathsf{P}) = \|\mathsf{u}_k - \mathsf{q}\|$ is the ***kth-nearest neighbor distance***. The nearest neighbor distance (i.e., $k = 1$) is $\mathsf{d}(\mathsf{q}, \mathsf{P}) = \min_{\mathsf{u} \in \mathsf{P}} \|\mathsf{u} - \mathsf{q}\|$. The global minimum of $\mathsf{d}_k(\mathsf{q}, \mathsf{P})$, denoted by $r_{\mathrm{opt}}(\mathsf{P}, k) = \min_{\mathsf{q} \in \mathbb{R}^d} \mathsf{d}_k(\mathsf{q}, \mathsf{P})$, is the radius of the smallest ball containing k points of P.

Observation 4.2.1 *For any $\mathsf{p}, \mathsf{u} \in \mathbb{R}^d$, k and a set $\mathsf{P} \subseteq \mathbb{R}^d$, we have that $\mathsf{d}_k(\mathsf{u}, \mathsf{P}) \leq \mathsf{d}_k(\mathsf{p}, \mathsf{P}) + \|\mathsf{u} - \mathsf{p}\|$.*

Namely, the function $\mathsf{d}_k(\mathsf{q}, \mathsf{P})$ is 1-Lipschitz. The problem at hand is to preprocess P such that given a query point q one can compute u_k quickly. The standard ***nearest neighbor problem*** is this problem for $k = 1$. In the $(1+\varepsilon, k)$***-approximate nearest neighbor*** ($(1+\varepsilon, k)$-ANN) problem, given q, k and $\varepsilon > 0$, one wants to find a point $\mathsf{u} \in \mathsf{P}$, such that $(1 - \varepsilon) \|\mathsf{u}_k - \mathsf{q}\| \leq \|\mathsf{u} - \mathsf{q}\| \leq (1 + \varepsilon) \|\mathsf{u}_k - \mathsf{q}\|$.

4.2.2 Basic tools

A ball b of radius r in $\mathbb{R}^d$, centered at a point p, can be interpreted as a point in $\mathbb{R}^{d+1}$, denoted by $\mathsf{b}' = (\mathsf{p}, r)$. For a regular point $\mathsf{p} \in \mathbb{R}^d$, its corresponding image under this transformation is the ***mapped*** point $\mathsf{p}' = (\mathsf{p}, 0) \in \mathbb{R}^{d+1}$.

Given point $\mathsf{u} = (\mathsf{u}_1, \ldots, \mathsf{u}_d) \in \mathbb{R}^d$ we will denote its Euclidean norm by $\|\mathsf{u}\|$. We will consider a point $\mathsf{u} = (\mathsf{u}_1, \mathsf{u}_2, \ldots, \mathsf{u}_{d+1}) \in \mathbb{R}^{d+1}$ to be in the product metric of $\mathbb{R}^d \times \mathbb{R}$ and endowed with the product metric norm

$$\|\mathsf{u}\|_{\oplus} = \sqrt{\mathsf{u}_1^2 + \cdots + \mathsf{u}_d^2} + |\mathsf{u}_{d+1}|.$$

It can be verified that the above defines a norm and the following holds for it.

Lemma 4.2.2 *For any $\mathsf{u} \in \mathbb{R}^{d+1}$ we have $\|\mathsf{u}\| \leq \|\mathsf{u}\|_{\oplus} \leq \sqrt{2}\, \|\mathsf{u}\|$.*

The distance of a point to a set under the $\|\cdot\|_{\oplus}$ norm is denoted by $\mathsf{d}_{\oplus}(\mathsf{u}, \mathsf{P})$.

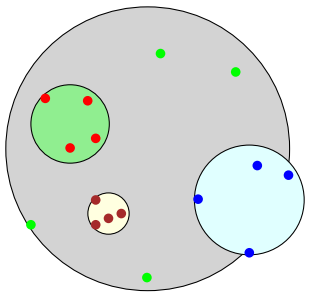


Figure 4.1: Quorum clustering for $n = 16$ and $k = 4$.

Assumption 4.2.3 *We assume that k divides n; otherwise one can easily add fake points as necessary at infinity.*

Assumption 4.2.4 *We also assume that the point set P is contained in $[1/2, 1/2 + 1/n]^d$, where $n = |\mathsf{P}|$. This can be achieved by scaling and translation (which does not affect the distance ordering). Moreover, we assume the queries are restricted to the unit cube $U = [0, 1]^d$.*

Quorum clustering

Given a set P of n points in $\mathbb{R}^d$, and a number $k \geq 1$, where $k|n$, we start with the smallest ball b_1 that contains k points of P, that is $\mathrm{radius}(\mathsf{b}_1) = r_{\mathrm{opt}}(\mathsf{P}, k)$. Let $\mathsf{P}_1 = \mathsf{P} \cap \mathsf{b}_1$. Continue on the set of points $\mathsf{P} \setminus \mathsf{P}_1$ by finding the smallest ball that contains k points of $\mathsf{P} \setminus \mathsf{P}_1$, and so on. Let $\mathsf{b}_1, \mathsf{b}_2, \ldots, \mathsf{b}_{n/k}$ denote the set of balls computed by this algorithm and let $\mathsf{P}_i = (\mathsf{P} \setminus (\mathsf{P}_1 \cup \cdots \cup \mathsf{P}_{i-1})) \cap \mathsf{b}_i$. See Figure 4.1 for an example. Let c_i and r_i denote the center and radius respectively, of b_i, for $i = 1, \ldots, n/k$. A slight symbolic perturbation can guarantee that (i) each ball b_i contains exactly k points of P, and (ii) all the centers $\mathsf{c}_1, \mathsf{c}_2, \ldots, \mathsf{c}_k$, are distinct points. Observe that $\mathsf{r}_1 \leq \mathsf{r}_2 \leq \cdots \leq \mathsf{r}_{n/k} \leq \mathsf{diam}(\mathsf{P})$. Such a partition of P into n/k clusters is a ***quorum clustering***. An algorithm for computing it is provided in Carmi *et al.* [CDH+05]. We assume we have a black-box procedure **QuorumCluster**(P, k) [CDH+05] that computes an ***approximate*** quorum clustering. It returns a list of balls, $(\mathsf{c}_1, \mathsf{r}_1), \ldots, (\mathsf{c}_{n/k}, \mathsf{r}_{n/k})$. The algorithm of Carmi *et al.* [CDH+05] computes such a sequence of balls, where each ball is a 2-approximation to the smallest ball containing k points of the remaining points. The following is an improvement over the result of Carmi *et al.* [CDH+05].

Lemma 4.2.5 *Given a set P of n points in $\mathbb{R}^d$ and parameter k, where $k|n$, one can compute, in $O(n \log n)$ time, a sequence of n/k balls, such that, for all $i, 1 \leq i \leq n/k$, we have*

(A) *For every ball $(\mathsf{c}_i, \mathsf{r}_i)$ there is an associated subset P_i of k points of $\mathsf{Q}_i = \mathsf{P} \setminus (\mathsf{P}_i \cup \ldots \cup \mathsf{P}_{i-1})$, that it covers.*

(B) *The ball* $(\mathsf{c}_i, \mathsf{r}_i)$ *is a 2-approximation to the smallest ball covering* k *points in* Q_i*; that is,* $\mathsf{r}_i/2 \leq r_{\mathrm{opt}}(\mathsf{Q}_i, k) \leq \mathsf{r}_i$.

Proof: The guarantee of Carmi *et al.* is slightly worse – their algorithm running time is $O(n \log^d n)$. They use a dynamic data-structure for answering $O(n)$ queries, that report how many points are inside a query canonical square. Since they use orthogonal range trees this requires $O(\log^d n)$ time per query. Instead, one can use dynamic quadtrees. More formally, we store the points using linear ordering [Har11], using any balanced data-structure. A query to decide the number of points inside a canonical node corresponds to an interval query (i.e., reporting the number of elements that are inside a query interval), and can be performed in $O(\log n)$ time. Plugging this data-structure into the algorithm of Carmi *et al.* [CDH+05] gives the desired result. ■

4.3 A $(15, k)$-ANN in sublinear space

Lemma 4.3.1 *Let* P *be a set of* n *points in* $\mathbb{R}^d$*,* $k \geq 1$ *be a number such that* $k|n$*,* $(\mathsf{c}_1, \mathsf{r}_1), \ldots, (\mathsf{c}_{n/k}, \mathsf{r}_{n/k})$*, be the list of balls returned by* **QuorumCluster**(P, k)*, and let* $x = \min_{i=1,\ldots,n/k} (\|\mathsf{c}_i - \mathsf{q}\| + \mathsf{r}_i)$*. We have that* $x/5 \leq \mathsf{d}_k(\mathsf{q}, \mathsf{P}) \leq x$.

Proof: For any $i = 1, \ldots, n/k$, we have $\mathsf{b}_i = \mathsf{ball}(\mathsf{c}_i, \mathsf{r}_i) \subseteq \mathsf{ball}(\mathsf{q}, \|\mathsf{c}_i - \mathsf{q}\| + \mathsf{r}_i)$. Since $|\mathsf{b}_i \cap \mathsf{P}| \geq k$, we have $\mathsf{d}_k(\mathsf{q}, \mathsf{P}) \leq \|\mathsf{c}_i - \mathsf{q}\| + \mathsf{r}_i$. As such, $\mathsf{d}_k(\mathsf{q}, \mathsf{P}) \leq x = \min_{i=1,\ldots,n/k} (\|\mathsf{c}_i - \mathsf{q}\| + \mathsf{r}_i)$.

For the other direction, let i be the first index such that $\mathsf{ball}(\mathsf{q}, \mathsf{d}_k(\mathsf{q}, \mathsf{P}))$ contains a point of P_i, where P_i is the set of k points of P assigned to b_i. Then, we have

$$\mathsf{r}_i/2 \leq r_{\mathrm{opt}}(\mathsf{Q}_i, k) \leq \mathsf{d}_k(\mathsf{q}, \mathsf{P}),$$

where $\mathsf{Q}_i = \mathsf{P} \setminus (\mathsf{P}_1 \cup \cdots \cup \mathsf{P}_{i-1})$, r_i is a 2-approximation to $r_{\mathrm{opt}}(\mathsf{Q}_i, k)$, and the last inequality follows as $X = \mathsf{ball}(\mathsf{q}, \mathsf{d}_k(\mathsf{q}, \mathsf{P})) \cap \mathsf{P}$ is a set of size k and $X \subseteq \mathsf{Q}_i$. Then,

$$\|\mathsf{c}_i - \mathsf{q}\| - \mathsf{r}_i \leq \mathsf{d}(\mathsf{q}, \mathsf{ball}(\mathsf{c}_i, \mathsf{r}_i)) \leq \mathsf{d}_k(\mathsf{q}, \mathsf{P}),$$

as the distance from q to any $\mathsf{u} \in \mathsf{ball}(\mathsf{c}_i, \mathsf{r}_i)$ satisfies $\|\mathsf{u} - \mathsf{q}\| \geq \|\mathsf{c}_i - \mathsf{q}\| - \mathsf{r}_i$ by the triangle inequality. Putting the above together, we get

$$x = \min_{j=1,\ldots,n/k} (\|\mathsf{c}_j - \mathsf{q}\| + \mathsf{r}_j) \leq \|\mathsf{c}_i - \mathsf{q}\| + \mathsf{r}_i = (\|\mathsf{c}_i - \mathsf{q}\| - \mathsf{r}_i) + 2\mathsf{r}_i \leq 5\mathsf{d}_k(\mathsf{q}, \mathsf{P}).$$

■

Theorem 4.3.2 *Given a set P of n points in $\mathbb{R}^d$, and a number $k \geq 1$ such that $k|n$, one can build a data-structure, in $O(n \log n)$ time, that uses $O(n/k)$ space, such that given any query point $\mathsf{q} \in \mathbb{R}^d$, one can compute, in $O(\log(n/k))$ time, a 15-approximation to $\mathsf{d}_k(\mathsf{q}, \mathsf{P})$.*

Proof: We invoke **QuorumCluster**(P, k) to compute the clusters $(\mathsf{c}_i, \mathsf{r}_i)$, for $i = 1, \ldots, n/k$. For $i = 1, \ldots, n/k$, let $\mathsf{b}'_i = (\mathsf{c}_i, \mathsf{r}_i) \in \mathbb{R}^{d+1}$. We preprocess the set $\mathcal{B}' = \left\{\mathsf{b}'_1, \ldots, \mathsf{b}'_{n/k}\right\}$ for 2-ANN queries (in $\mathbb{R}^{d+1}$ under the Euclidean norm). The preprocessing time for the ANN data structure is $O((n/k)\log(n/k))$, the space used is $O(n/k)$ and the query time is $O(\log(n/k))$ [Har11].

Given a query point $\mathsf{q} \in \mathbb{R}^d$ the algorithm computes a 2-ANN to $\mathsf{q}' = (\mathsf{q}, 0)$, denoted by b'_j, and returns $\left\|\mathsf{q}' - \mathsf{b}'_j\right\|_\oplus$ as the approximate distance.

Observe that, for any i, we have $\|\mathsf{b}'_i - \mathsf{q}'\| \leq \|\mathsf{q}' - \mathsf{b}'_i\|_\oplus \leq \sqrt{2}\,\|\mathsf{b}'_i - \mathsf{q}'\|$ by Lemma 4.2.2. As such, the returned distance to b'_j is a 2-approximation to $\mathsf{d}(\mathsf{q}', \mathcal{B}')$; that is,

$$\mathsf{d}_\oplus(\mathsf{q}', \mathcal{B}') \leq \left\|\mathsf{q}' - \mathsf{b}'_j\right\|_\oplus \leq \sqrt{2}\left\|\mathsf{b}'_j - \mathsf{q}'\right\| \leq 2\sqrt{2}\mathsf{d}(\mathsf{q}', \mathcal{B}') \leq 2\sqrt{2}\mathsf{d}_\oplus(\mathsf{q}', \mathcal{B}')\,.$$

By Lemma 4.3.1, $\mathsf{d}_\oplus(\mathsf{q}', \mathcal{B}')/5 \leq \mathsf{d}_k(\mathsf{q}, \mathsf{P}) \leq \mathsf{d}_\oplus(\mathsf{q}', \mathcal{B}')$. Namely,

$$\left\|\mathsf{q}' - \mathsf{b}'_j\right\|_\oplus / (10\sqrt{2}) \leq \mathsf{d}_k(\mathsf{q}, \mathsf{P}) \leq \left\|\mathsf{q}' - \mathsf{b}'_j\right\|_\oplus,$$

implying the claim. ■

Remark 4.3.3 *The algorithm of Theorem 4.3.2 works for any metric space. Given a set P of n points in a metric space, one can compute n/k points in the product space induced by adding an extra coordinate, such that approximating the distance to the kth nearest neighbor, is equivalent to answering ANN queries on the reduced point set, in the product space.*

4.4 Approximate Voronoi diagram for $\mathsf{d}_k(\mathsf{q}, \mathsf{P})$

Here, we are given a set P of n points in $\mathbb{R}^d$, and our purpose is to build an AVD that approximates the k-ANN distance, while using (roughly) $O(n/k)$ space.

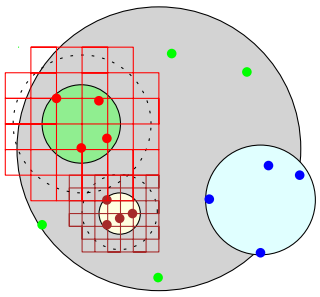


Figure 4.2: Quorum clustering, immediate environs and grids.

4.4.1 Construction

Preprocessing

(A) Compute a quorum clustering for P using Lemma 5.4.6. Let the list of balls returned be $\mathsf{b}_1 = (\mathsf{c}_1, \mathsf{r}_1), \ldots, \mathsf{b}_{n/k} = (\mathsf{c}_{n/k}, \mathsf{r}_{n/k})$.

(B) Compute an exponential grid around each quorum cluster. Specifically, let

$$\mathcal{X} = \bigcup_{i=1}^{n/k} \bigcup_{j=0}^{\lceil \log(32/\varepsilon)+1 \rceil} \boxplus \left(\mathsf{ball}\left(\mathsf{c}_i, 2^j \mathsf{r}_i\right), \frac{\varepsilon}{\zeta_1 d} 2^j \mathsf{r}_i \right) \tag{4.1}$$

be the set of grid cells (see Definition 2.5.4) covering the quorum clusters and their immediate environ, where ζ_1 is a sufficiently large constant, see Figure 4.2.

(C) Intuitively, $\mathcal{X}$ takes care of the region of space immediately next to a quorum cluster[1]. For the other regions of space, we can apply a construction of an approximate Voronoi diagram for the centers of the clusters (the details are somewhat more involved). To this end, lift the quorum clusters into points in $\mathbb{R}^{d+1}$, as follows

$$\mathcal{B}' = \left\{ \mathsf{b}'_1, \ldots, \mathsf{b}'_{n/k} \right\},$$

where $\mathsf{b}'_i = (\mathsf{c}_i, \mathsf{r}_i) \in \mathbb{R}^{d+1}$, for $i = 1, \ldots, n/k$. Note that all points in $\mathcal{B}'$ belong to $U' = [0,1]^{d+1}$ by Assumption 4.2.4. Now build a $(1+\varepsilon/8)$-AVD for $\mathcal{B}'$ using the algorithm of Arya and Malamatos [AM02]. The AVD construction provides a list of canonical cubes covering $[0,1]^{d+1}$ such that in the smallest cube containing the query point, the associated point of $\mathcal{B}'$, is a $(1+\varepsilon/8)$-ANN to the query point. (Note that these cubes are not necessarily disjoint. In particular, the smallest cube containing the query point q is the one that determines the assigned approximate nearest neighbor to q.)

Clip this collection of cubes to the hyperplane $x_{d+1} = 0$ (i.e., throw away cubes that do not have a

[1] That is, intuitively, if the query point falls into one of the grid cells of $\mathcal{X}$, we can answer a query in constant time.

face on this hyperplane). For a cube $\square$ in this collection, denote by $\mathrm{nn}'(\square)$, the point of $\mathcal{B}'$ assigned to it. Let $\mathcal{S}$ be this resulting set of canonical d-dimensional cubes.

(D) Let $\mathcal{W}$ be the space decomposition resulting from overlaying the two collection of cubes, i.e., $\mathcal{X}$ and $\mathcal{S}$. Formally, we compute a compressed quadtree $\mathcal{T}$ that has all the canonical cubes of $\mathcal{X}$ and $\mathcal{S}$ as nodes, and $\mathcal{W}$ is the resulting decomposition of space into cells. One can overlay two compressed quadtrees representing the two sets in linear time [BHTT10, Har11]. Here, a cell associated with a leaf is a canonical cube, and a cell associated with a compressed node is the set difference of two canonical cubes. Each node in this compressed quadtree contains two pointers – to the smallest cube of $\mathcal{X}$, and to the smallest cube of $\mathcal{S}$, that contains it. This information can be computed by doing a BFS on the tree.

For each cell $\square \in \mathcal{W}$ we store the following.

(I) An arbitrary representative point $\square_{\mathsf{rep}} \in \square$.

(II) The point $\mathrm{nn}'(\square) \in \mathcal{B}'$ that is associated with the smallest cell of $\mathcal{S}$ that contains this cell. We also store an arbitrary point, $\mathbf{p}(\square) \in \mathsf{P}$, that is one of the k points belonging to the cluster specified by $\mathrm{nn}'(\square)$.

(III) A number $\beta_k(\square_{\mathsf{rep}})$ that satisfies $\mathsf{d}_k(\square_{\mathsf{rep}}, \mathsf{P}) \leq \beta_k(\square_{\mathsf{rep}}) \leq (1+\varepsilon/4)\mathsf{d}_k(\square_{\mathsf{rep}}, \mathsf{P})$, and a point $\mathrm{nn}_k(\square_{\mathsf{rep}}) \in \mathsf{P}$ that realizes this distance. In order to compute $\beta_k(\square_{\mathsf{rep}})$ and $\mathrm{nn}_k(\square_{\mathsf{rep}})$ use the data-structure of Section 4.6 (see Theorem 4.6.3) or the data-structure of Arya *et al.* [AMM05].

Answering a query

Given a query point q, compute the leaf cell (equivalently the smallest cell) in $\mathcal{W}$ that contains q by performing a point-location query in the compressed quadtree $\mathcal{T}$. Let $\square$ be this cell. Return

$$\min\Big(\left\|\mathsf{q}' - \mathrm{nn}'(\square)\right\|_{\oplus} , \beta_k(\square_{\mathsf{rep}}) + \|\square_{\mathsf{rep}} - \mathsf{q}\| \Big), \tag{4.2}$$

as the approximate value to $\mathsf{d}_k(\mathsf{q}, \mathsf{P})$. Return either $\mathbf{p}(\square)$ or $\mathrm{nn}_k(\square_{\mathsf{rep}})$ depending on which of the two distances $\left\|\mathsf{q}' - \mathrm{nn}'(\square)\right\|_{\oplus}$ or $\beta_k(\square_{\mathsf{rep}}) + \|\square_{\mathsf{rep}} - \mathsf{q}\|$ is smaller (this is the returned approximate value of $\mathsf{d}_k(\mathsf{q}, \mathsf{P})$), as the approximate kth-nearest neighbor.

4.4.2 Correctness

Lemma 4.4.1 *Let $\square \in \mathcal{W}$ and $\mathsf{q} \in \square$. Then the number computed by the algorithm is an upper bound on $\mathsf{d}_k(\mathsf{q}, \mathsf{P})$.*

Proof: By Observation 4.2.1, $\mathsf{d}_k(\mathsf{q}, \mathsf{P}) \leq \mathsf{d}_k(\square_{\mathsf{rep}}, \mathsf{P}) + \|\square_{\mathsf{rep}} - \mathsf{q}\| \leq \beta_k(\square_{\mathsf{rep}}) + \|\square_{\mathsf{rep}} - \mathsf{q}\|$. Now, let $\mathrm{nn}'(\square) = (\mathsf{c}, \mathsf{r})$. We have, by Lemma 4.3.1, that $\mathsf{d}_k(\mathsf{q}, \mathsf{P}) \leq \|\mathsf{c} - \mathsf{q}\| + \mathsf{r} = \|\mathsf{q}' - \mathrm{nn}'(\square)\|_{\oplus}$. As the returned value is the minimum of these two numbers, the claim holds. ■

Lemma 4.4.2 *Consider any query point $\mathsf{q} \in [0,1]^d$, and let $\square$ be the smallest cell of $\mathcal{W}$ that contains the query point. Then, $\mathsf{d}(\mathsf{q}', \mathcal{B}') \leq \|\mathrm{nn}'(\square) - \mathsf{q}'\| \leq (1 + \varepsilon/8)\mathsf{d}(\mathsf{q}', \mathcal{B}')$.*

Proof: Observe that the space decomposition generated by $\mathcal{W}$ is a refinement of the decomposition generated by the Arya and Malamatos [AM02] AVD construction, when applied to $\mathcal{B}'$, and restricted to the d dimensional subspace we are interested in (i.e., $x_{d+1} = 0$). As such, $\mathrm{nn}'(\square)$ is the point returned by the AVD for this query point before the refinement, thus implying the claim. ■

The query point is close to a quorum cluster of the right size

Lemma 4.4.3 *Consider a query point q, and let $\square \subseteq \mathbb{R}^d$ be any set with $\mathsf{q} \in \square$, such that $\mathsf{diam}(\square) \leq \varepsilon \mathsf{d}_k(\mathsf{q}, \mathsf{P})$. Then, for any $\mathsf{u} \in \square$, we have*

$$(1 - \varepsilon)\mathsf{d}_k(\mathsf{q}, \mathsf{P}) \leq \mathsf{d}_k(\mathsf{u}, \mathsf{P}) \leq (1 + \varepsilon)\mathsf{d}_k(\mathsf{q}, \mathsf{P}) .$$

Proof: By Observation 4.2.1, we have

$$\mathsf{d}_k(\mathsf{q}, \mathsf{P}) \leq \mathsf{d}_k(\mathsf{u}, \mathsf{P}) + \|\mathsf{q} - \mathsf{u}\| \leq \mathsf{d}_k(\mathsf{u}, \mathsf{P}) + \mathsf{diam}(\square) \leq \mathsf{d}_k(\mathsf{u}, \mathsf{P}) + \varepsilon \mathsf{d}_k(\mathsf{q}, \mathsf{P}) .$$

The other direction follows by a symmetric argument. ■

Lemma 4.4.4 *If the smallest region $\square \in \mathcal{W}$ that contains q has diameter $\mathsf{diam}(\square) \leq \varepsilon \mathsf{d}_k(\mathsf{q}, \mathsf{P})/4$, then the algorithm returns a distance which is between $\mathsf{d}_k(\mathsf{q}, \mathsf{P})$ and $(1 + \varepsilon)\mathsf{d}_k(\mathsf{q}, \mathsf{P})$.*

Proof: Let $\square_{\mathsf{rep}}$ be the representative stored with the cell. Let α be the number returned by the algorithm. By Lemma 4.4.1 we have that $\mathsf{d}_k(\mathsf{q}, \mathsf{P}) \leq \alpha$. Since the algorithm returns the minimum of two numbers, one

of which is $\beta_k(\square_{\mathsf{rep}}) + \|\square_{\mathsf{rep}} - \mathsf{q}\|$, we have by Lemma 4.4.3,

$$\begin{aligned}
\alpha &\leq \beta_k(\square_{\mathsf{rep}}) + \|\square_{\mathsf{rep}} - \mathsf{q}\| \leq (1+\varepsilon/4)\mathsf{d}_k(\square_{\mathsf{rep}}, \mathsf{P}) + \|\square_{\mathsf{rep}} - \mathsf{q}\| \\
&\leq (1+\varepsilon/4)\Big(\mathsf{d}_k(\mathsf{q}, \mathsf{P}) + \mathsf{diam}(\square)\Big) + \mathsf{diam}(\square) \\
&\leq (1+\varepsilon/4)(\mathsf{d}_k(\mathsf{q}, \mathsf{P}) + \varepsilon \mathsf{d}_k(\mathsf{q}, \mathsf{P})/4) + \varepsilon \mathsf{d}_k(\mathsf{q}, \mathsf{P})/4 \\
&= (1+\varepsilon/4)^2 \mathsf{d}_k(\mathsf{q}, \mathsf{P}) + \varepsilon \mathsf{d}_k(\mathsf{q}, \mathsf{P})/4 \leq (1+\varepsilon)\mathsf{d}_k(\mathsf{q}, \mathsf{P}),
\end{aligned}$$

establishing the claim. ■

Definition 4.4.5 Consider a query point $\mathsf{q} \in \mathbb{R}^d$. The first quorum cluster $\mathsf{b}_i = \mathsf{ball}(\mathsf{c}_i, \mathsf{r}_i)$ that intersects $\mathsf{ball}(\mathsf{q}, \mathsf{d}_k(\mathsf{q}, \mathsf{P}))$ is the *anchor cluster* of q. The corresponding ***anchor point*** is $(\mathsf{c}_i, \mathsf{r}_i) \in \mathbb{R}^{d+1}$.

Lemma 4.4.6 *For any query point* q*, we have that*

(i) the anchor point (c, r) *is well defined,*

(ii) $\mathsf{r} \leq 2\mathsf{d}_k(\mathsf{q}, \mathsf{P})$*,*

(iii) for $\mathsf{b} = \mathsf{ball}(\mathsf{c}, \mathsf{r})$ *we have* $\mathsf{b} \cap \mathsf{ball}(\mathsf{q}, \mathsf{d}_k(\mathsf{q}, \mathsf{P})) \neq \emptyset$*, and*

(iv) $\|\mathsf{c} - \mathsf{q}\| \leq 3\mathsf{d}_k(\mathsf{q}, \mathsf{P})$*.*

Proof: Consider the k closest points to q in P. As $\mathsf{P} \subseteq \mathsf{b}_1 \cup \cdots \cup \mathsf{b}_{n/k}$ it must be that $\mathsf{ball}(\mathsf{q}, \mathsf{d}_k(\mathsf{q}, \mathsf{P}))$ intersects some b_i. Consider the first cluster $\mathsf{ball}(\mathsf{c}, \mathsf{r})$ in the quorum clustering that intersects $\mathsf{ball}(\mathsf{q}, \mathsf{d}_k(\mathsf{q}, \mathsf{P}))$. Then (c, r) is by definition the anchor point and we immediately have $\mathsf{ball}(\mathsf{c}, \mathsf{r}) \cap \mathsf{ball}(\mathsf{q}, \mathsf{d}_k(\mathsf{q}, \mathsf{P})) \neq \emptyset$. Claim (ii) is implied by the proof of Lemma 4.3.1. Finally, as for (iv), we have $\mathsf{r} \leq 2\mathsf{d}_k(\mathsf{q}, \mathsf{P})$ and the ball around q of radius $\mathsf{d}_k(\mathsf{q}, \mathsf{P})$ intersects $\mathsf{ball}(\mathsf{c}, \mathsf{r})$, thus implying that $\|\mathsf{c} - \mathsf{q}\| \leq \mathsf{d}_k(\mathsf{q}, \mathsf{P}) + \mathsf{r} \leq 3\mathsf{d}_k(\mathsf{q}, \mathsf{P})$. ■

Lemma 4.4.7 *Consider a query point* q*. If there is a cluster* $\mathsf{ball}(\mathsf{c}, \mathsf{r})$ *in the quorum clustering computed, such that* $\|\mathsf{c} - \mathsf{q}\| \leq 6\mathsf{d}_k(\mathsf{q}, \mathsf{P})$ *and* $\varepsilon \mathsf{d}_k(\mathsf{q}, \mathsf{P})/4 \leq \mathsf{r} \leq 6\mathsf{d}_k(\mathsf{q}, \mathsf{P})$*, then the output of the algorithm is correct.*

Proof: We have

$$\frac{32\mathsf{r}}{\varepsilon} \geq \frac{32(\varepsilon \mathsf{d}_k(\mathsf{q}, \mathsf{P})/4)}{\varepsilon} = 8\mathsf{d}_k(\mathsf{q}, \mathsf{P}) \geq \|\mathsf{c} - \mathsf{q}\|.$$

Thus, by construction, the expanded environ of the quorum cluster $\mathsf{ball}(\mathsf{c}, \mathsf{r})$ contains the query point, see Eq. (5.1). Let j be the smallest integer such that $2^j \mathsf{r} \geq \|\mathsf{c} - \mathsf{q}\|$. We have that, $2^j \mathsf{r} \leq \max(\mathsf{r}, 2\|\mathsf{c} - \mathsf{q}\|)$. As

such, if $\square$ is the smallest cell in $\mathcal{W}$ containing the query point q, then

$$\begin{aligned}\mathsf{diam}(\square) &\leq \frac{\varepsilon}{\zeta_1 d} 2^j \mathsf{r} \leq \frac{\varepsilon}{\zeta_1 d} \cdot \max(\mathsf{r}, 2\left\|\mathsf{c}-\mathsf{q}\right\|) \leq \frac{\varepsilon}{\zeta_1 d} \cdot \max\Big(6\mathsf{d}_k(\mathsf{q},\mathsf{P}), 12\mathsf{d}_k(\mathsf{q},\mathsf{P})\Big) \\ &\leq \frac{\varepsilon}{4d}\mathsf{d}_k(\mathsf{q},\mathsf{P}),\end{aligned}$$

by Eq. (5.1) and if $\zeta_1 \geq 48$. As such, $\mathsf{diam}(\square) \leq \varepsilon \mathsf{d}_k(\mathsf{q},\mathsf{P})/4$, and the claim follows by Lemma 4.4.3. ■

The general case

Lemma 4.4.8 *The data-structure constructed above returns* $(1+\varepsilon)$*-approximation to* $\mathsf{d}_k(\mathsf{q},\mathsf{P})$*, for any query point* q*.*

Proof: Consider the query point q and its anchor point (c,r). By Lemma 4.4.6, we have $\mathsf{r} \leq 2\mathsf{d}_k(\mathsf{q},\mathsf{P})$ and $\|\mathsf{c}-\mathsf{q}\| \leq 3\mathsf{d}_k(\mathsf{q},\mathsf{P})$. This implies that

$$\mathsf{d}(\mathsf{q}', \mathcal{B}') \leq \|(\mathsf{c},\mathsf{r}) - \mathsf{q}'\| \leq \|\mathsf{c}-\mathsf{q}\| + \mathsf{r} \leq 5\mathsf{d}_k(\mathsf{q},\mathsf{P}). \tag{4.3}$$

Let the returned point, which is a $(1+\varepsilon/8)$-ANN for q' in $\mathcal{B}'$, be $(\mathsf{c}_\mathsf{q},\mathsf{r}_\mathsf{q}) = \mathrm{nn}'(\square)$, where $\mathsf{q}' = (\mathsf{q},0)$. We have that $\big\|(\mathsf{c}_\mathsf{q},\mathsf{r}_\mathsf{q}) - \mathsf{q}'\big\| \leq (1+\varepsilon/8)\mathsf{d}(\mathsf{q}',\mathcal{B}') \leq 6\mathsf{d}_k(\mathsf{q},\mathsf{P})$. In particular, $\big\|\mathsf{c}_\mathsf{q} - \mathsf{q}\big\| \leq 6\mathsf{d}_k(\mathsf{q},\mathsf{P})$ and $\mathsf{r}_\mathsf{q} \leq 6\mathsf{d}_k(\mathsf{q},\mathsf{P})$.

Thus, if $\mathsf{r}_\mathsf{q} \geq \varepsilon \mathsf{d}_k(\mathsf{q},\mathsf{P})/4$ or $\mathsf{r} \geq \varepsilon \mathsf{d}_k(\mathsf{q},\mathsf{P})/4$ we are done, by Lemma 4.4.7. Otherwise, we have

$$\big\|\big(\mathsf{c}_\mathsf{q},\mathsf{r}_\mathsf{q}\big) - \mathsf{q}'\big\| \leq (1+\varepsilon/8)\,\|(\mathsf{c},\mathsf{r}) - \mathsf{q}'\|,$$

as $\big(\mathsf{c}_\mathsf{q},\mathsf{r}_\mathsf{q}\big)$ is a $(1+\varepsilon/8)$ approximation to $\mathsf{d}(\mathsf{q}',\mathcal{B}')$. As such,

$$\frac{\big\|(\mathsf{c}_\mathsf{q},\mathsf{r}_\mathsf{q}) - \mathsf{q}'\big\|}{1+\varepsilon/8} \leq \|(\mathsf{c},\mathsf{r}) - \mathsf{q}'\| \leq \|\mathsf{c}-\mathsf{q}\| + \mathsf{r}. \tag{4.4}$$

As $\mathsf{ball}(\mathsf{c},\mathsf{r}) \cap \mathsf{ball}(\mathsf{q},\mathsf{d}_k(\mathsf{q},\mathsf{P})) \neq \emptyset$ we have, by the triangle inequality, that

$$\|\mathsf{c}-\mathsf{q}\| - \mathsf{r} \leq \mathsf{d}_k(\mathsf{q},\mathsf{P}). \tag{4.5}$$

By Eq. (4.4) and Eq. (4.5) we have

$$\frac{\big\|(\mathsf{c}_\mathsf{q},\mathsf{r}_\mathsf{q}) - \mathsf{q}'\big\|}{1+\varepsilon/8} - 2\mathsf{r} \leq \|\mathsf{c}-\mathsf{q}\| - \mathsf{r} \leq \mathsf{d}_k(\mathsf{q},\mathsf{P}).$$

By the above and as $\max\bigl(\mathsf{r}, \mathsf{r}_{\mathsf{q}}\bigr) < \varepsilon \mathsf{d}_k(\mathsf{q}, \mathsf{P})/4$, we have

$$\begin{aligned}\bigl\|\mathsf{c}_{\mathsf{q}} - \mathsf{q}\bigr\| + \mathsf{r}_{\mathsf{q}} &\leq \bigl\|(\mathsf{c}_{\mathsf{q}}, \mathsf{r}_{\mathsf{q}}) - \mathsf{q}'\bigr\| + \mathsf{r}_{\mathsf{q}} \leq (1+\varepsilon/8)(\mathsf{d}_k(\mathsf{q}, \mathsf{P}) + 2\mathsf{r}) + \mathsf{r}_{\mathsf{q}} \\ &\leq (1+\varepsilon/8)(\mathsf{d}_k(\mathsf{q}, \mathsf{P}) + \varepsilon \mathsf{d}_k(\mathsf{q}, \mathsf{P})/2) + \varepsilon \mathsf{d}_k(\mathsf{q}, \mathsf{P})/4 \leq (1+\varepsilon)\mathsf{d}_k(\mathsf{q}, \mathsf{P}).\end{aligned}$$

Since the algorithm returns for q a value that is at most $\bigl\|\mathsf{c}_{\mathsf{q}} - \mathsf{q}\bigr\| + \mathsf{r}_{\mathsf{q}}$, the result is correct. ■

4.4.3 The result

Theorem 4.4.9 *Given a set P of n points in $\mathbb{R}^d$, a number $k \geq 1$ such that $k|n$, and $0 < \varepsilon$ sufficiently small, one can preprocess P, in $O\Bigl(n \log n + \frac{n}{k} C_\varepsilon \log n + \frac{n}{k} C'_\varepsilon\Bigr)$ time, where $C_\varepsilon = O\bigl(\varepsilon^{-d} \log \varepsilon^{-1}\bigr)$ and $C'_\varepsilon = O\bigl(\varepsilon^{-2d+1} \log \varepsilon^{-1}\bigr)$. The space used by the data-structure is $O(C_\varepsilon n/k)$. This data structure answers a $(1+\varepsilon, k)$-ANN query in $O\Bigl(\log \frac{n}{k\varepsilon}\Bigr)$ time. The data-structure also returns a point of P that is approximately the desired k-nearest neighbor.*

Proof: Computing the quorum clustering takes time $O(n \log n)$ by Lemma 5.4.6. Observe that $|\mathcal{X}| = O\bigl(\frac{n}{k\varepsilon^d} \log \frac{1}{\varepsilon}\bigr)$. From the construction of Arya and Malamatos [AM02], we have $|\mathcal{S}| = O\bigl(\frac{n}{k\varepsilon^d} \log \frac{1}{\varepsilon}\bigr)$ (note that since we clip the construction to a hyperplane, we get $1/\varepsilon^d$ in the bound and not $1/\varepsilon^{d+1}$). A careful implementation of this stage takes time $O\bigl(n \log n + |\mathcal{W}|\bigl(\log n + \frac{1}{\varepsilon^{d-1}}\bigr)\bigr)$. Overlaying the two compressed quadtrees representing them, takes linear time in their size, that is $O(|\mathcal{X}| + |\mathcal{S}|)$.

The most expensive step is to perform the $(1+\varepsilon/4, k)$-ANN query for each cell in the resulting decomposition of $\mathcal{W}$, see Eq. (5.2) (i.e., computing $\beta_k(\square_{\mathsf{rep}})$ for each cell $\square \in \mathcal{W}$). Using the data-structure of Section 4.6 (see Theorem 4.6.3) each query takes $O\bigl(\log n + 1/\varepsilon^{d-1}\bigr)$ time (alternatively, we could use the data-structure of Arya *et al.* [AMM05]), As such, this takes

$$O\left(n \log n + |\mathcal{W}|\left(\log n + \frac{1}{\varepsilon^{d-1}}\right)\right) = O\left(n \log n + \frac{n}{k\varepsilon^d} \log \frac{1}{\varepsilon} \log n + \frac{n}{k\varepsilon^{2d-1}} \log \frac{1}{\varepsilon}\right)$$

time, and this bounds the overall construction time.

The query algorithm is a point location query followed by an $O(1)$ time computation and takes time $O\bigl(\log\bigl(\frac{n}{k\varepsilon}\bigr)\bigr)$.

Finally, one needs to argue that the returned point of P is indeed the desired approximate k-nearest neighbor. This follows by arguing in a similar fashion to the correctness proof; the distance to the returned point is a $(1+\varepsilon)$-approximation to the kth-nearest neighbor distance. We omit the tedious but straightforward details. ■

Using a single point for each AVD cell

The AVD generated can be viewed as storing two points in each cell $\square$ of the AVD. These two points are in $\mathbb{R}^{d+1}$, and for a cell $\square$, they are

(i) the point $\mathrm{nn}'(\square) \in \mathcal{B}'$, and

(ii) the point $(\square_{\mathsf{rep}}, \beta_k(\square_{\mathsf{rep}}))$.

The algorithm for $\mathsf{d}_k(\mathsf{q}, \mathsf{P})$ can be viewed as computing the nearest neighbor of $(\mathsf{q}, 0)$ to one of the above two points using the $\|\cdot\|_\oplus$ norm to define the distance. Using standard AVD algorithms we can subdivide each such cell $\square$ into $O\bigl(1/\varepsilon^d \log \varepsilon^{-1}\bigr)$ cells to answer this query approximately. By using this finer subdivision we can have a single point inside each cell for which the closest distance is the approximation to $\mathsf{d}_k(\mathsf{q}, \mathsf{P})$. This incurs an increase by a factor of $O\bigl(1/\varepsilon^d \log \varepsilon^{-1}\bigr)$ in the number of cells.

4.4.4 A generalization – weighted version of k ANN

We consider a generalization of the $(1+\varepsilon, k)$-ANN problem. Specifically, we are given a set of points $\mathsf{P} \subseteq \mathbb{R}^d$, a weight $w_{\mathsf{p}} \geq 0$ for each $\mathsf{p} \in \mathsf{P}$, and a number $\varepsilon > 0$. Given a query q and weight $\tau \geq 0$, its **τ-NN** distance to P, is the minimum r such that the closed ball $\mathsf{ball}(\mathsf{q}, r)$ contains points of P of total weight at least τ. Formally, the τ-NN distance for q is

$$\mathsf{d}_\tau(\mathsf{q}, \mathsf{P}) = \min \left\{ r \;\middle|\; w\Bigl(\mathsf{ball}(\mathsf{q}, r) \cap \mathsf{P}\Bigr) \geq \tau \right\},$$

where $w(X) = \sum_{x \in X} w_x$. A ***$(1+\varepsilon)$-approximate τ-NN distance*** is a distance ℓ, such that $(1-\varepsilon)\mathsf{d}_\tau(\mathsf{q}, \mathsf{P}) \leq \ell \leq (1+\varepsilon)\mathsf{d}_\tau(\mathsf{q}, \mathsf{P})$ and a ***$(1+\varepsilon)$-approximate τ-NN*** is a point of P that realizes such a distance. The ***$(1+\varepsilon, \tau)$-ANN problem*** is to preprocess P, such that a $(1+\varepsilon)$-approximate τ-NN can be computed efficiently for any query point q.

The $(1+\varepsilon, k)$-ANN problem is the special case $w_{\mathsf{p}} = 1$ for all $\mathsf{p} \in \mathsf{P}$ and $\tau = k$. Clearly, the function $\mathsf{d}_\tau(\cdot, \mathsf{P})$ is also a 1-Lipschitz function of its argument. If we are given τ at the time of preprocessing, it can be verified that the 1-Lipschitz property is enough to guarantee correctness of the AVD construction for the $(1+\varepsilon, k)$-ANN problem. However, we need to compute a τ quorum clustering, where now each quorum cluster has weight at least τ. A slight modification of the algorithm in Lemma 5.4.6 allows this. Moreover, for the preprocessing step which requires us to solve the $(1+\varepsilon, \tau)$-ANN problem for the representative points, one can use the algorithm of Section 4.6.3. We get the following result,

Theorem 4.4.10 *Given a set of n weighted points P in $\mathbb{R}^d$, a number $\tau > 0$ and $0 < \varepsilon$ sufficiently small, one can preprocess P in $O\left(n \log n + \dfrac{w(\mathsf{P})}{\tau} C_\varepsilon \log n + \dfrac{w(\mathsf{P})}{\tau} C'_\varepsilon\right)$ time, where $C_\varepsilon = O\bigl(\varepsilon^{-d} \log \varepsilon^{-1}\bigr)$ and*

$C'_\varepsilon = O\left(\varepsilon^{-2d+1}\log\varepsilon^{-1}\right)$ *and* $w(\mathsf{P}) = \sum_{\mathsf{p}\in\mathsf{P}} w_(\mathsf{p})$. *The space used by the data-structure is* $O(C_\varepsilon w(\mathsf{P})/\tau)$. *This data structure answers a* $(1+\varepsilon,\tau)$-*ANN query in* $O\left(\log \frac{w(\mathsf{P})}{\tau\varepsilon}\right)$ *time. The data-structure also returns a point of* P *that is a* $(1+\varepsilon)$*-approximation to the* τ*-nearest neighbor of the query point.*

4.5 Density estimation

Given a point set $\mathsf{P} \subseteq \mathbb{R}^d$, and a query point $\mathsf{q} \in \mathbb{R}^d$, consider the point $\mathsf{v}(\mathsf{q}) = (\mathsf{d}_1(\mathsf{q},\mathsf{P}), \ldots, \mathsf{d}_n(\mathsf{q},\mathsf{P}))$. This is a point in $\mathbb{R}^n$, and several problems in Computational Geometry can be viewed as computing some interesting function of $\mathsf{v}(\mathsf{q})$. For example, one could view the nearest neighbor distance as the function that returns the first coordinate of $\mathsf{v}(\mathsf{q})$. Another motivating example is a geometric version of discrete density measures from Guibas *et al.* [GMM11]. In their problem one is interested in computing $g_k(\mathsf{q}) = \sum_{i=1}^{k} \mathsf{d}_i(\mathsf{q},\mathsf{P})$. In this section, we show that a broad class of functions (that include g_k), can be approximated to within $(1\pm\varepsilon)$, by a data structure requiring space $\widetilde{O}(n/k)$.

4.5.1 Performing point-location in several quadtrees simultaneously

Lemma 4.5.1 *Consider a rooted tree* T *with* m *nodes, where the nodes are colored by* I *colors (a node might have several colors). Assume that there are* $O(m)$ *pairs of such* $(node, color)$ *associations. One can preprocess the tree in* $O(m)$ *time and space, such that given a query leaf* v *of* T*, one can report the nodes* $v_1, \ldots, v_I$ *in* $O(I)$ *time. Here,* v_i *is the lowest node in the tree along the path from the root to* v *that is colored with color* i*.*

Proof: We start with the naive solution – perform a **DFS** on T, and keep an array $\mathcal{X}$ of I entries storing the latest node of each color encountered so far along the path from the root to the current node. Storing a snapshot of this array $\mathcal{X}$ at each node would require $O(mI)$ space. But then one can answer a query in $O(I)$ time. As such, the challenge is to reduce the required space.

To this end, interpret the **DFS** to be a Eulerian traversal of the tree. The traversal has length $2m-2$, and every edge traveled contains updates to the array $\mathcal{X}$. Indeed, if the **DFS** traverses down from a node u to a child node w, the updates would be updating all the colors that are stored in w, to indicate that w is the lowest node for these colors. Similarly, if the **DFS** goes up from w to u, we restore all the colors stored in w to their value just before the **DFS** visited w. Now, the **DFS** traversal of T becomes a list of $O(m)$ updates. Each update is still an $O(I)$ operation. This is however a technicality, and can be resolved as follows. For each edge traveled we store the updates for all colors separately, each update being for a single color. Also each update entry stores the current node, i.e., the destination vertex of the edge traveled. The total length

of the update list is still $O(m)$, as follows from a simple charging argument, and the assumption about the number of (node, color) pairs. We simply charge each restore to its corresponding "forward going" update, and the number of forward going updates is exactly equal to the number of (node, color) pairs. For each leaf we store its last location in this list of updates.

So, let L be this list of updates. At each kth update, for $k = tI$ for some integer t, store a snapshot of the array of colors as updated if we scan the list from the beginning till this point. Along with this we store the node at this point and auxiliary information allowing us to compute the next update i.e., if the snapshot stored is between all updates at this node. Clearly, all these snapshots can be computed in $O(m)$ time, and require $O((m/I)I) = O(m)$ space.

Now, given a query leaf v, we go to its location in the list L, and jump back to the last snapshot stored. We copy this snapshot, and then scan the list from the snapshot till the location for v. This would require re-doing at most $O(I)$ updates, and can be done in $O(I)$ time overall. ■

Lemma 4.5.2 *Given I compressed quadtrees $\mathcal{D}_1, \ldots, \mathcal{D}_I$ of total size m in $\mathbb{R}^d$, one can preprocess them in $O(m \log I)$ time, using $O(m)$ space, such that given a query point $\mathtt{q}$, one can perform point-location queries in all I quadtrees, simultaneously for $\mathtt{q}$, in $O(\log m + I)$ time.*

Proof: Overlay all these compressed quadtrees together. Overlaying I quadtrees is equivalent to merging I sorted lists [Har11] and can be done in $O(m \log I)$ time. Let $\mathcal{D}$ denote the resulting compressed quadtree. Note that any node of $\mathcal{D}_i$, for $i = 1, \ldots, I$, must be a node in $\mathcal{D}$.

Given a query point $\mathtt{q}$, we need to extract the I nodes in the original quadtrees $\mathcal{D}_i$, for $i = 1, \ldots, I$, that contain the query point (these nodes can be compressed nodes). So, let $\square$ be the leaf node of $\mathcal{D}$ containing the query point $\mathtt{q}$. Consider the path π from the root to the node $\square$. We are interested in the lowest node of π that belongs to $\mathcal{D}_i$, for $i = 1, \ldots, I$. To this end, color all the nodes of $\mathcal{D}_i$ that appear in $\mathcal{D}$, by color i, for $i = 1, \ldots, I$. Now, we build the data-structure of Lemma 4.5.1 for $\mathcal{D}$. We can use this data-structure to answer the desired query in $O(I)$ time. ■

4.5.2 Slowly growing functions

Definition 4.5.3 A monotonic increasing function $f : \mathbb{R}^+ \to \mathbb{R}$ is ***slowly growing*** if there is a constant $c > 0$, such that for ε sufficiently small, we have $(1-\varepsilon)f(x) \leq f((1-\varepsilon/c)x) \leq f((1+\varepsilon/c)x) \leq (1+\varepsilon)f(x)$, for all $x \in \mathbb{R}^+$. The constant c is the ***growth constant*** of f. The family of slowly growing functions is denoted by $\mathcal{F}_{\mathsf{sg}}$.

$\mathcal{F}_{\mathsf{sg}}$	The class of slowly growing functions, see Definition 4.5.3.
f	A function in $\mathcal{F}_{\mathsf{sg}}$ or a monotonic increasing function from $\mathbb{R}$ to $\mathbb{R}^+$.
$F(\mathsf{q})$	$\sum_{i=1}^{k} f\Big(\mathsf{d}_i(\mathsf{q},\mathsf{P})\Big)$
$F_1(\mathsf{q})$	$\sum_{i=\lceil k\varepsilon/8\rceil}^{k} f(\mathsf{d}_i(\mathsf{q},\mathsf{P}))$
$\mathcal{I}$	$\mathcal{I} \subseteq \Big\{\lceil k\varepsilon/8\rceil, \ldots, k\Big\}$, is a coreset, see Lemma 4.5.5.
$w_i, i \in \mathcal{I}$	$w_i \geq 0$ are associated weights for coreset elements.
$F_2(\mathsf{q})$	$\sum_{i\in\mathcal{I}} w_i f(\mathsf{d}_i(\mathsf{q},\mathsf{P}))$

Figure 4.3: Notations used.

Clearly, $\mathcal{F}_{\mathsf{sg}}$ includes polynomial functions, but it does not include, for example, the function e^x. We assume that given x, one can evaluate the function $f(x)$ in constant time. In this section, using the AVD construction of Section 4.4, we show how to approximate any function $F(\cdot)$ that can be expressed as

$$F(\mathsf{q}) = \sum_{i=1}^{k} f\Big(\mathsf{d}_i(\mathsf{q},\mathsf{P})\Big),$$

where $f \in \mathcal{F}_{\mathsf{sg}}$. See Figure 4.3 for a summary of the notations used in this section.

Lemma 4.5.4 *Let $f : \mathbb{R} \to \mathbb{R}^+$ be a monotonic increasing function. Now, let $F_1(\mathsf{q}) = \sum_{i=\lceil k\varepsilon/8\rceil}^{k} f(\mathsf{d}_i(\mathsf{q},\mathsf{P}))$. Then, for any query point q, we have that $F_1(\mathsf{q}) \leq F(\mathsf{q}) \leq (1+\varepsilon/4)F_1(\mathsf{q})$, where $F(\mathsf{q}) = \sum_{i=1}^{k} f\Big(\mathsf{d}_i(\mathsf{q},\mathsf{P})\Big)$.*

Proof: The first inequality is obvious. As for the second inequality, observe that $\mathsf{d}_i(\mathsf{q},\mathsf{P})$ is a monotonically increasing function of i, and so is $f(\mathsf{d}_i(\mathsf{q},\mathsf{P}))$. We are dropping the smallest $k(\varepsilon/8)$ terms of the summation $F(\mathsf{q})$ that is made out of k terms. As such, the claim follows. ■

The next lemma exploits a coreset construction, so that we have to evaluate only few terms of the summation.

Lemma 4.5.5 *Let $f : \mathbb{R} \to \mathbb{R}^+$ be a monotonic increasing function. There is a set of indices $\mathcal{I} \subseteq \Big\{\lceil k\varepsilon/8\rceil, \ldots, k\Big\}$, and integer weights $w_i \geq 0$, for $i \in \mathcal{I}$, such that:*

(A) $|\mathcal{I}| = O\Big(\frac{\log k}{\varepsilon}\Big)$.

(B) For any query point q, we have that $F_2(\mathsf{q}) = \sum_{i\in\mathcal{I}} w_i f(\mathsf{d}_i(\mathsf{q},\mathsf{P}))$ is a good estimate for $F_1(\mathsf{q})$; that is,

$$(1-\varepsilon/4)F_2(\mathsf{q}) \leq F_1(\mathsf{q}) \leq (1+\varepsilon/4)F_2(\mathsf{q}), \text{ where } F_1(\mathsf{q}) = \sum_{i=\lceil k\varepsilon/8\rceil}^{k} f(\mathsf{d}_i(\mathsf{q},\mathsf{P})).$$

Furthermore, the set $\mathcal{I}$ can be computed in $O(|\mathcal{I}|)$ time.

Proof: Given a query point q consider the function $g_{\mathsf{q}} : \{1, 2, \ldots, n\} \to \mathbb{R}^+$ defined as $g_{\mathsf{q}}(i) = f\Big(\mathsf{d}_i(\mathsf{q},\mathsf{P})\Big)$. Clearly, since $f \in \mathcal{F}_{\mathsf{sg}}$, it follows that g_{q} is a monotonic increasing function. The existence of $\mathcal{I}$ follows

from Lemma 3.2 in Har-Peled's paper [Har06], as applied to $(1 \pm \varepsilon/4)$-approximating the function $F_1(\mathsf{q}) = \sum_{i=\lceil k\varepsilon/8 \rceil}^{k} f(\mathsf{d}_i(\mathsf{q}, \mathsf{P}))$; that is, $(1-\varepsilon/4)F_2(\mathsf{q}) \leq F_1(\mathsf{q}) \leq (1+\varepsilon/4)F_2(\mathsf{q})$. ■

4.5.3 The data-structure

We are given a set of n points $\mathsf{P} \subseteq \mathbb{R}^d$, a function $f \in \mathcal{F}_{\mathsf{sg}}$, an integer k with $1 \leq k \leq n$, and $\varepsilon > 0$ sufficiently small. We describe how to build a data-structure to approximate $F(\mathsf{q}) = \sum_{i=1}^{k} f\Big(\mathsf{d}_i(\mathsf{q}, \mathsf{P})\Big)$.

Construction

In the following, let $\alpha = 4c$, where c is the growth constant of f (see Definition 4.5.3). Consider the coreset $\mathcal{I}$ from Lemma 4.5.5. For each $i \in \mathcal{I}$ we compute, using Theorem 4.4.9, a data-structure (i.e., a compressed quadtree) $\mathcal{D}_i$ for answering $(1+\varepsilon/\alpha, i)$-ANN queries for P. We then overlay all these quadtrees into a single quadtree, using Lemma 4.5.2.

Answering a Query. Given a query point q, perform a simultaneous point-location query in $\mathcal{D}_1, \ldots, \mathcal{D}_I$, by using $\mathcal{D}$, as described in Lemma 4.5.2. This results in a $(1+\varepsilon/\alpha)$ approximation z_i to $\mathsf{d}_i(\mathsf{q}, \mathsf{P})$, for $i \in \mathcal{I}$, and takes $O(\log m + I)$ time, where m is the size of $\mathcal{D}$, and $I = |\mathcal{I}|$. We return $\xi = \sum_{i \in \mathcal{I}} w_i f(z_i)$, where w_i is the weight associated with the index i of the coreset of Lemma 4.5.5.

Bounding the quality of approximation. We only prove the upper bound on ξ. The proof for the lower bound is similar. As the z_i are $(1 \pm \varepsilon/\alpha)$ approximations to $\mathsf{d}_i(\mathsf{q}, \mathsf{P})$ we have, $(1-\varepsilon/\alpha)z_i \leq \mathsf{d}_i(\mathsf{q}, \mathsf{P})$, for $i \in \mathcal{I}$, and it follows from definitions that,

$$(1-\varepsilon/4)w_i f(z_i) \leq w_i f\Big((1-\varepsilon/\alpha)z_i\Big) \leq w_i f(\mathsf{d}_i(\mathsf{q}, \mathsf{P}))\,,$$

for $i \in \mathcal{I}$. Therefore,

$$(1-\varepsilon/4)\xi = (1-\varepsilon/4)\sum_{i \in \mathcal{I}} w_i f(z_i) \leq \sum_{i \in \mathcal{I}} w_i f(\mathsf{d}_i(\mathsf{q}, \mathsf{P})) = F_2(\mathsf{q}). \tag{4.6}$$

Using Eq. (4.6) and Lemma 4.5.5 it follows that,

$$(1-\varepsilon/4)^2 \xi \leq (1-\varepsilon/4)F_2(\mathsf{q}) \leq F_1(\mathsf{q}). \tag{4.7}$$

Finally, by Eq. (4.7) and Lemma 4.5.4 we have,

$$(1-\varepsilon/4)^2\xi \leq F_1(\mathsf{q}) \leq F(\mathsf{q}).$$

Therefore we have, $(1-\varepsilon)\xi \leq (1-\varepsilon/4)^2\xi \leq F(\mathsf{q})$, as desired.

Preprocessing space and time analysis. We have that $I = |\mathcal{I}| = O\bigl(\varepsilon^{-1}\log k\bigr)$. Let $C_x = O\bigl(x^{-d}\log x^{-1}\bigr)$. By Theorem 4.4.9 the total size of all the $\mathcal{D}_i$s (and thus the size of the resulting data-structure) is

$$S = \sum_{i\in\mathcal{I}} O\Bigl(C_{\varepsilon/\alpha}\frac{n}{i}\Bigr) = O\biggl(C_{\varepsilon/\alpha}\frac{n\log k}{k\varepsilon^2}\biggr). \tag{4.8}$$

Indeed, the maximum of the terms involving n/i is $O(n/k\varepsilon)$ and $I = O\bigl(\varepsilon^{-1}\log k\bigr)$. By Theorem 4.4.9 the total time taken to construct all the $\mathcal{D}_i$ is

$$\sum_{i\in\mathcal{I}} O\Bigl(n\log n + \frac{n}{i}C_{\varepsilon/\alpha}\log n + \frac{n}{i}C'_{\varepsilon/\alpha}\Bigr) = O\biggl(\frac{n\log n\log k}{\varepsilon} + \frac{n\log n\log k}{k\varepsilon^2}C_{\varepsilon/\alpha} + \frac{n\log k}{k\varepsilon^2}C'_{\varepsilon/\alpha}\biggr),$$

where $C'_x = O\bigl(x^{-2d+1}\log x^{-1}\bigr)$. The time to construct the final quadtree is $O(S\log I)$, but this is subsumed by the construction time above.

The result

Summarizing the above, we get the following result.

Theorem 4.5.6 *Let P be a set of n points in $\mathbb{R}^d$. Given any slowly growing, monotonic increasing function f (i.e $f \in \mathcal{F}_{\mathsf{sg}}$, see Definition 4.5.3), an integer k with $1 \leq k \leq n$, and $\varepsilon \in (0,1)$, one can build a data-structure to approximate $F(\cdot)$. Specifically, we have:*

(A) *The construction time is $O(C_1 n\log n\log k)$, where $C_1 = O\bigl(\varepsilon^{-2d-1}\log\varepsilon^{-1}\bigr)$.*

(B) *The space used is $O\Bigl(C_2\frac{n}{k}\log k\Bigr)$, where $C_2 = O\bigl(\varepsilon^{-d-2}\log\varepsilon^{-1}\bigr)$.*

(C) *For any query point q, the data-structure computes a number ξ, such that $(1-\varepsilon)\xi \leq F(\mathsf{q}) \leq (1+\varepsilon)\xi$, where $F(\mathsf{q}) = \sum_{i=1}^{k} f(\mathsf{d}_i(\mathsf{q},\mathsf{P}))$.*

(D) *The query time is $O\biggl(\log n + \frac{\log k}{\varepsilon}\biggr)$.*

(The O notation here hides constants that depend on f.)

4.6 ANN queries where k and ε are part of the query

Given a set P of n points in $\mathbb{R}^d$, we present a data-structure for answering $(1+\varepsilon, k)$-ANN queries, in time $O\big(\log n + 1/\varepsilon^{d-1}\big)$. Here k and ε are not known during the preprocessing stage, but are specified during query time. In particular, different queries can use different values of k and ε. Unlike our main result, this data-structure requires linear space, and the amount of space used is independent of k and ε. Previous data-structures required knowing ε in advance [AMM05].

4.6.1 Rough approximation

Observe that a fast constant approximation to $\mathsf{d}_k(\mathsf{q}, \mathsf{P})$ is implied by Theorem 4.3.2 if k is known in advance. We describe a polynomial approximation when k is not available during preprocessing. We sketch the main ideas; our argument closely follows the exposition in Har-Peled's book [Har11].

Lemma 4.6.1 *Given a set P of n points in $\mathbb{R}^d$, one can preprocess it, in $O(n \log n)$ time, such that given any query point q and k with $1 \le k \le n$, one can find, in $O(\log n)$ time, a number R satisfying $\mathsf{d}_k(\mathsf{q}, \mathsf{P}) \le R \le n^c \mathsf{d}_k(\mathsf{q}, \mathsf{P})$. The result is correct with high probability, i.e., at least $1 - 1/n^{c-2}$, where c is an arbitrary constant.*

Proof: By an appropriate scaling and translation ensure that $\mathsf{P} \subseteq [1/2, 3/4]^d$. Consider a compressed quadtree decomposition $\mathcal{T}$ of $\mathbf{b} + [0,1]^d$ for P, whose shift $\mathbf{b}$ is a random vector in $[0, 1/2]^d$. By a bottom-up traversal, compute, for each node v of $\mathcal{T}$, the axis parallel bounding box B_v of the subset of P stored in its subtree, and the number of those points.

Given a query point $\mathsf{q} \in [1/2, 3/4]^d$, locate the lowest node ν of $\mathcal{T}$ whose region contains q (this takes $O(\log n)$ time, see [Har11]). By performing a binary search on the root to ν path locate the lowest node ν_k whose subtree contains k or more points from P. The algorithm returns R, the distance of the query point to the furthest point of B_{ν_k}, as the approximate distance.

To see that the quality of approximation is as claimed, consider the ball b centered at q with radius $r = \mathsf{d}_k(\mathsf{q}, \mathsf{P})$. Next, consider the smallest canonical grid having side length $\alpha \ge n^{c-1} r$ (thus, $\alpha \le 2n^{c-1} r$). Randomly translating this grid, we have with probability $\ge 1 - 2rd/\alpha \ge 1 - 1/n^{c-2}$, that the ball b is contained inside a canonical cell $\square$ of this grid. This implies that the diameter of B_{ν_k} is bounded by $\sqrt{d}\alpha$, Indeed, if the cell of ν_k is contained in $\square$, then this clearly holds. Otherwise, if $\square$ is contained in the cell ν_k, then ν_k must be a compressed node, the inner portion of its cell is contained in $\square$, and the outer portion of the cell can not contain any point of P. As such, the claim holds.

Moreover, for the returned distance R, we have that

$$r = \mathsf{d}_k(\mathsf{q}, \mathsf{P}) \leq R \leq \mathsf{diam}(B_{\nu_k}) + r \leq \sqrt{d}\alpha + r \leq \sqrt{d} 2 n^{c-1} r + r \leq n^c r.$$

■

An alternative to the argument used in Lemma 4.6.1, is to use two shifted quadtrees, and return the smaller distance returned by the two trees. It is not hard to argue that in expectation the returned distance is an $O(1)$-approximation to the desired distance (which then implies the desired result via Markov's inequality). One can also derandomize the shifted quadtrees and use $d+1$ quadtrees instead [Har11].

We next show how to refine this approximation.

Lemma 4.6.2 *Given a set P of n points in $\mathbb{R}^d$, one can preprocess it in $O(n \log n)$ time, so that given a query point q, one can output a number β satisfying, $\mathsf{d}_k(\mathsf{q}, \mathsf{P}) \leq \beta \leq (1+\varepsilon)\mathsf{d}_k(\mathsf{q}, \mathsf{P})$, in $O\big(\log n + 1/\varepsilon^{d-1}\big)$ time. Furthermore, one can return a point $\mathsf{p} \in \mathsf{P}$ such that $(1-\varepsilon)\mathsf{d}_k(\mathsf{q}, \mathsf{P}) \leq \|\mathsf{p} - \mathsf{q}\| \leq (1+\varepsilon)\mathsf{d}_k(\mathsf{q}, \mathsf{P})$.*

Proof: Assume that $\mathsf{P} \cup \{\mathsf{q}\} \subseteq [1/2, 1/2 + 1/n]^d$. The algorithm of Lemma 4.6.1 returns the distance R between q and some point of P; as such we have, $\mathsf{d}_k(\mathsf{q}, \mathsf{P}) \leq R \leq n^{O(1)}\mathsf{d}_k(\mathsf{q}, \mathsf{P}) \leq \mathsf{diam}(\mathsf{P} \cup \{\mathsf{q}\}) \leq d/n$. We start with a compressed quadtree for P having $U = [0,1]^d$ as the root. We look at the set of canonical cells X_0 with side length at least R, that intersect the ball $\mathsf{ball}(\mathsf{q}, R)$. Clearly, the kth nearest neighbor of q lies in this set of cubes. The set X_0 can be computed in $O(|X_0| \log n)$ time using cell queries [Har11].

For each node v in the compressed quadtree there is a ***level*** associated with it. This is $\mathrm{lvl}(v) = \log_2 \mathrm{sidelength}(\square_v)$. The root has level 0 and it decreases as we go down the compressed quadtree. Intuitively, $-\mathrm{lvl}(v)$ is the depth of the node if it was a node in a regular quadtree.

We maintain a queue of such canonical grid cells. Each step in the search consists of replacing cells in the current level with their children in the quadtree, and deciding if we want to descend a level. In the ith iteration, we replace every node of X_{i-1} by its children in the next level, and put them into the set X_i.

We then update our estimate of $\mathsf{d}_k(\mathsf{q}, \mathsf{P})$. Initially, we set $I_0 = [\mathsf{l}_0, \mathsf{h}_0] = [0, R]$. For every node $v \in X_i$, we compute the closest and furthest point of its cube (that is the cell of this node) from the query point (this can be done in $O(1)$ time). This specifies a collection of intervals I_v one for each node $v \in X_i$. Let n_v denote the number of points stored in the subtree of v. For a real number x, let $L(x), M(x), R(x)$ denote the total number of points in the intervals, that are to the left of x, contains x, and are to the right of x, respectively. Using median selection, one can compute in linear time (in the number of nodes of X_i) the minimum x such that $L(x) \geq k$. Let this value be h_i. Similarly, in linear time, compute the minimum x such that $L(x) + M(x) \geq k$, and let this value be l_i. Clearly, the desired distance is in the interval $I_i = [\mathsf{l}_i, \mathsf{h}_i]$.

The algorithm now iterates over $v \in X_i$. If I_v is strictly to the left of l_i, v is discarded (it is too close to the query and can not contain the kth nearest neighbor), setting $k \leftarrow k - n_v$. Similarly, if I_v is to the right of h_i it can be thrown away. The algorithm then moves to the next iteration.

The algorithm stops as soon as the diameter of all the cells of X_i is smaller than $(\varepsilon/8)\mathsf{l}_i$. A representative point is chosen from each node of X_i (each node of the quadtree has an arbitrary representative point precomputed for it out of the subset of points stored in its subtree), and the furthest such point is returned as the $(1+\varepsilon)$- approximate k nearest neighbor. To see that the returned answer is indeed correct, observe that $\mathsf{l}_i \leq \mathsf{d}_k(\mathsf{q},\mathsf{P}) \leq \mathsf{h}_i$ and $\mathsf{h}_i - \mathsf{l}_i \leq (\varepsilon/8)\mathsf{l}_i$, which implies the claim. The distance of the returned point from q is in the interval $[\alpha,\beta]$, where $\alpha = \mathsf{l}_i - (\varepsilon/8)\mathsf{l}_i$ and $\beta = \mathsf{h}_i \leq \mathsf{l}_i + (\varepsilon/8)\mathsf{l}_i \leq (1+\varepsilon/2)(1-\varepsilon/8)\mathsf{l}_i \leq (1+\varepsilon/2)\alpha$. This interval also contains $\mathsf{d}_k(\mathsf{q},\mathsf{P})$. As such, β is indeed the required approximation.

Since we are working with compressed quadtrees, a child node might be many levels below the level of its parent. In particular, if a node's level is below the current level, we freeze it and just move it on the set of the next level. We replace it by its children only when its level has been reached.

The running time is clearly $O(|X_0|\log n + \sum_i |X_i|)$. Let Δ_i be the diameter of the cells in the level being handled in the ith iteration. Clearly, we have that $\mathsf{h}_i \leq \mathsf{l}_i + \Delta_i$. All the cells of X_i that survive must intersect the ring with inner and outer radii l_i and h_i respectively, around q. By a simple packing argument, $|X_i| \leq n_i = O\big((\mathsf{l}_i/\Delta_i + 1)^{d-1}\big)$. As long as $\Delta_i \geq \mathsf{d}_k(\mathsf{q},\mathsf{P})$, we have that $n_i = O(1)$, as $\mathsf{l}_i \leq \mathsf{d}_k(\mathsf{q},\mathsf{P})$. This clearly holds for the first $O(\log n)$ iterations. It can be verified that once this no longer holds, the algorithm performs at most $\lceil \log_2(1/\varepsilon) \rceil + O(1)$ additional iterations, as then $\Delta_i \leq (\varepsilon/16)\mathsf{d}_k(\mathsf{q},\mathsf{P})$ and the algorithm stops. Clearly, the n_is in this range can grow exponentially, but the last one is $O(1/\varepsilon^{d-1})$. This implies that $\sum_i |X_i| = O\big(\log n + 1/\varepsilon^{d-1}\big)$, as desired. ∎

4.6.2 The result

Theorem 4.6.3 *Given a set P of n points in $\mathbb{R}^d$, one can preprocess them in $O(n \log n)$ time, into a data structure of size $O(n)$, such that given a query point q, an integer k with $1 \leq k \leq n$ and $\varepsilon > 0$ one can compute, in $O\big(\log n + 1/\varepsilon^{d-1}\big)$ time, a number β such that $\mathsf{d}_k(\mathsf{q},\mathsf{P}) \leq \beta \leq (1+\varepsilon)\mathsf{d}_k(\mathsf{q},\mathsf{P})$. The data-structure also returns a point $\mathsf{p} \in \mathsf{P}$ such that $(1-\varepsilon)\mathsf{d}_k(\mathsf{q},\mathsf{P}) \leq \|\mathsf{p}-\mathsf{q}\| \leq (1+\varepsilon)\mathsf{d}_k(\mathsf{q},\mathsf{P})$.*

4.6.3 Weighted version of $(1+\varepsilon,k)$-ANN

We now consider the weighted version of the $(1+\varepsilon,k)$-ANN problem as defined in Section 4.4.4. Knowledge of the threshold weight τ is not required at the time of preprocessing. By a straightforward adaptation of the arguments in this section we get the following.

Theorem 4.6.4 *Given a set* P *of* n *weighted points in* $\mathbb{R}^d$ *one can preprocess them, in* $O(n \log n)$ *time, into a data structure of size* $O(n)$*, such that one can efficiently answer* $(1+\varepsilon, \tau)$-*ANN queries. Here a query is made out of (i) a query point* q*, (ii) a weight* $\tau \geq 0$*, and (iii) an approximation parameter* $\varepsilon > 0$*. Specifically, for such a query, one can compute, in* $O\big(\log n + 1/\varepsilon^{d-1}\big)$ *time, a number* β *such that* $(1-\varepsilon)\mathsf{d}_\tau(\mathsf{q}, \mathsf{P}) \leq \beta \leq (1+\varepsilon)\mathsf{d}_\tau(\mathsf{q}, \mathsf{P})$*. The data-structure also returns a point* $\mathsf{p} \in \mathsf{P}$ *such that* $(1-\varepsilon)\mathsf{d}_\tau(\mathsf{q}, \mathsf{P}) \leq \|\mathsf{p} - \mathsf{q}\| \leq (1+\varepsilon)\mathsf{d}_\tau(\mathsf{q}, \mathsf{P})$.

4.7 Conclusions

In this chapter, we presented a data-structure for answering $(1+\varepsilon, k)$-ANN queries in $\mathbb{R}^d$ where d is a constant. Our data-structure has the surprising property that the space required is $\widetilde{O}(n/k)$. One can verify that up to noise this is the best one can do for this problem. This data-structure also suggests a natural way of compressing geometric data, such that the resulting sketch can be used to answer meaningful proximity queries on the original data. We then used this data-structure to answer various proximity queries using roughly the same space and query time. We also presented a data-structure for answering $(1+\varepsilon, k)$-ANN queries where both k and ε are specified during query time. This data-structure is simple and practical.

There are many interesting questions for further research.

(A) In the vein of the authors recent work [HK11], one can verify that our results extends in a natural way to metrics of low doubling dimensions ([HK11] describes what an approximate Voronoi diagram is for doubling metrics). It also seems believable that the result would extend to the problem where the data is high dimensional but the queries arrive from a low dimensional manifold.

(B) It is natural to ask what one can do for this problem in high dimensional Euclidean space. In particular, can one get query time close to the one required for approximate nearest neighbor [IM98, HIM12]. Of particular interest is getting a query time that is sublinear in k and n while having subquadratic space and preprocessing time.

(C) The dependency on ε in our data-structures may not be optimal. One can probably get space/time tradeoffs, as done by Arya *et al.* [AMM09].

Chapter 5

Sublinear Space Data-Structures for k-ANN on Balls

5.1 Problem definition and notation

Input. We are given a set of disjoint[1] balls $\mathcal{B} = \{\mathsf{b}_1, \ldots, \mathsf{b}_n\}$, where $\mathsf{b}_i = \mathsf{ball}(\mathsf{c}_i, \mathsf{r}_i)$, for $i = 1, \ldots, n$. Here $\mathsf{ball}(\mathsf{c}, \mathsf{r}) \subseteq \mathbb{R}^d$ denotes the (closed) ball with center c and radius $\mathsf{r} \geq 0$. Additionally, we are given an approximation parameter $\varepsilon \in (0, 1)$. For a point $\mathsf{q} \in \mathbb{R}^d$, the ***distance*** of q to a ball $\mathsf{b} = \mathsf{ball}(\mathsf{c}, \mathsf{r})$ is $\mathsf{d}(\mathsf{q}, \mathsf{b}) = \max\Big(\|\mathsf{q} - \mathsf{c}\| - \mathsf{r},\, 0 \Big)$.

Observation 5.1.1 *For two balls $\mathsf{b}_1 \subseteq \mathsf{b}_2 \subseteq \mathbb{R}^d$, and any point $\mathsf{q} \in \mathbb{R}^d$, we have $\mathsf{d}(\mathsf{q}, \mathsf{b}_1) \geq \mathsf{d}(\mathsf{q}, \mathsf{b}_2)$.*

The ***kth-nearest neighbor distance*** of q to $\mathcal{B}$, denoted by $\mathsf{d}_k(\mathsf{q}, \mathcal{B})$, is the kth smallest number in $\mathsf{d}(\mathsf{q}, \mathsf{b}_1), \ldots, \mathsf{d}(\mathsf{q}, \mathsf{b}_n)$. Similarly, for a given set of points P, $\mathsf{d}_k(\mathsf{q}, \mathsf{P})$ denotes the kth-nearest neighbor distance of q to P.

Problem definition. We aim to build a data-structure to answer $(1 \pm \varepsilon)$-approximate kth-nearest neighbor (i.e., **(k, ε)-ANN**) queries, where for any query point $\mathsf{q} \in \mathbb{R}^d$ one needs to output a ball $\mathsf{b} \in \mathcal{B}$ such that, $(1 - \varepsilon)\mathsf{d}_k(\mathsf{q}, \mathcal{B}) \leq \mathsf{d}(\mathsf{q}, \mathsf{b}) \leq (1 + \varepsilon)\mathsf{d}_k(\mathsf{q}, \mathcal{B})$. There are different variants depending on whether ε and k are provided with the query or in advance.

Notations. We use ***cube*** to denote a square ($d = 2$), a cube ($d = 3$), or a hypercube ($d > 3$), as the dimension might be.

Observation 5.1.2 *For any set of balls $\mathcal{B}$, the function $\mathsf{d}_k(\mathsf{q}, \mathcal{B})$ is a 1-Lipschitz function; that is, for any two points u, v, we have that $\mathsf{d}_k(\mathsf{u}, \mathcal{B}) \leq \mathsf{d}_k(\mathsf{v}, \mathcal{B}) + \|\mathsf{v} - \mathsf{u}\|$.*

Assumption 5.1.3 *We assume all the balls are contained inside the cube $\big[1/2 - \delta, 1/2 + \delta\big]^d$, which can be ensured by translation and scaling (which preserves order of distances), where $\delta = \varepsilon/4$. As such, we can ignore queries outside the unit cube $[0, 1]^d$, as any input ball is a valid answer in this case.*

[1] Our data-structure and algorithm work for the more general case where the balls are interior disjoint, where we define the interior of a "point ball", i.e., a ball of radius 0, as the point itself. This is not the usual topological definition.

Let $\mathcal{B}$ be a family of balls in $\mathbb{R}^d$. Given a set $X \subseteq \mathbb{R}^d$, let

$$\mathcal{B}(X) = \left\{ \mathsf{b} \in \mathcal{B} \;\middle|\; \mathsf{b} \cap X \neq \emptyset \right\}$$

denote the set of all balls in $\mathcal{B}$ that intersect X.

For two compact sets $X, Y \subseteq \mathbb{R}^d$, $X \preceq Y$ if and only if $\operatorname{diam}(X) \leq \operatorname{diam}(Y)$. For a set X and a set of balls $\mathcal{B}$, let $\mathcal{B}_{\succeq}(X) = \left\{ \mathsf{b} \in \mathcal{B} \;\middle|\; \mathsf{b} \cap X \neq \emptyset \text{ and } \mathsf{b} \succeq X \right\}$. Let c_d denote the maximum number of pairwise disjoint balls of radius at least r, that may intersect a given ball of radius r in $\mathbb{R}^d$. It can be verified that $2 \leq \mathsf{c}_d \leq 3^d$ for all d. Clearly, we have $|\mathcal{B}_{\succeq}(\mathsf{b})| \leq \mathsf{c}_d$ for any ball b.

Definition 5.1.4 For a parameter $\delta \geq 0$, a function $f : \mathbb{R}^+ \to \mathbb{R}^+$ is ***δ-monotonic***, if for every $x \geq 0$, $f(x/(1+\delta)) \leq f(x)$.

5.2 Approximate range counting for balls

Data-structure 5.2.1 For a given set of balls $\mathcal{B} = \{\mathsf{b}_1, \ldots, \mathsf{b}_n\}$ in $\mathbb{R}^d$, we build the following data-structure, that is useful in performing several of the tasks at hand.

(A) **Store balls in a (compressed) quadtree**. For $i = 1, 2, \ldots, n$, let $G_i = \mathsf{G}_{\approx}(\mathsf{b}_i)$ (see Definition 2.5.2), and let $G = \bigcup_{i=1}^{n} G_i$ denote the union of these cells. Let $\mathcal{T}$ be a compressed quadtree decomposition of $[0,1]^d$, such that all the cells of G are cells of $\mathcal{T}$. We preprocess $\mathcal{T}$ to answer point location queries. This takes $O(n \log n)$ time, see [Har11].

(B) **Compute list of "large" balls intersecting each cell.** For each node u of $\mathcal{T}$, there is a list of balls registered with it. Formally, ***register*** a ball b_i with all the cells of G_i. Clearly, each ball is registered with $O(1)$ cells, and it is easy to see that each cell has $O(1)$ balls registered with it, since the balls are disjoint.

Next, for a cell $\square$ in $\mathcal{T}$ we compute a list storing $\mathcal{B}_{\succeq}(\square)$, and these balls are ***associated*** with this cell. These lists are computed in a top-down manner. To this end, propagate from a node u its list $\mathcal{B}_{\succeq}(\square)$ (which we assume is already computed) down to its children. For a node receiving such a list, it scans it, and keep only the balls that intersect its cell (adding to this list the balls already registered with this cell). For a node $\nu \in \mathcal{T}$, let $\mathcal{B}_\nu$ be this list.

(C) **Build compressed quadtree on centers of balls.** Let $\mathcal{C}$ be the set of centers of the balls of $\mathcal{B}$. Build in $O(n \log n)$ time, a compressed quadtree $\mathcal{T}_{\mathcal{C}}$ storing $\mathcal{C}$.

(D) **ANN for centers of balls.** Build a data structure $\mathcal{D}$, for answering 2-approximate k-nearest neighbor distances on $\mathcal{C}$, the set of centers of the balls, see [HK12] or Chapter 4, where k and ε are provided

with the query. The data structure $\mathcal{D}$, returns a point $\mathsf{c} \in \mathcal{C}$ such that, $\mathsf{d}_k(\mathsf{q}, \mathcal{C}) \leq \mathsf{d}(\mathsf{q}, \mathsf{c}) \leq 2\mathsf{d}_k(\mathsf{q}, \mathcal{C})$.

(E) **Answering approximate range searching for the centers of balls.**

Given a query ball $\mathsf{b}_\mathsf{q} = \mathsf{ball}(\mathsf{q}, x)$ and a parameter $\delta > 0$, one can, using $\mathcal{T}_\mathcal{C}$, report (approximately), in $O(\log n + 1/\delta^d)$ time, the points in $\mathsf{b}_\mathsf{q} \cap \mathcal{C}$. Specifically, the query process computes $O(1/\delta^d)$ sets of points, such that their union X, has the property that $\mathsf{b}_\mathsf{q} \cap \mathcal{C} \subseteq X \subseteq (1+\delta)\mathsf{b}_\mathsf{q} \cap \mathcal{C}$, where $(1+\delta)\mathsf{b}_\mathsf{q}$ is the scaling of b_q by a factor of $1+\delta$ around its center. Indeed, compute the set $\mathsf{G}_\approx\big(\mathsf{b}_\mathsf{q}\big)$, and then using cell queries in $\mathcal{T}_\mathcal{C}$ compute the corresponding cells (this takes $O(\log n)$ time). Now, descend to the relevant level of the quadtree to all the cells of the right size, that intersect b_q. Clearly, the union of points stored in their subtrees are the desired set. This takes overall $O(\log n + 1/\delta^d)$ time.

A similar data-structure for approximate range searching is provided by Arya and Mount [AM00], and our description above is provided for the sake of completeness.

Overall, it takes $O(n \log n)$ time to build this data-structure, and we denote it by $\mathcal{DS}_{5.2.1}$.

5.2.1 Approximate range counting among balls

We need the ability to answer approximate range counting queries on a set of balls. Specifically, given a set of balls $\mathcal{B}$, and a query ball b, the target is to compute the size of the set $\mathsf{b} \cap \mathcal{B} = \Big\{ b' \in \mathcal{B} \;\Big|\; b' \cap \mathsf{b} \neq \emptyset \Big\}$. To make this query computationally fast, we allow an approximation. More precisely, for a ball b a set $\widetilde{\mathsf{b}}$ is a $(1+\delta)$-***ball*** of b, if $\mathsf{b} \subseteq \widetilde{\mathsf{b}} \subseteq (1+\delta)\mathsf{b}$, where $(1+\delta)\mathsf{b}$ is the $(1+\delta)$-scaling of b around its center. The purpose here, given a query ball b, is to compute the size of the set $\widetilde{\mathsf{b}} \cap \mathcal{B}$ for some $(1+\delta)$-ball $\widetilde{\mathsf{b}}$ of b. It turns out that this is challenging, as the query ball can be approximated, but the balls in $\mathcal{B}$ remain the same. This is to prevent the undesired situation where a "giant" ball is a valid answer for any ANN query.

Some useful tools

Lemma 5.2.2 *Given a compressed quadtree $\mathcal{T}$ of size n, a convex set X, and a parameter $\delta > 0$, one can compute the set of nodes in $\mathcal{T}$, that realizes $\mathsf{G}_\approx(X, \delta)$ (see Definition 2.5.2), in $O\big(\log n + 1/\delta^d\big)$ time. Specifically, this output a set X_N of nodes, of size $O\big(1/\delta^d\big)$, such that their cells intersect $\mathsf{G}_\approx(X, \delta)$, and their parents cell diameter is larger than $\delta\mathrm{diam}(X)$. Note that the cells in X_N might be significantly larger if they are leafs of $\mathcal{T}$.*

Proof: Let $\mathsf{G}_\approx = \mathsf{G}_\approx(X, 1)$ be the grid approximation to X. Using cell queries on the compressed quadtree, one can compute the cells of $\mathcal{T}$ that corresponds to these canonical cells. Specifically, for each cube $\square \in \mathsf{G}_\approx(X)$, the query either returns a node for which this is its cell, or it returns a compressed edge of the quadtree; that is, two cells (one is a parent of the other), such that $\square$ is contained in of them and contains

the other. Such a cell query takes $O(\log n)$ time [Har11]. This returns $O(1)$ nodes in $\mathcal{T}$ such that their cells cover $\mathsf{G}_{\approx}(X)$.

Now, traverse down the compressed quadtree starting from these nodes and collect all the nodes of the quadtree that are relevant. Clearly, one has to go at most $O(\log 1/\delta)$ levels down the quadtree to get these nodes, and this takes $O(1/\delta^d)$ time overall. ■

Lemma 5.2.3 *Let X be any convex set in $\mathbb{R}^d$, and let $\delta > 0$ be a parameter. Using $\mathcal{DS}_{5.2.1}$, one can compute, in $O\big(\log n + 1/\delta^d\big)$ time, all the balls of $\mathcal{B}$ that intersects X, with diameter $\geq \delta\mathrm{diam}(X)$.*

Proof: We compute the cells of the quadtree realizing $\mathsf{G}_{\approx}(X, \delta)$ using Lemma 5.2.2. Now, from each such cell (and its parent), we extract the list of large balls intersecting it (there are $O(1/\delta^d)$ such nodes, and the size of each such list is $O(1)$). Next we check for each such ball if it intersects X and if its diameter is at least $\delta\mathrm{diam}(X)$. We return the list of all such balls. ■

Answering a query

Given a query ball $\mathsf{b_q} = \mathsf{ball}(\mathsf{q}, x)$, and an approximation parameter $\delta > 0$, our purpose is to compute a number N, such that $\Big|\mathcal{B}\Big(\mathsf{ball}\Big(\mathsf{q}, x\Big)\Big)\Big| \leq N \leq \Big|\mathcal{B}\Big(\mathsf{ball}\Big(\mathsf{q}, (1+\delta)x\Big)\Big)\Big|$.

The query algorithm works as follows:

(A) Using Lemma 5.2.3, compute a set X of all the balls that intersect $\mathsf{b_q}$ and are of radius $\geq \delta x/4$.

(B) Using $\mathcal{DS}_{5.2.1}$, compute $O(1/\delta^d)$ cells of $\mathcal{T}_{\mathcal{C}}$ that corresponds to $\mathsf{G}_{\approx}\big(\mathsf{b_q}(1+\delta/4), \delta/4\big)$. Let N' be the total number of points in $\mathcal{C}$ stored in these nodes.

(C) The quantity $N' + |X|$ is almost the desired quantity, except that we might be counting some of the balls of X twice. To this end, let N'' be the number of balls in X with centers in $\mathsf{G}_{\approx}\big(\mathsf{b_q}(1+\delta/4), \delta/4\big)$

(D) Let $N \leftarrow N' + |X| - N''$. Return N.

Correctness. We only sketch the proof, as it is straightforward. Indeed, the union of the cells of $\mathsf{G}_{\approx}\big(\mathsf{b_q}(1+\delta/4), \delta/4\big)$ contains $\mathsf{ball}(\mathsf{q}, x(1+\delta/4))$ and is contained in $\mathsf{ball}(\mathsf{q}, (1+\delta)x)$. All the balls with radius smaller than $\delta x/4$ and intersecting $\mathsf{ball}(\mathsf{q}, x)$ have their centers in cells of $\mathsf{G}_{\approx}\big(\mathsf{b_q}(1+\delta/4), \delta/4\big)$, and their number is computed correctly. Similarly, the "large" balls are computed correctly. The last stage ensures we do not over-count by 1 each large ball that also has its center in $\mathsf{G}_{\approx}\big(\mathsf{b_q}(1+\delta/4), \delta/4\big)$. It is also easy to check that $|\mathcal{B}(\mathsf{ball}(\mathsf{q}, x))| \leq N \leq |\mathcal{B}(\mathsf{ball}(\mathsf{q}, x(1+\delta)))|$. The same result can be used for $x/(1+\delta)$ to get δ-monotonicity of N.

Query running time. Computing all the cells of $\mathsf{G}_{\approx}\big(\mathsf{b}_{\mathsf{q}}(1+\delta/4),\delta/4\big)$ takes $O(\log n + 1/\delta^d)$ time. Computing the "large" balls takes $O\big(\log n + 1/\delta^d\big)$ time. Checking for each large ball if it is already counted by the "small" balls takes $O(1/\delta^d)$ by using a grid.

Result. The above implies the following.

Lemma 5.2.4 (rangeCount) *Given a set $\mathcal{B}$ of n balls in $\mathbb{R}^d$, it can be preprocessed, in $O(n \log n)$ time, into a data structure of size $O(n)$, such that given a query ball $\mathsf{ball}(\mathsf{q}, x)$ and approximation parameter $\delta > 0$, the query algorithm* rangeCount(q, x, δ) *returns, in $O(\log n + 1/\delta^d)$ time, a number N satisfying the following:*

(A) $N \leq |\mathcal{B}(\mathsf{ball}(\mathsf{q}, (1+\delta)x))|$,

(B) $|\mathcal{B}(\mathsf{ball}(\mathsf{q}, x))| \leq N$, *and*

(C) for a query ball $\mathsf{ball}(\mathsf{q}, x)$ and δ, the number N is δ-monotonic as a function of x, see Definition 5.1.4.

5.3 Answering k-ANN queries among balls

5.3.1 Computing a constant factor approximation to $\mathsf{d}_k(\mathsf{q}, \mathcal{B})$

Lemma 5.3.1 *Let $\mathcal{B}$ be a set of disjoint balls in $\mathbb{R}^d$, and consider a ball $\mathsf{b} = \mathsf{ball}(\mathsf{q}, r)$ that intersects at least k balls of $\mathcal{B}$. Then, among the k nearest neighbors of q from $\mathcal{B}$, there are at least $\max(0, k - \mathsf{c}_d)$ balls of radius at most r. The centers of all these balls are in $\mathsf{ball}(\mathsf{q}, 2r)$.*

Proof: Consider the k nearest neighbors of q from $\mathcal{B}$. Any such ball that has its center outside $\mathsf{ball}(\mathsf{q}, 2r)$, has radius at least r, since it intersects $\mathsf{b} = \mathsf{ball}(\mathsf{q}, r)$. Since the number of balls that are of radius at least r and intersect b is bounded by c_d, there must be at least $\max(0, k - \mathsf{c}_d)$ balls among the k nearest neighbors, each having radius less than r. Now, $\mathsf{ball}(\mathsf{q}, 2r)$ will contain the centers of all such balls. ■

Corollary 5.3.2 *Let $\gamma = \min(k, \mathsf{c}_d)$. Then, $\mathsf{d}_{k-\gamma}(\mathsf{q}, \mathcal{C})/2 \leq \mathsf{d}_k(\mathsf{q}, \mathcal{B})$.*

Idea. The basic observation is that we only need a rough approximation to the right radius, as using approximate range counting (i.e., Lemma 5.2.4), one can improve the approximation.

Let x_i denote the distance of q to the ith closest center in $\mathcal{C}$. Let $d_k = \mathsf{d}_k(\mathsf{q}, \mathcal{B})$. Let i be the minimum index, such that $d_k \leq x_i$. Since $d_k \leq x_k$, it must be that $i \leq k$. There are several possibilities:

(A) If $i \leq k - \mathsf{c}_d$ (i.e., $d_k \leq x_{k-\mathsf{c}_d}$) then, by Lemma 5.3.1, the ball $\mathsf{ball}(\mathsf{q}, 2d_k)$ contains at least $k - \mathsf{c}_d$ centers. As such, $d_k < x_{k-\mathsf{c}_d} \leq 2d_k$, and $x_{k-\mathsf{c}_d}$ is a good approximation to d_k.

(B) If $i > k - \mathsf{c}_d$, and $d_k \leq 4x_{i-1}$, then x_{i-1} is the desired approximation.

(C) If $i > k - \mathsf{c}_d$, and $d_k \geq x_i/4$, then x_i is the desired approximation.

(D) Otherwise, it must be that $i > k - \mathsf{c}_d$, and $x_{i-1}/4 \leq d_k \leq x_i/4$. Let $\mathsf{b}_j = \mathsf{ball}(\mathsf{c}_j, \mathsf{r}_j)$ be the jth closest ball to q, for $j = 1, \ldots, k$. It must be that $\mathsf{b}_i, \ldots, \mathsf{b}_k$ are much larger than $\mathsf{ball}(\mathsf{q}, d_k)$. But then, the balls $\mathsf{b}_i, \ldots, \mathsf{b}_k$ must intersect $\mathsf{ball}(\mathsf{q}, x_i/2)$, and their radius is at least $x_i/2$. We can easily compute these big balls using $\mathcal{DS}_{5.2.1}$ (B), and the number of centers of the small balls close to query, and then compute d_k exactly.

Preprocessing. We build $\mathcal{DS}_{5.2.1}$ in $O(n \log n)$ time.

Notations. For $x \geq 0$, let $N(x)$ denote the number of balls in $\mathcal{B}$ that intersect $\mathsf{ball}(\mathsf{q}, x)$; that is $N(x) = \left|\left\{\mathsf{b} \in \mathcal{B} \;\middle|\; \mathsf{b} \cap \mathsf{ball}(\mathsf{q}, x) \neq \emptyset\right\}\right|$, and $C(x)$ denote the number of centers in $\mathsf{ball}(\mathsf{q}, x)$, i.e., $C(x) = |\mathcal{C} \cap \mathsf{ball}(\mathsf{q}, x)|$. Also, let $\#(x)$ denote the 2-approximation to the number of balls of $\mathcal{B}$ intersecting $\mathsf{ball}(\mathsf{q}, x)$, as computed by Lemma 5.2.4; that is $N(x) \leq \#(x) \leq N(2x)$.

Answering a query. We are given a query point $\mathsf{q} \in \mathbb{R}^d$ and a number k.

Using $\mathcal{DS}_{5.2.1}$, compute a 2-approximation for the smallest ball containing $k - i$ centers of $\mathcal{B}$, for $i = 0, \ldots, \gamma$, where $\gamma = \min(k, \mathsf{c}_d)$, and let r_{k-i} be this radius. That is, for $i = 0, \ldots, \gamma$, we have $C(r_{k-i}/2) \leq k - i \leq C(r_{k-i})$. For $i = 0, \ldots, \gamma$, compute $N_{k-i} = \#(r_{k-i})$ (Lemma 5.2.4).

Let α be the maximum index such that $N_{k-\alpha} \geq k$. Clearly, α is well defined as $N_k \geq k$. The algorithm is executed in the following steps.

(A) If $\alpha = \gamma$ we return $2r_{k-\gamma}$.

(B) If $\#(r_{k-\alpha}/4) < k$, we return $2r_{k-\alpha}$.

(C) Otherwise, compute all the balls of $\mathcal{B}$ that are of radius at least $r_{k-\alpha}/4$ and intersect the ball $\mathsf{ball}(\mathsf{q}, r_{k-\alpha}/4)$, using $\mathcal{DS}_{5.2.1}$ (B). For each such ball b, compute the distance $\zeta = \mathsf{d}(\mathsf{q}, \mathsf{b})$ of q to it. Return 2ζ for the minimum such number ζ such that $\#(\zeta) \geq k$.

Lemma 5.3.3 *Given a set of n disjoint balls $\mathcal{B}$ in $\mathbb{R}^d$, one can preprocess them, in $O(n \log n)$ time, into a data structure of size $O(n)$, such that given a query point $\mathsf{q} \in \mathbb{R}^d$, and a number k, one can compute, in $O(\log n)$ time, a number x such that, $x/4 \leq \mathsf{d}_k(\mathsf{q}, \mathcal{B}) \leq 4x$.*

Proof: The data-structure and query algorithm are described above. We next prove correctness. To prove that (A) returns the correct answer observe that under the given assumptions,

$$r_{k-\gamma}/4 \leq \mathsf{d}_{k-\gamma}(\mathsf{q}, \mathcal{C})/2 \leq \mathsf{d}_k(\mathsf{q}, \mathcal{B}) \leq 2r_{k-\gamma},$$

where the second inequality follows from Corollary 5.3.2, and the third inequality follows as $N(2r_{k-\gamma}) \geq \#(r_{k-\gamma}) \geq k$, while $\mathsf{d}_k(\mathsf{q}, \mathcal{B})$ is the smallest number x such that $N(x) \geq k$.

For (B) observe that we have that $N(r_{k-\gamma}/4) \leq \#(r_{k-\gamma}/4) < k$ and as such we have $r_{k-\gamma}/4 < \mathsf{d}_k(\mathsf{q}, \mathcal{B})$. But by assumption, $\#(r_{k-\gamma}) \geq k$ and so $N(2r_{k-\gamma}) \geq \#(r_{k-\gamma}) \geq k$, thus $\mathsf{d}_k(\mathsf{q}, \mathcal{B}) \leq 2r_{k-\gamma}$.

For (C), first observe that $\alpha < \gamma$ as the algorithm did not return in (A). Since α is the maximum index such that $\#(r_{k-\alpha}) \geq k$, so $N(r_{k-\alpha-1}) \leq \#(r_{k-\alpha-1}) < k$ implying, $r_{k-\alpha-1} < \mathsf{d}_k(\mathsf{q}, \mathcal{B})$. Also, $\mathsf{d}_k(\mathsf{q}, \mathcal{B}) \leq r_{k-\alpha}/4$, as the algorithm did not return in (B). Now the ball $\mathsf{ball}(\mathsf{q}, r_{k-\alpha-1})$ contains at least $k - \alpha - 1$ centers from $\mathcal{C}$, but it does not contain $k - \alpha$ centers. Indeed, otherwise we would have $\mathsf{d}_{k-\alpha}(\mathsf{q}, \mathcal{C}) \leq r_{k-\alpha-1}$ and so $r_{k-\alpha} \leq 2\mathsf{d}_{k-\alpha}(\mathsf{q}, \mathcal{C}) \leq 2r_{k-\alpha-1}$, but on the other hand $r_{k-\alpha-1} < \mathsf{d}_k(\mathsf{q}, \mathcal{B}) \leq r_{k-\alpha}/4$, which would be a contradiction. Similarly, there is no center of any ball whose distance from q is in the range $(r_{k-\alpha-1}, r_{k-\alpha}/2)$ otherwise we would have that $\mathsf{d}_{k-\alpha}(\mathsf{q}, \mathcal{C}) < r_{k-\alpha}/2$ and this would mean that $r_{k-\alpha} \leq 2\mathsf{d}_{k-\alpha}(\mathsf{q}, \mathcal{C}) < r_{k-\alpha}$, a contradiction. Now, the center of the kth closest ball is clearly more than $r_{k-\alpha-1}$ away from q. As such its distance from q is at least $r_{k-\alpha}/2$. Since $\mathsf{d}_k(\mathsf{q}, \mathcal{B}) \leq r_{k-\alpha}/4$ it follows that the kth closest ball intersects $\mathsf{ball}(\mathsf{q}, r_{k-\alpha}/4)$ and moreover, its radius is at least $r_{k-\alpha}/4$. Since we compute all such balls in (C), we do encounter the kth closest ball. It is easy to see that in this case we return a number ζ satisfying, $\zeta/2 \leq \mathsf{d}_k(\mathsf{q}, \mathcal{B}) \leq 2\zeta$.

As for the running time, notice that we need to use the algorithm of Lemma 5.2.4 $O(1)$ times, each iteration taking time $O(\log n)$. After this we need another $O(\log n)$ time for the invocation of the algorithm in Lemma 5.2.3. As such, the total query time is $O(\log n)$. ■

Refining the approximation of $\mathsf{d}_k(\mathsf{q}, \mathcal{B})$

Lemma 5.3.4 *Given a set $\mathcal{B}$ of n balls in $\mathbb{R}^d$, it can be preprocessed, in $O(n \log n)$ time, into a data structure of size $O(n)$. Given a query point q, numbers k, x, and an approximation parameter $\varepsilon > 0$, such that $x/4 \leq \mathsf{d}_k(\mathsf{q}, \mathcal{B}) \leq 4x$, one can find a ball $\mathsf{b} \in \mathcal{B}$ such that, $(1-\varepsilon)\mathsf{d}_k(\mathsf{q}, \mathcal{B}) \leq \mathsf{d}(\mathsf{q}, \mathsf{b}) \leq (1+\varepsilon)\mathsf{d}_k(\mathsf{q}, \mathcal{B})$, in $O\big(\log n + 1/\varepsilon^d\big)$ time.*

Proof: We are going to use the same data-structure as Lemma 5.2.4, for the query ball $\mathsf{b}_\mathsf{q} = \mathsf{ball}(\mathsf{q}, 4x(1+\varepsilon))$. We compute all large balls of $\mathcal{B}$ that intersect b_q. Here a large ball is a ball of radius $> x\varepsilon$, and a ball of radius at most $x\varepsilon$ is considered to be a small ball. Consider the $O(1/\varepsilon^d)$ grid cells of $\mathsf{G}_\approx\big(\mathsf{b}_\mathsf{q}, \varepsilon/16\big)$. In $O(1/\varepsilon^d)$ time we can record the number of centers of large balls inside any such cell. Clearly, any small ball that intersects $\mathsf{ball}(\mathsf{q}, 4x)$ has its center in some cell of $\mathsf{G}_\approx\big(\mathsf{b}_\mathsf{q}, \varepsilon/16\big)$. We use the quadtree $\mathcal{T}_\mathcal{C}$ to find out exactly the number of centers, $N_\square$, of small balls in each cell $\square$ of $\mathsf{G}_\approx\big(\mathsf{b}_\mathsf{q}, \varepsilon/16\big)$, by finding the total number of centers using $\mathcal{T}_\mathcal{C}$, and decreasing this by the count of centers of large balls in that cell. This can

be done in time $O(\log n + 1/\varepsilon^d)$. We pick an arbitrary point in $\square$, and assign it weight $N_\square$, and treat it as representing all the small balls in this grid cell – clearly, this introduces an error of size $\leq \varepsilon x$ in the distance of such a ball from q, and as such we can ignore it in our argument. In the end of this snapping process, we have $O(1/\varepsilon^d)$ weighted points, and $O(1/\varepsilon^d)$ large balls. We know the distance of the query point from each one of these points/balls. This results in $O(1/\varepsilon^d)$ weighted distances, and we want the smallest ℓ, such that the total weight of the distances $\leq \ell$ is at least k. This can be done by weighted median selection in linear time in the number of distances, which is $O(1/\varepsilon^d)$. Once we get the required point we can output any ball b corresponding to the point. Clearly, b satisfies the required conditions. ■

The result

Theorem 5.3.5 *Given a set of n disjoint balls $\mathcal{B}$ in $\mathbb{R}^d$, one can preprocess them in time $O(n \log n)$ into a data structure of size $O(n)$, such that given a query point $\mathsf{q} \in \mathbb{R}^d$, a number k with $1 \leq k \leq n$ and $\varepsilon > 0$, one can find in time $O\big(\log n + \varepsilon^{-d}\big)$ a ball $\mathsf{b} \in \mathcal{B}$, such that, $(1-\varepsilon)\mathsf{d}_k(\mathsf{q}, \mathcal{B}) \leq \mathsf{d}(\mathsf{q}, \mathsf{b}) \leq (1+\varepsilon)\mathsf{d}_k(\mathsf{q}, \mathcal{B})$.*

5.4 Quorum clustering

We are given a set $\mathcal{B}$ of n disjoint balls in $\mathbb{R}^d$, and we describe how to compute quorum clustering for them quickly.

Let ξ be some constant. Let $\mathcal{B}_0 = \emptyset$. For $i = 1, \ldots, m$, let $\mathcal{R}_i = \mathcal{B} \setminus (\bigcup_{j=0}^{i-1} \mathcal{B}_j)$, and let $\Lambda_i = \mathsf{ball}(\mathsf{w}_i, \mathsf{x}_i)$ be any ball that satisfies,

(A) Λ_i contains $\min(k - \mathsf{c}_d, |\mathcal{R}_i|)$ balls of $\mathcal{R}_i$ completely inside it,

(B) Λ_i intersects at least k balls of $\mathcal{B}$, and

(C) the radius of Λ_i is at most ξ times the radius of the smallest ball satisfying the above conditions.

Next, we remove any $k - \mathsf{c}_d$ balls that are contained in Λ_i from $\mathcal{R}_i$ to get the set $\mathcal{R}_{i+1}$. We repeat this process till all balls are extracted. Notice that at each step i we only require that the Λ_i intersects k balls of $\mathcal{B}$ (and not $\mathcal{R}_i$), but that it must contain $k - \mathsf{c}_d$ balls from $\mathcal{R}_i$. Also, the last quorum ball may contain fewer balls. The balls $\Lambda_1, \ldots, \Lambda_m$, are the resulting ***$\xi$-approximate quorum clustering***.

5.4.1 Computing an approximate quorum clustering

Definition 5.4.1 For a set P of n points in $\mathbb{R}^d$, and an integer ℓ, with $1 \leq \ell \leq n$, let $r_{\mathrm{opt}}(\mathsf{P}, \ell)$ denote the radius of the smallest ball which contains at least ℓ points from P, i.e., $r_{\mathrm{opt}}(\mathsf{P}, \ell) = \min_{\mathsf{q} \in \mathbb{R}^d} \mathsf{d}_\ell(\mathsf{q}, \mathsf{P})$.

Similarly, for a set $\mathcal{R}$ of n balls in $\mathbb{R}^d$, and an integer ℓ, with $1 \leq \ell \leq n$, let $R_{\text{opt}}(\mathcal{R}, \ell)$ denote the radius of the smallest ball which completely contains at least ℓ balls from $\mathcal{R}$.

Lemma 5.4.2 ([HK12]) *Given a set P of n points in $\mathbb{R}^d$ and integer ℓ, with $1 \leq \ell \leq n$, one can compute, in $O(n \log n)$ time, a sequence of $\lceil n/\ell \rceil$ balls, $\mathsf{o}_1 = \mathsf{ball}(\mathsf{u}_1, \psi_1), \ldots, \mathsf{o}_{\lceil n/\ell \rceil} = \mathsf{ball}\big(\mathsf{u}_{\lceil n/\ell \rceil}, \psi_{\lceil n/\ell \rceil}\big)$, such that, for all $i, 1 \leq i \leq \lceil n/\ell \rceil$, we have*

(A) *For every ball o_i, there is an associated subset P_i of $\min(\ell, |\mathsf{Q}_i|)$ points of $\mathsf{Q}_i = \mathsf{P} \setminus (\mathsf{P}_i \cup \ldots \cup \mathsf{P}_{i-1})$, that it covers.*

(B) *The ball $\mathsf{o}_i = \mathsf{ball}(\mathsf{u}_i, \psi_i)$ is a 2-approximation to the smallest ball covering $\min(\ell, |\mathsf{Q}_i|)$ points in Q_i; that is, $\psi_i/2 \leq r_{\text{opt}}(\mathsf{Q}_i, \min(\ell, |\mathsf{Q}_i|)) \leq \psi_i$.*

The algorithm. The algorithm to construct an approximate quorum clustering is as follows. We use the algorithm of Lemma 5.4.2 with the set of points $\mathsf{P} = \mathcal{C}$, and $\ell = k - \mathsf{c}_d$ to get a list of $m = \lceil n/(k - \mathsf{c}_d) \rceil$ balls $\mathsf{o}_1 = \mathsf{ball}(\mathsf{u}_1, \psi_1), \ldots, \mathsf{o}_m = \mathsf{ball}(\mathsf{u}_m, \psi_m)$, satisfying the conditions of Lemma 5.4.2. Next we use the algorithm of Theorem 5.3.5, to compute (k, ε)-ANN distances from the centers $\mathsf{u}_1, \ldots, \mathsf{u}_m$, to the balls of $\mathcal{B}$.

Thus, we get numbers γ_i satisfying, $(1/2)\mathsf{d}_k(\mathsf{u}_i, \mathcal{B}) \leq \gamma_i \leq (3/2)\mathsf{d}_k(\mathsf{u}_i, \mathcal{B})$. Let $\zeta_i = \max(2\gamma_i, 3\psi_i)$, for $i = 1, \ldots, m$. Sort $\zeta_1, \ldots, \zeta_m$ (we assume for the sake of simplicity of exposition that ζ_m, being the radius of the last cluster is the largest number). Suppose the sorted order is the permutation π of $\{1, \ldots, m\}$ (by assumption $\pi(m) = m$). We output the balls $\Lambda_i = \mathsf{ball}\big(\mathsf{u}_{\pi(i)}, \zeta_{\pi(i)}\big)$, for $i = 1, \ldots, m$, as the approximate quorum clustering.

Correctness

Lemma 5.4.3 *Let $\mathcal{B} = \{\mathsf{b}_1, \ldots, \mathsf{b}_n\}$ be a set of n disjoint balls, where $\mathsf{b}_i = \mathsf{ball}(\mathsf{c}_i, \mathsf{r}_i)$, for $i = 1, \ldots, n$. Let $\mathcal{C} = \{\mathsf{c}_1, \ldots, \mathsf{c}_n\}$ be the set of centers of these balls. Let $\mathsf{b} = \mathsf{ball}(\mathsf{c}, \mathsf{r})$ be any ball that contains at least ℓ centers from $\mathcal{C}$, for some $2 \leq \ell \leq n$. Then $\mathsf{ball}(\mathsf{c}, 3\mathsf{r})$ contains the ℓ balls that correspond to those centers.*

Proof: Without loss of generality suppose b contains the ℓ centers $\mathsf{c}_1, \ldots, \mathsf{c}_\ell$, from $\mathcal{C}$. Now consider any index i with $1 \leq i \leq \ell$, and consider any $j \neq i$, which exists as $\ell \geq 2$ by assumption. Since $\mathsf{ball}(\mathsf{c}, \mathsf{r})$ contains both c_i and c_j, $2\mathsf{r} \geq \|\mathsf{c}_i - \mathsf{c}_j\|$ by the triangle inequality. On the other hand, as the balls b_i and b_j are disjoint we have that $\|\mathsf{c}_i - \mathsf{c}_j\| \geq \mathsf{r}_i + \mathsf{r}_j \geq \mathsf{r}_i$. It follows that $\mathsf{r}_i \leq 2\mathsf{r}$ for all $1 \leq i \leq \ell$. As such the ball $\mathsf{ball}(\mathsf{c}, 3\mathsf{r})$ must contain the entire ball b_i, and thus it contains all the ℓ balls $\mathsf{b}_1, \ldots, \mathsf{b}_\ell$, corresponding to the centers. ■

Lemma 5.4.4 *Let $\mathcal{B} = \{\mathsf{b}_1 = \mathsf{ball}(\mathsf{c}_1, \mathsf{r}_1), \ldots, \mathsf{b}_n = \mathsf{ball}(\mathsf{c}_n, \mathsf{r}_n)\}$ be a set of n disjoint balls in $\mathbb{R}^d$. Let $\mathcal{C} = \{\mathsf{c}_1, \ldots, \mathsf{c}_n\}$ be the corresponding set of centers, and let ℓ be an integer with $2 \leq \ell \leq n$. Then, $r_{\text{opt}}(\mathcal{C}, \ell) \leq R_{\text{opt}}(\mathcal{B}, \ell) \leq 3r_{\text{opt}}(\mathcal{C}, \ell)$.*

Proof: The first inequality follows since the ball realizing the optimal covering of ℓ balls, clearly contains their centers as well, and therefore ℓ points from $\mathcal{C}$. To see the second inequality, consider the ball $\mathsf{b} = \mathsf{ball}(\mathsf{c}, \mathsf{r})$ realizing $r_{\mathrm{opt}}(\mathcal{C}, \ell)$, and use Lemma 5.4.3 on it. This implies $R_{\mathrm{opt}}(\mathcal{B}, \ell) \leq 3r_{\mathrm{opt}}(\mathcal{C}, \ell)$. ■

Lemma 5.4.5 *The balls $\Lambda_1, \ldots \Lambda_m$ computed above are a 12-approximate quorum clustering of $\mathcal{B}$.*

Proof: Consider the balls $\mathsf{o}_1 = \mathsf{ball}(\mathsf{u}_1, \psi_1), \ldots, \mathsf{o}_m = \mathsf{ball}(\mathsf{u}_m, \psi_m)$ computed by the algorithm of Lemma 5.4.2. Suppose $\mathcal{C}_i$, for $1 = 1, \ldots, m$, is the set of centers assigned to the balls b_i. That is, $\mathcal{C}_1, \ldots, \mathcal{C}_m$ form a disjoint decomposition of $\mathcal{C}$, each of size $k - \mathsf{c}_d$ (except for the last set, which might be smaller – a technicality that we ignore for the sake of simplicity of exposition).

For $i = 1, \ldots, m$, let $\mathcal{B}_i$ denote the set of balls corresponding to the centers in $\mathcal{C}_i$. Now while constructing the approximate quorum clusters we are going to assign the set of balls $\mathcal{B}_{\pi(i)}$ for $i = 1, \ldots, m$, to Λ_i. Now, fix a i with $1 \leq i \leq m-1$. The balls of $\bigcup_{j=1}^{i} \mathcal{B}_{\pi(j)}$ have been used up. Consider an optimal ball, i.e., a ball $\mathsf{b} = \mathsf{ball}(\mathsf{c}, \mathsf{r})$ that contains completely $k - \mathsf{c}_d$ balls among $\bigcup_{j=i+1}^{m} \mathcal{B}_{\pi(j)}$ and intersects k balls from $\mathcal{B}$, and is the smallest such possible. Fix some $k - \mathsf{c}_d$ balls from $\bigcup_{j=i+1}^{m} \mathcal{B}_{\pi(j)}$ that this optimal ball contains. Consider the sets of centers $\mathcal{C}'$ of these balls. The quorum clusters $\mathsf{o}_{\pi(j)}$ for $j = i+1, \ldots, m$, contain all these centers, by construction. Out of these indices, i.e., out of the indices $\{\pi(i+1), \ldots, \pi(m)\}$, suppose p is the minimum index such that o_p contains one of these centers. When o_p was constructed, i.e., at the pth iteration of the algorithm of Lemma 5.4.2, all the centers from $\mathcal{C}'$ were available. Now since the optimal ball $\mathsf{b} = \mathsf{ball}(\mathsf{c}, \mathsf{r})$ contains $k - \mathsf{c}_d$ available centers too, it follows that $\psi_p \leq 2\mathsf{r}$ since Lemma 5.4.2 guarantees this. Since $k - \mathsf{c}_d \geq 2$, by Lemma 5.4.3, $\mathsf{ball}(\mathsf{u}_p, 3\psi_p)$ contains the balls of $\mathcal{B}_p$. Moreover, by the Lipschitz property, see Observation 5.1.2, it follows that $\mathsf{d}_k(\mathsf{u}_p, \mathcal{B}) \leq \mathsf{d}_k(\mathsf{c}, \mathcal{B}) + \|\mathsf{u}_p - \mathsf{c}\| \leq \mathsf{r} + (\mathsf{r} + \psi_p) \leq 4\mathsf{r}$, where the second last inequality follows as the balls $\mathsf{b} = \mathsf{ball}(\mathsf{c}, \mathsf{r})$ and the ball $\mathsf{o}_p = \mathsf{ball}(\mathsf{u}_p, \psi_p)$ intersect. Therefore, for the index p we have that, $\mathsf{d}_k(\mathsf{u}_p, \mathcal{B}) \leq 2\gamma_p \leq 3\mathsf{d}_k(\mathsf{u}_p, \mathcal{B}) \leq 12\mathsf{r}$, and also that $3\psi_p \leq 6\mathsf{r}$. As such $\zeta_p = \max(2\gamma_p, 3\psi_p) \leq 12\mathsf{r}$. The index $\pi(i+1)$ minimizes this quantity among the indices $\{\pi(i+1), \ldots, \pi(m)\}$ (as we took the sorted order), as such it follows that $\zeta_{i+1} \leq 12\mathsf{r}$. ■

Lemma 5.4.6 *Given a set $\mathcal{B}$ of n disjoint balls in $\mathbb{R}^d$, such that $(k - \mathsf{c}_d) | n$, and a number k with $2\mathsf{c}_d < k \leq n$, in $O(n \log n)$ time, one can output a sequence of $m = n/(k - \mathsf{c}_d)$ balls $\Lambda_1, \ldots, \Lambda_m$, such that*

(A) For each ball Λ_i, there is an associated subset $\mathcal{B}_i$ of $k - \mathsf{c}_d$ balls of $\mathcal{R}_i = \mathcal{B} \setminus (\mathcal{B}_1 \cup \ldots \cup \mathcal{B}_{i-1})$, that it completely covers.

(B) The ball Λ_i intersects at least k balls from $\mathcal{B}$.

(C) The radius of the ball Λ_i is at most 12 times that of the smallest ball covering $k - \mathsf{c}_d$ balls of $\mathcal{R}_i$ completely, and intersecting k balls of $\mathcal{B}$.

Proof: The correctness was proved in Lemma 5.4.5. To see the time bound is also easy as the computation time is dominated by the time in Lemma 5.4.2, which is $O(n \log n)$. ■

5.5 Construction of the sublinear space data-structure for (k, ε)-ANN

Here we show how to compute an approximate Voronoi diagram for approximating the kth-nearest ball, that takes $O(n/k)$ space. We assume $k > 2\mathsf{c}_d$ without loss of generality, and we let $m = \lceil n/(k - \mathsf{c}_d) \rceil = O(n/k)$. Here k and ε are prespecified in advance.

5.5.1 Preliminaries

The following notation was introduced in [HK12], see also Chapter 4. A ball b of radius r in $\mathbb{R}^d$, centered at a point c, can be interpreted as a point in $\mathbb{R}^{d+1}$, denoted by $\mathsf{b}' = (\mathsf{c}, \mathsf{r})$. For a regular point $\mathsf{p} \in \mathbb{R}^d$, its corresponding image under this transformation is the ***mapped*** point $\mathsf{p}' = (\mathsf{p}, 0) \in \mathbb{R}^{d+1}$, i.e., we view it as a ball of radius 0 and use the mapping defined on balls. Given point $\mathsf{u} = (\mathsf{u}_1, \ldots, \mathsf{u}_d) \in \mathbb{R}^d$ we will denote its Euclidean norm by $\|\mathsf{u}\|$. We will consider a point $\mathsf{u} = (\mathsf{u}_1, \mathsf{u}_2, \ldots, \mathsf{u}_{d+1}) \in \mathbb{R}^{d+1}$ to be in the product metric of $\mathbb{R}^d \times \mathbb{R}$ and endowed with the product metric norm

$$\|\mathsf{u}\|_{\oplus} = \sqrt{\mathsf{u}_1^2 + \cdots + \mathsf{u}_d^2} + |\mathsf{u}_{d+1}| \,.$$

It can be verified that the above defines a norm, and for any $\mathsf{u} \in \mathbb{R}^{d+1}$ we have $\|\mathsf{u}\| \leq \|\mathsf{u}\|_{\oplus} \leq \sqrt{2}\,\|\mathsf{u}\|$.

Construction

The input is a set $\mathcal{B}$ of n disjoint balls in $\mathbb{R}^d$, and parameters k and ε.

The construction of the data-structure is similar to the construction of the kth-nearest neighbor data-structure from the authors' paper [HK12], see also Chapter 4. We compute, using Lemma 5.4.6, a ξ-approximate quorum clustering of $\mathcal{B}$ with $m = n/(k - \mathsf{c}_d) = O(n/k)$ balls,

$$\Sigma = \{\Lambda_1 = \mathsf{ball}(\mathsf{w}_1, \mathsf{x}_1), \ldots, \Lambda_m = \mathsf{ball}(\mathsf{w}_m, \mathsf{x}_m)\},$$

where $\xi \leq 12$. The algorithm then continues as follows:

(A) Compute an exponential grid around each quorum cluster. Specifically, let

$$\mathcal{I} = \bigcup_{i=1}^{m} \bigcup_{j=0}^{\lceil \log(32\xi/\varepsilon) \rceil} \mathsf{G}_{\approx}\left(\mathsf{ball}\left(\mathsf{w}_i, 2^j \mathsf{x}_i\right), \frac{\varepsilon}{\zeta_1} \right) \tag{5.1}$$

be the set of grid cells covering the quorum clusters and their immediate environ (see Definition 2.5.2), where ζ_1 is a sufficiently large constant (say, $\zeta_1 = 256\xi$).

(B) Intuitively, $\mathcal{I}$ takes care of the region of space immediately next to a quorum cluster[2]. For the other regions of space, we can apply a construction of an approximate Voronoi diagram for the centers of the clusters (the details are somewhat more involved). To this end, lift the quorum clusters into points in $\mathbb{R}^{d+1}$, as follows

$$\Sigma' = \{\Lambda'_1, \ldots, \Lambda'_m\},$$

where $\Lambda'_i = (\mathsf{w}_i, \mathsf{x}_i) \in \mathbb{R}^{d+1}$, for $i = 1, \ldots, m$. Note that all points in Σ' belong to $U' = [0,1]^{d+1}$ by Assumption 5.1.3. Now build a $(1+\varepsilon/8)$-AVD for Σ' using the algorithm of Arya and Malamatos [AM02], for distances specified by the $\|\cdot\|_{\oplus}$ norm. The AVD construction provides a list of canonical cubes covering $[0,1]^{d+1}$ such that in the smallest cube containing the query point, the associated point of Σ', is a $(1+\varepsilon/8)$-ANN to the query point. (Note that these cubes are not necessarily disjoint. In particular, the smallest cube containing the query point q is the one that determines the assigned approximate nearest neighbor to q.)

Clip this collection of cubes to the hyperplane $x_{d+1} = 0$ (i.e., throw away cubes that do not have a face on this hyperplane). For a cube $\square$ in this collection, denote by $\mathrm{nn}'(\square)$, the point of Σ' assigned to it. Let $\mathcal{S}$ be this resulting set of canonical d-dimensional cubes.

(C) Let $\mathcal{W}$ be the space decomposition resulting from overlaying the two collection of cubes, i.e., $\mathcal{I}$ and $\mathcal{S}$. Formally, we compute a compressed quadtree $\mathcal{T}$ that has all the canonical cubes of $\mathcal{I}$ and $\mathcal{S}$ as nodes, and $\mathcal{W}$ is the resulting decomposition of space into cells. One can overlay two compressed quadtrees representing the two sets in linear time [BHTT10, Har11]. Here, a cell associated with a leaf is a canonical cube, and a cell associated with a compressed node is the set difference of two canonical cubes. Each node in this compressed quadtree contains two pointers – to the smallest cube of $\mathcal{I}$, and to the smallest cube of $\mathcal{S}$, that contains it. This information can be computed by doing a BFS on the tree.

For each cell $\square \in \mathcal{W}$ we store the following.

[2] That is, intuitively, if the query point falls into one of the grid cells of $\mathcal{I}$, we can answer a query in constant time.

(I) An arbitrary representative point $\square_{\mathsf{rep}} \in \square$.

(II) The point $\mathrm{nn}'(\square) \in \Sigma'$ that is associated with the smallest cell of $\mathcal{S}$ that contains this cell. We also store an arbitrary ball, $\mathbf{b}(\square) \in \mathcal{B}$, that is one of the balls completely inside the cluster specified by $\mathrm{nn}'(\square)$ – we assume we stored such a ball inside each quorum cluster, when it was computed.

(III) A number $\beta_k(\square_{\mathsf{rep}})$ that satisfies $\mathsf{d}_k(\square_{\mathsf{rep}}, \mathcal{B}) \leq \beta_k(\square_{\mathsf{rep}}) \leq (1+\varepsilon/4)\mathsf{d}_k(\square_{\mathsf{rep}}, \mathcal{B})$, and a ball $\mathrm{nn}_k(\square_{\mathsf{rep}}) \in \mathcal{B}$ that realizes this distance. In order to compute $\beta_k(\square_{\mathsf{rep}})$ and $\mathrm{nn}_k(\square_{\mathsf{rep}})$ use the data-structure of Section 5.3, see Theorem 5.3.5.

Answering a query

Given a query point q, compute the leaf cell (equivalently the smallest cell) in $\mathcal{W}$ that contains q by performing a point-location query in the compressed quadtree $\mathcal{T}$. Let $\square$ be this cell. Let,

$$\lambda^* = \min\left(\left\|\mathsf{q}' - \mathrm{nn}'(\square)\right\|_{\oplus}, \beta_k(\square_{\mathsf{rep}}) + \|\mathsf{q} - \square_{\mathsf{rep}}\|\right). \tag{5.2}$$

If $\mathsf{diam}(\square) \leq (\varepsilon/8)\lambda^*$ we return $\mathrm{nn}_k(\square_{\mathsf{rep}})$ as the approximate kth-nearest neighbor, else we return $\mathbf{b}(\square)$.

5.5.2 Correctness

Lemma 5.5.1 *The number* $\lambda^* = \min\left(\left\|\mathsf{q}' - \mathrm{nn}'(\square)\right\|_{\oplus}, \beta_k(\square_{\mathsf{rep}}) + \|\mathsf{q} - \square_{\mathsf{rep}}\|\right)$ *satisfies,* $\mathsf{d}_k(\mathsf{q}, \mathcal{B}) \leq \lambda^*$.

Proof: This follows by the Lipschitz property, see Observation 5.1.2. ■

Lemma 5.5.2 *Let* $\square \in \mathcal{W}$ *be any cell containing* q. *If* $\mathsf{diam}(\square) \leq \varepsilon \mathsf{d}_k(\mathsf{q}, \mathcal{B})/4$, *then* $\mathrm{nn}_k(\square_{\mathsf{rep}})$ *is a valid* $(1 \pm \varepsilon)$*-approximate* k*th-nearest neighbor of* q.

Proof: For the point $\square_{\mathsf{rep}}$, by Observation 5.1.2, we have that

$$\mathsf{d}_k(\square_{\mathsf{rep}}, \mathcal{B}) \leq \mathsf{d}_k(\mathsf{q}, \mathcal{B}) + \|\mathsf{q} - \square_{\mathsf{rep}}\| \leq \mathsf{d}_k(\mathsf{q}, \mathcal{B}) + \mathsf{diam}(\square) \leq (1+\varepsilon/4)\mathsf{d}_k(\mathsf{q}, \mathcal{B}).$$

Therefore, the ball $\mathrm{nn}_k(\square_{\mathsf{rep}})$ satisfies

$$\mathsf{d}(\square_{\mathsf{rep}}, \mathrm{nn}_k(\square_{\mathsf{rep}})) \leq (1+\varepsilon/4)\mathsf{d}_k(\square_{\mathsf{rep}}, \mathcal{B}) \leq (1+\varepsilon/4)^2 \mathsf{d}_k(\mathsf{q}, \mathcal{B}) \leq (1+3\varepsilon/4)\mathsf{d}_k(\mathsf{q}, \mathcal{B}).$$

As such we have that

$$\mathsf{d}(\mathsf{q}, \mathrm{nn}_k(\square_{\mathsf{rep}})) \leq \mathsf{d}(\square_{\mathsf{rep}}, \mathrm{nn}_k(\square_{\mathsf{rep}})) + \|\mathsf{q} - \square_{\mathsf{rep}}\| \leq ((1 + 3\varepsilon/4) + \varepsilon/4)\, \mathsf{d}_k(\mathsf{q}, \mathcal{B}) \leq (1+\varepsilon)\mathsf{d}_k(\mathsf{q}, \mathcal{B})\,.$$

Similarly, using the Lipschitz property, we can argue that, $\mathsf{d}(\mathsf{q}, \mathrm{nn}_k(\square_{\mathsf{rep}})) \geq (1-\varepsilon)\mathsf{d}_k(\mathsf{q}, \mathcal{B})$, and therefore we have, $(1-\varepsilon)\mathsf{d}_k(\mathsf{q}, \mathcal{B}) \leq \mathsf{d}(\mathsf{q}, \mathrm{nn}_k(\square_{\mathsf{rep}})) \leq (1+\varepsilon)\mathsf{d}_k(\mathsf{q}, \mathcal{B})$, and the required guarantees are satisfied. ■

Lemma 5.5.3 *For any point $\mathsf{q} \in \mathbb{R}^d$ there is a quorum ball $\Lambda_i = \mathsf{ball}(\mathsf{w}_i, \mathsf{x}_i)$ such that (A) Λ_i intersects $\mathsf{ball}(\mathsf{q}, \mathsf{d}_k(\mathsf{q}, \mathcal{B}))$, (B) $\mathsf{x}_i \leq 3\xi\mathsf{d}_k(\mathsf{q}, \mathcal{B})$, and (C) $\|\mathsf{q} - \mathsf{w}_i\| \leq 4\xi\mathsf{d}_k(\mathsf{q}, \mathcal{B})$.*

Proof: By assumption, $k > 2\mathsf{c}_d$, and so by Lemma 5.3.1 among the k nearest neighbor of q, there are $k - \mathsf{c}_d$ balls of radius at most $\mathsf{d}_k(\mathsf{q}, \mathcal{B})$. Let $\mathcal{B}'$ denote the set of these balls. Among the indices $1, \ldots, m$, let i be the minimum index such that one of these $k - \mathsf{c}_d$ balls is completely covered by the quorum cluster $\Lambda_i = \mathsf{ball}(\mathsf{w}_i, \mathsf{x}_i)$. Since $\mathsf{ball}(\mathsf{q}, \mathsf{d}_k(\mathsf{q}, \mathcal{B}))$ intersects the ball while Λ_i completely contains it, clearly Λ_i intersects $\mathsf{ball}(\mathsf{q}, \mathsf{d}_k(\mathsf{q}, \mathcal{B}))$. Now consider the time Λ_i was constructed, i.e, the ith iteration of the quorum clustering algorithm. At this time, by assumption, all of $\mathcal{B}'$ was available, i.e., none of its balls were assigned to earlier quorum clusters. The ball $\mathsf{ball}(\mathsf{q}, 3\mathsf{d}_k(\mathsf{q}, \mathcal{B}))$ contains $k - \mathsf{c}_d$ unused balls and touches k balls from $\mathcal{B}$, as such the smallest such ball had radius at most $3\mathsf{d}_k(\mathsf{q}, \mathcal{B})$. By the guarantee on quorum clustering, $\mathsf{x}_i \leq 3\xi\mathsf{d}_k(\mathsf{q}, \mathcal{B})$. As for the last part, as the balls $\mathsf{ball}(\mathsf{q}, \mathsf{d}_k(\mathsf{q}, \mathcal{B}))$ and $\Lambda_i = \mathsf{ball}(\mathsf{w}_i, \mathsf{x}_i)$ intersect, and $\mathsf{x}_i \leq 3\xi\mathsf{d}_k(\mathsf{q}, \mathcal{B})$, we have by the triangle inequality that $\|\mathsf{q} - \mathsf{w}_i\| \leq (1 + 3\xi)\mathsf{d}_k(\mathsf{q}, \mathcal{B}) \leq 4\xi\mathsf{d}_k(\mathsf{q}, \mathcal{B})$, as $\xi \geq 1$. ■

Definition 5.5.4 For a given query point, any quorum cluster that satisfies the conditions of Lemma 5.5.3 is defined to be an ***anchor cluster***. By Lemma 5.5.3 an anchor cluster always exists.

Lemma 5.5.5 *Suppose that among the quorum cluster balls $\Lambda_1, \ldots, \Lambda_m$, there is some ball $\Lambda_i = \mathsf{ball}(\mathsf{w}_i, \mathsf{x}_i)$ which satisfies that $\|\mathsf{q} - \mathsf{w}_i\| \leq 8\xi\mathsf{d}_k(\mathsf{q}, \mathcal{B})$ and $\varepsilon\mathsf{d}_k(\mathsf{q}, \mathcal{B})/4 \leq \mathsf{x}_i \leq 8\xi\mathsf{d}_k(\mathsf{q}, \mathcal{B})$ then the output of the algorithm is correct.*

Proof: We have

$$\frac{32\xi\mathsf{x}_i}{\varepsilon} \geq \frac{32\xi(\varepsilon\mathsf{d}_k(\mathsf{q}, \mathcal{B})/4)}{\varepsilon} = 8\xi\mathsf{d}_k(\mathsf{q}, \mathcal{B}) \geq \|\mathsf{q} - \mathsf{w}_i\|\,.$$

Thus, by construction, the expanded environ of the quorum cluster $\mathsf{ball}(\mathsf{w}_i, \mathsf{x}_i)$ contains the query point, see Eq. (5.1). Let j be the smallest non-negative integer such that $2^j\mathsf{x}_i \geq \mathsf{d}(\mathsf{q}, \mathsf{w}_i)$. We have that, $2^j\mathsf{x}_i \leq$

$\max(\mathsf{x}_i, 2\mathsf{d}(\mathsf{q}, \mathsf{w}_i))$. As such, if $\square$ is the smallest cell in $\mathcal{W}$ containing the query point q, then

$$\begin{aligned}\mathsf{diam}(\square) &\leq \frac{\varepsilon}{\zeta_1} 2^{j+1} \mathsf{x}_i \leq \frac{\varepsilon}{\zeta_1} \cdot \max(2\mathsf{x}_i, 4\mathsf{d}(\mathsf{q}, \mathsf{w}_i)) \leq \frac{\varepsilon}{\zeta_1} \cdot \max\Big(16\xi \mathsf{d}_k(\mathsf{q}, \mathcal{B}), 32\xi \mathsf{d}_k(\mathsf{q}, \mathcal{B})\Big) \\ &\leq \frac{\varepsilon}{8} \mathsf{d}_k(\mathsf{q}, \mathcal{B}),\end{aligned}$$

by Eq. (5.1), and if $\zeta_1 \geq 256\xi$. Now, by Lemma 5.5.1 we have that $\lambda^* \geq \mathsf{d}_k(\mathsf{q}, \mathcal{B})$, so $\mathsf{diam}(\square) \leq (\varepsilon/8)\lambda^*$. Therefore, the algorithm returns $\mathrm{nn}_k(\square_{\mathsf{rep}})$ as the $(1 \pm \varepsilon)$-approximate kth-nearest neighbor, but then by Lemma 5.5.2 it is a correct answer. ■

Lemma 5.5.6 *The query algorithm always outputs a correct approximate answer, i.e., the output ball* b *satisfies* $(1-\varepsilon)\mathsf{d}_k(\mathsf{q}, \mathcal{B}) \leq \mathsf{d}(\mathsf{q}, \mathsf{b}) \leq (1+\varepsilon)\mathsf{d}_k(\mathsf{q}, \mathcal{B})$.

Proof: Suppose that among the quorum cluster balls $\Lambda_1 = \mathsf{ball}(\mathsf{w}_1, \mathsf{x}_1), \ldots, \Lambda_m = \mathsf{ball}(\mathsf{w}_m, \mathsf{x}_m)$, there is some ball Λ_i such that $\|\mathsf{q} - \mathsf{w}_i\| \leq 8\xi \mathsf{d}_k(\mathsf{q}, \mathcal{B})$ and $(\varepsilon/4)\mathsf{d}_k(\mathsf{q}, \mathcal{B}) \leq \mathsf{x}_i \leq 8\xi \mathsf{d}_k(\mathsf{q}, \mathcal{B})$, then by Lemma 5.5.5 the algorithm returns a valid answer. Assume this condition is not satisfied. Let the anchor cluster be $\Lambda = \mathsf{ball}(\mathsf{w}, \mathsf{x})$. Since the anchor cluster satisfies $\|\mathsf{q} - \mathsf{w}\| \leq 4\xi \mathsf{d}_k(\mathsf{q}, \mathcal{B})$ and $\mathsf{x} \leq 3\xi \mathsf{d}_k(\mathsf{q}, \mathcal{B})$, it must be the case that, $\mathsf{x} < (\varepsilon/4)\mathsf{d}_k(\mathsf{q}, \mathcal{B})$. Since the anchor cluster intersects $\mathsf{ball}(\mathsf{q}, \mathsf{d}_k(\mathsf{q}, \mathcal{B}))$, we have that $\|\mathsf{q} - \mathsf{w}\| \leq (1+\varepsilon/4)\mathsf{d}_k(\mathsf{q}, \mathcal{B})$. Thus, $\|\mathsf{q}' - \Lambda'\|_{\oplus} = \|\mathsf{q} - \mathsf{w}\| + \mathsf{x} \leq (1+\varepsilon/2)\mathsf{d}_k(\mathsf{q}, \mathcal{B})$. Let $\square$ be the smallest cell in which q is located. Now consider the point $\mathrm{nn}'(\square) \in \Sigma'$. Suppose it corresponds to the cluster Λ_j, i.e., $\Lambda'_j = \mathrm{nn}'(\square)$. Since $\mathrm{nn}'(\square)$ is a $(1+\varepsilon/8)$-ANN to q among the points of Σ', $\|\mathsf{q}' - \mathrm{nn}'(\square)\|_{\oplus} \leq (1+\varepsilon/8)\|\mathsf{q}' - \Lambda'\|_{\oplus} \leq (1+\varepsilon/8)(1+\varepsilon/2)\mathsf{d}_k(\mathsf{q}, \mathcal{B}) \leq (1+\varepsilon)\mathsf{d}_k(\mathsf{q}, \mathcal{B}) \leq 2\mathsf{d}_k(\mathsf{q}, \mathcal{B}) \leq 8\xi \mathsf{d}_k(\mathsf{q}, \mathcal{B})$. It follows that, $\|\mathsf{q} - \mathsf{w}_j\| \leq 8\xi \mathsf{d}_k(\mathsf{q}, \mathcal{B})$, and $\mathsf{x}_j \leq 8\xi \mathsf{d}_k(\mathsf{q}, \mathcal{B})$. By our assumption, it must be the case that, $\mathsf{x}_j < (\varepsilon/4)\mathsf{d}_k(\mathsf{q}, \mathcal{B})$. Now, there are two cases. Suppose that, $\mathsf{diam}(\square) \leq (\varepsilon/8)\lambda^*$. Then, since we have $\lambda^* \leq \|\mathsf{q}' - \mathrm{nn}'(\square)\|_{\oplus}$ so $\lambda^* \leq 2\mathsf{d}_k(\mathsf{q}, \mathcal{B})$. As such, $\mathsf{diam}(\square) \leq (\varepsilon/4)\mathsf{d}_k(\mathsf{q}, \mathcal{B})$. In this case we return $\mathrm{nn}_k(\square)$ by the algorithm, but the result is correct by Lemma 5.5.2. On the other hand, if we return $\mathbf{b}(\square)$, it is easy to see that $\mathsf{d}(\mathsf{q}, \mathbf{b}(\square)) \leq \|\mathsf{q} - \mathsf{w}_j\| + \mathsf{x}_j \leq (1+\varepsilon)\mathsf{d}_k(\mathsf{q}, \mathcal{B})$. Also, as $\mathbf{b}(\square)$ lies completely inside Λ_j it follows by Observation 5.1.1, that $\mathsf{d}(\mathsf{q}, \mathbf{b}(\square)) \geq \mathsf{d}(\mathsf{q}, \Lambda_j) \geq \|\mathsf{q} - \mathsf{w}_j\| - \mathsf{x}_j \geq (\|\mathsf{q} - \mathsf{w}_j\| + \mathsf{x}_j) - 2\mathsf{x}_j \geq \mathsf{d}_k(\mathsf{q}, \mathcal{B}) - (\varepsilon/2)\mathsf{d}_k(\mathsf{q}, \mathcal{B}) \geq (1-\varepsilon/2)\mathsf{d}_k(\mathsf{q}, \mathcal{B})$, where the second last inequality follows by Lemma 5.5.1. ■

5.5.3 The result

Theorem 5.5.7 *Given a set* $\mathcal{B}$ *of* n *disjoint balls in* $\mathbb{R}^d$*, a number* k*, with* $1 \leq k \leq n$*, and* $\varepsilon \in (0,1)$*, one can preprocess* $\mathcal{B}$*, in* $O\Big(n \log n + \frac{n}{k} C_\varepsilon \log n + \frac{n}{k} C'_\varepsilon\Big)$ *time, where* $C_\varepsilon = O\big(\varepsilon^{-d} \log \varepsilon^{-1}\big)$ *and* $C'_\varepsilon = O\big(\varepsilon^{-2d} \log \varepsilon^{-1}\big)$*. The space used by the data-structure is* $O(C_\varepsilon n/k)$*. Given a query point* q*, this data-structure outputs a ball*

$\mathsf{b} \in \mathcal{B}$ *in* $O\left(\log \frac{n}{k\varepsilon}\right)$ *time, such that* $(1-\varepsilon)\mathsf{d}_k(\mathsf{q},\mathcal{B}) \leq \mathsf{d}(\mathsf{q},\mathsf{b}) \leq (1+\varepsilon)\mathsf{d}_k(\mathsf{q},\mathcal{B})$.

Proof: If $k \leq 2\mathsf{c}_d$ then Theorem 5.3.5 provides the desired result.

For $k > 2\mathsf{c}_d$, the correctness was proved in Lemma 5.5.6. We only need to bound the construction time and space as well as the query time. Computing the quorum clustering takes time $O(n \log n)$ by Lemma 5.4.6. Observe that $|\mathcal{I}| = O\left(\frac{n}{k\varepsilon^d} \log \frac{1}{\varepsilon}\right)$. From the construction of Arya and Malamatos [AM02], we have $|\mathcal{S}| = O\left(\frac{n}{k\varepsilon^d} \log \frac{1}{\varepsilon}\right)$ (note, that since we clip the construction to a hyperplane, we get $1/\varepsilon^d$ in the bound and not $1/\varepsilon^{d+1}$). A careful implementation of this stage takes time $O\big(n \log n + |\mathcal{W}|\big(\log n + \frac{1}{\varepsilon^{d-1}}\big)\big)$. Overlaying the two compressed quadtrees representing them takes linear time in their size, that is $O(|\mathcal{I}| + |\mathcal{S}|)$.

The most expensive step is to perform the $(1 \pm \varepsilon/4)$-approximate kth-nearest neighbor query for each cell in the resulting decomposition of $\mathcal{W}$, see Eq. (5.2) (i.e., computing $\beta_k(\square_{\mathsf{rep}})$ for each cell $\square \in \mathcal{W}$). Using the data-structure of Section 5.3 (see Theorem 5.3.5) each query takes $O\big(\log n + 1/\varepsilon^d\big)$ time.

$$O\left(n \log n + |\mathcal{W}|\left(\log n + \frac{1}{\varepsilon^d}\right)\right) = O\left(n \log n + \frac{n}{k\varepsilon^d} \log \frac{1}{\varepsilon} \log n + \frac{n}{k\varepsilon^{2d}} \log \frac{1}{\varepsilon}\right)$$

time, and this bounds the overall construction time.

The query algorithm is a point location query followed by an $O(1)$ time computation and takes time $O\left(\log\left(\frac{n}{k\varepsilon}\right)\right)$. ∎

Note that the space decomposition generated by Theorem 5.5.7 can be interpreted as a space decomposition of complexity $O(C_\varepsilon n/k)$, where every cell has two input balls associated with it, which are the candidates to be the desired (k,ε)-ANN. That is, Theorem 5.5.7 compute a $(k.\varepsilon)$-AVD of the input balls.

5.6 Conclusions

In this chapter, we presented a generalization of the usual $(1 \pm \varepsilon)$-approximate kth-nearest neighbor problem in $\mathbb{R}^d$, where the input are balls of arbitrary radius, while the query is a point. We first presented a data structure that takes $O(n)$ space, and the query time is $O(\log n + \varepsilon^{-d})$. Here, both k and ε could be supplied at query time. Next we presented an (k,ε)-AVD taking $\widetilde{O}(n/k)$ space. Thus showing, surprisingly, that the problem can be solved in sublinear space if k is sufficiently large.

Chapter 6

Generalized Proximity Search

6.1 Introduction

Given a set of functions $\mathcal{F} = \left\{ f_i : \mathbb{R}^d \to \mathbb{R} \;\middle|\; i = 1, \ldots, n \right\}$, their minimization diagram is the function $f_{\min}(\mathsf{q}) = \min_{i=1,\ldots,n} f_i(\mathsf{q})$, for any $\mathsf{q} \in \mathbb{R}^d$. By viewing the graphs of these functions as manifolds in $\mathbb{R}^{d+1}$, the graph of the minimization diagram, also known as the ***lower envelope*** of $\mathcal{F}$, is the manifold that can be viewed by an observer at $-\infty$ on the x_{d+1} axis. Given a set of functions $\mathcal{F}$ as above, many problems in Computational Geometry can be viewed as computing the minimization diagram; that is, one preprocesses $\mathcal{F}$, and given a query point q, one needs to compute $f_{\min}(\mathsf{q})$ quickly. This typically requires $n^{O(d)}$ space if one is interested in logarithmic query time. If one is restricted to using linear space, then the query time deteriorates to $O\big(n^{1-O(1/d)}\big)$ [Mat92, Cha10]. There is substantial work on bounding the complexity of the lower envelope in various cases, how to compute it efficiently, and performing range search on them; see the book by Sharir and Agarwal [SA95].

Nearest-neighbor. One natural problem that falls into this framework is the nearest-neighbor (NN) search problem. Here, given a set P of n data points in a metric space $\mathcal{X}$, we need to preprocess P, such that given a query point $\mathsf{q} \in \mathcal{X}$, one can find (quickly) the point $\mathsf{nn}_{\mathsf{q}} \in \mathsf{P}$ closest to q. Nearest-neighbor search is a fundamental task used in numerous domains including machine learning, clustering, document retrieval, databases, statistics, and many others.

To see the connection to lower envelopes, consider a set of data points $\mathsf{P} = \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}$ in $\mathbb{R}^d$. Next, consider the set of functions $\mathcal{F} = \{f_1, \ldots, f_n\}$, where $f_i(\mathsf{q}) = \|\mathsf{q} - \mathsf{p}_i\|$, for $i = 1, \ldots, n$. The graph of f_i is the set of points $\left\{ (\mathsf{q}, f_i(\mathsf{q})) \mid \mathsf{q} \in \mathbb{R}^d \right\}$ (which is a cone in $\mathbb{R}^{d+1}$ with apex at $(\mathsf{p}_i, 0)$). Clearly the NN search problem is to evaluate the minimization diagram of these functions at a query point q, and to find the function f_i which achieves the minimum at q.

More generally, given a set of n functions, one can think of the functions as defining distances, and the minimization diagram defining a "nearest-neighbor" under those distance functions, by analogy with the

above. The nearest-neighbor distance of a query point here, is simply the "height" of the lower envelope at that point.

Generalized distance functions: motivation. The algorithms for approximate nearest-neighbor extend to various metrics in $\mathbb{R}^d$, for example the well known ℓ_p metrics. In particular, previous constructions of AVD's extend to ℓ_p metrics [Har01, AM02] as well. However, these constructions fail even for a relatively simple and natural extension; specifically, multiplicative weighted Voronoi diagrams. Here, every site p, in the given point set P, has a weight ω_{p}, and the "distance" of a query point q to p is $f_{\mathsf{p}}(\mathsf{q}) = \omega_{\mathsf{p}} \|\mathsf{p} - \mathsf{q}\|$. As with ordinary Voronoi diagrams, one can define the weighted Voronoi diagram as a partition of space into disjoint regions, one for each site p, such that in the region for p, the function f_{p} is the one realizing the minimum among all the functions induced by the points of P. Such a weighted Voronoi diagram is known as the *multiplicative Voronoi diagram*, and even in the plane it can have quadratic complexity. Intuitively, such multiplicative Voronoi diagrams can be used to model facilities, where the price of delivery to a client depends on the facility and the distance to the client. Of course, this is only one possible distance function, and there are many other such functions that are of interest.

When is fast proximity and small space not possible? Consider a set of segments in the plane, and we are interested in the nearest segment to a query point. Given n such segments and n such query points, this is an extension of Hopcroft's problem, which requires only to decide if there is any of the given points on any of the segments. There are lower bounds (in reasonable computation models) which show that a solution to Hopcroft's problem requires $\Omega\big(n^{4/3}\big)$ time [Eri96]. This implies that no multiplicative-error approximation for proximity search in this case is possible, if one insists on near-linear preprocessing, and logarithmic query time.

When is fast ANN possible? So, consider a set of geometric objects where each one of them induces a natural distance function, measuring how far a point in space is from this object. Given such a collection of functions, the nearest-neighbor for a query point is simply the function that defines the lower envelope "above" the query point (i.e., the object closest to the query point under its distance function). Clearly, this approach allows a generalization of the proximity search problem. In particular, the above question becomes, for what classes of functions, can the lower envelope be approximated up to $(1+\varepsilon)$-multiplicative error, in logarithmic time? Here the preprocessing space used by the data-structure should be near-linear.

6.2 The framework and summary of results

For the sake of simplicity of exposition, throughout this chapter we assume that all the "action" takes place in the unit cube $[0,1]^d$. Among other things, this implies that all the queries are in this region. This can always be guaranteed by an appropriate scaling and translation of space. The scaling and translation, along with the conditions on functions in our framework, implies that outside the unit cube the approximation to the lower envelope can be obtained in constant time.

6.2.1 Problem statement

We are given a set $\mathcal{F}$ of n functions from $\mathbb{R}^d$ to $\mathbb{R}^+$, and a parameter ε. Our task is to build a data-structure of near-linear size, such that given a query point q, one can quickly compute a value x, such that $\mathbb{d}(\mathsf{q}) \leq x \leq (1+\varepsilon)\mathbb{d}(\mathsf{q})$, where

$$\mathbb{d}(\mathsf{q}) = \mathbb{d}(\mathsf{q}, \mathcal{F}) = \min_{i=1,\ldots,n} f_i(\mathsf{q}) \tag{6.1}$$

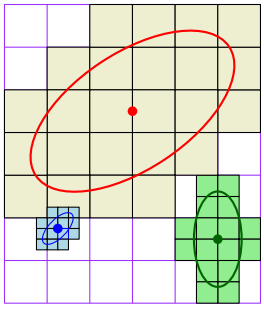
is the height of the lower envelope of $\mathcal{F}$ at q; that is, $\mathbb{d}(\mathsf{q})$ is the ***distance*** of q from $\mathcal{F}$. Since the functions of $\mathcal{F}$ are distance functions in our applications, we refer to these approximate minimization diagram queries as ***approximate nearest neighbor*** (**ANN**) queries.

For such a set of functions $\mathcal{F}$, an ***approximate Voronoi diagram*** (**AVD**) is a decomposition of space into near-linear number of cells, each of low descriptive complexity, such that every cell c in the AVD has a small number of functions $\mathcal{G}_{\mathsf{c}}$ associated with it, such that for any query point q, if c is the cell of the AVD that contains q, then $\mathbb{d}(\mathsf{q}, \mathcal{G}_{\mathsf{c}})$ approximates $\mathbb{d}(\mathsf{q}, \mathcal{F})$ well. As such, given an ANN query, it can be answered by doing a point-location query in the AVD to compute c, and then computing $\mathbb{d}(\mathsf{q}, \mathcal{G}_{\mathsf{c}})$ explicitly. Usually, such an AVD is constructed using compressed quadtrees and yields an $O(\log n)$ time ANN queries, see [Har11] for details.

6.2.2 Informal description of the technique

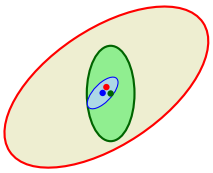
Consider n points in the plane $\mathsf{p}_1, \ldots, \mathsf{p}_n$, where the "distance" from the ith point to a query q, is the minimum scaling of an ellipse $\mathcal{E}_i$ (centered at p_i), till it covers q, and let f_i denote this distance function. Assume that these ellipses are fat, i.e., for some constant $\alpha > 0$, there is a ball of some radius r_i, centered at p_i, that lies inside $\mathcal{E}_i$, while the ball of radius αr_i covers $\mathcal{E}_i$. Clearly each function f_i has a graph that is a deformed cone. Given a query point $\mathsf{q} \in \mathbb{R}^2$, we are interested in the first function graph being hit by a vertical ray shot upward from $(\mathsf{q}, 0)$. In particular, let $\mathbb{d}(\mathsf{q}) = \min_{i=1,\ldots,n} f_i(\mathsf{q})$ be the height of the

minimization diagram of these functions at q.



As a first step in computing $\mathrm{d}(\mathsf{q})$, consider the decision version of this problem. Given a value r, we are interested in deciding if $\mathrm{d}(\mathsf{q}) \leq r$. That is, we want to decide if $\mathsf{q} \in \bigcup_i (\mathsf{p}_i + r\mathcal{E}_i)$. Of course, this is by itself a computationally expensive task, and as such we satisfy ourselves with an approximate decision procedure. Formally, we replace every ellipse by a collection of grid cells (of the right resolution), such that approximately it is enough to decide if the query point lies inside any of these grid cells – if it does, we know that $\mathrm{d}(\mathsf{q}) \leq (1+\varepsilon)r$, otherwise $\mathrm{d}(\mathsf{q}) > r$. Of course, as depicted in the right, since the ellipses are of different sizes, the grid cells generated for each ellipse might belong to different resolutions, and might be of different sizes. Nevertheless, one can perform this point-location query among the marked grid squares quickly using a compressed quadtree.

If we were interested only in the case where $\mathrm{d}(\mathsf{q})$ is guaranteed to be in some interval $[\alpha, \beta]$, then the problem would be easily solvable. Indeed, build a sequence of the above deciders $\mathcal{D}_1, \ldots, \mathcal{D}_m$, where $\mathcal{D}_i$ is for the distance $(1+\varepsilon)^i\alpha$, and $m = \log_{1+\varepsilon}(\beta/\alpha)$. Clearly, doing a binary search over these deciders would resolve the distance query.



Sketchable. Unfortunately, in general, there is no such guarantee – this makes the problem significantly more challenging. Fortunately, for truly "large" distances, a collection of such ellipses looks like a constant number of ellipse (at least in the approximate case). In the example of the figure above, for large enough distance, the ellipses looks like a single ellipse, as demonstrated in the figure on the right. Slightly more formally, if $\bigcup_i (\mathsf{p}_i + r\mathcal{E}_i)$ is connected, then the set $\bigcup_i (\mathsf{p}_i + R\mathcal{E}_i)$ can be $(1+\varepsilon)$-approximated by a constant number of these ellipses, if $R > \Omega(nr/\varepsilon)$. A family of functions having this property is *sketchable*. This suggests the problem is easy for very large distances.

Critical values to search over. The above suggests that connectivity is the underlying property that enables us to simplify and replace a large set of ellipses, by a few ellipses, if we are looking at them from sufficiently far. This implies that the critical values, when the level-set of the functions changes its connectivity, are the values we should search over during the nearest-neighbor search. Specifically, let r_i be the minimal r when the set $\bigcup_{k=1}^{n} (\mathsf{p}_k + r_i\mathcal{E}_k)$ has $n-i$ connected components, for $i = 0, 1, \ldots, n-1$, and let $0 = r_0 < r_1 \leq r_2 \leq \cdots \leq r_{n-1} < r_n = \infty$, be the resulting sequence. Using the above decision procedure, and a binary search, we can find the largest index j, such that $r_j \leq \mathrm{d}(\mathsf{q}) < r_{j+1}$. Furthermore, the decision procedure for the distance r_{j+1}, reports which connected components of $\bigcup_{k=1}^{n} (\mathsf{p}_k + r_j\mathcal{E}_k)$ contains the query

point q in their union, at distance r_{j+1}. Assume the set of these connected component is formed by the first t functions; that is, $\bigcup_{k=1}^{t}(\mathsf{p}_k + r_{j+1}\mathcal{E}_k)$ contains q. There are two possibilities:

(A) If $\mathrm{d}(\mathsf{q}) \in \left[r_j, c_a(t/\varepsilon)r_j\right]$, then a binary search with the decision procedure would approximate $\mathrm{d}(\mathsf{q})$, where c_a is some constant.

(B) If $\mathrm{d}(\mathsf{q}) > (t/\varepsilon)r_j$ then, each connected component of $\bigcup_{k=1}^{t}(\mathsf{p}_k + r_j\mathcal{E}_k)$ can be sketched and replaced by a constant number of representative functions (the sketch), and the nearest-neighbor search can now be restricted to the union of the sketches. This is an instance of the problem with a smaller number of functions.

Challenges

There are several challenges in realizing the above scheme:

(A) We are interested in more general distance functions. To this end, we carefully formalize what conditions the underlying distance functions induced by each point has to fulfill, so that our framework applies.

(B) The above scheme requires (roughly) quadratic space to be realized. To reduce the space to near-linear, we need be more aggressive about replacing clusters of points/functions, by sketches. To this end, we replace our global scheme by a recursive scheme that starts with the "median" critical value, and fork the search at this value using the decision procedure. Now, when continuing the search above this value, we replace every cluster (at this resolution) by its sketch.

(C) Computing this "median" value directly is too expensive. Instead, we randomly select a function and compute the connectivity radius of this single distance function with the remaining functions. With good probability this value turns out to be good.

(D) We need to be very careful to avoid accumulation in the error as we replace clusters by sketches.

6.2.3 Notations and basic definitions

Given $\mathsf{q} \in \mathbb{R}^d$ and $\mathsf{P} \subseteq \mathbb{R}^d$ a non-empty closed set, the ***distance*** of q to P is $\mathsf{d}(\mathsf{q}, \mathsf{P}) = \min_{x \in \mathsf{P}} \|\mathsf{q} - x\|$.

Definition 6.2.1 For $\ell \geq 0$ and a function $f : \mathbb{R}^d \to \mathbb{R}$, the ℓ ***sublevel set*** of f is the set $f_{\preceq \ell} = \left\{ \mathsf{q} \in \mathbb{R}^d \;\middle|\; f(\mathsf{q}) \leq \ell \right\}$. The number ℓ will be the ***level*** of the sublevel set $f_{\preceq \ell}$. For a set of functions $\mathcal{F}$, let $\mathcal{F}_{\preceq \ell} = \bigcup_{f \in \mathcal{F}} f_{\preceq \ell}$.

For a value r, the approximate versions of the sublevel sets are $f_{\preceq \ell, \approx r} = \mathsf{G}_{\approx r}(f_{\preceq \ell})$ and $\mathcal{F}_{\preceq \ell, \approx r} = \cup_{f \in \mathcal{F}} \mathsf{G}_{\approx r}(f_{\preceq \ell})$, see Definition 2.5.3.

Definition 6.2.2 Given a function f and $\mathsf{q} \in \mathbb{R}^d$, their distance is $\mathrm{d}(\mathsf{q}, f) = f(\mathsf{q})$. Given two functions f and g, their ***distance*** $\mathrm{d}(f, g)$ is the minimum $\ell \geq 0$ such that $f_{\preceq \ell} \cap g_{\preceq \ell} \neq \emptyset$. Similarly, for two sets of function, $\mathcal{F}$ and $\mathcal{G}$, their ***distance*** is $\mathrm{d}(\mathcal{F}, \mathcal{G}) = \min_{f \in \mathcal{F}, g \in \mathcal{G}} \mathrm{d}(f, g)$. See also Eq. (6.1).

Example 6.2.3 *To decipher these somewhat cryptic definitions, the reader might want to consider the standard settings of regular Voronoi diagrams. Here, we have a set P of n points. The ith point $\mathsf{p}_i \in \mathsf{P}$ induces the natural function $f_i(\mathsf{q}) = \|\mathsf{p}_i - \mathsf{q}\|$. We have:*

(A) The graph of f_i in $\mathbb{R}^{d+1}$ is a cone "opening upwards" with an apex at $(\mathsf{p}_i, 0)$.

(B) The ℓ sublevel set of f_i (i.e., $(f_i)_{\preceq \ell}$) is a ball of radius ℓ centered at p_i.

(C) The distance of q from f_i, i.e., $\mathrm{d}(\mathsf{q}, f_i)$, is the Euclidean distance between q and p_i.

(D) Consider two subsets of points $X, Y \subseteq \mathsf{P}$ and let $\mathcal{F}_X$ and $\mathcal{F}_Y$ be the corresponding sets of functions. The distance $\ell = \mathrm{d}(\mathcal{F}_X, \mathcal{F}_Y)$ is the minimum radius of balls centered at points of X and Y, such that there are two balls, from the two sets, that intersect; that is, ℓ is half the minimum distance between a point of X and a point of Y. In particular, if the union of balls of radius ℓ centered at X is connected, i.e., $(\mathcal{F}_X)_{\preceq \ell}$ is connected, and similarly for Y, then $(\mathcal{F}_X \cup \mathcal{F}_Y)_{\preceq \ell}$ is connected. This is the critical value where two connected components of the sublevel sets merge.

The distance function behaves to some extent like one would expect an usual metric to behave: (i) $\mathrm{d}(f, g)$ always exists, and (ii) (symmetry) $\mathrm{d}(f, g) = \mathrm{d}(g, f)$, Also, we have $f_{\preceq \mathrm{d}(f,g)} \neq \emptyset$. Note that the triangle inequality does not hold for $\mathrm{d}(\cdot, \cdot)$.

Observation 6.2.4 *Suppose that f and g are two functions such that $\mathrm{d}(f, g) > 0$ and $\mathsf{q} \in \mathbb{R}^d$. Then, $\max(\mathrm{d}(\mathsf{q}, f), \mathrm{d}(\mathsf{q}, g)) \geq \mathrm{d}(f, g)$.*

Definition 6.2.5 Let $B_1, B_2, \ldots, B_m$ be connected, nonempty sets in $\mathbb{R}^d$. This collection of sets is ***connected*** if $\cup_i B_i$ is connected.

Definition 6.2.6 An infinite family of functions $\mathfrak{F}$ from $\mathbb{R}^d$ to $\mathbb{R}^+$, is ***well behaved***, if for any m functions $\{f_1, \ldots, f_m\}$ chosen from this family, their graphs as surfaces in $\mathbb{R}^{d+1}$, can be preprocessed in $m^{\gamma \cdot d}$ time, so that the projection of their lower envelope is computed explicitly, as a partition of space into regions of the same total complexity, and these regions can be preprocessed for point-location queries in the same time and space, (i.e., $m^{\gamma d}$), and it can answer point-location queries in $O(\log m)$ time. Here, the constant γ is termed as the ***wellness constant***.

Sketches

A key idea underlying our approach is that any set of functions of interest should look like a single (or a small number of functions) from "far" enough. Indeed, given a set of points $\mathsf{P} \subseteq \mathbb{R}^d$, they look like a single point (as far as distance), if the distance from $\mathcal{CH}(\mathsf{P})$ is at least $2\mathsf{diam}(\mathsf{P})/\varepsilon$.

Definition 6.2.7 ($\ell_{\mathbf{con}}(\mathcal{F})$) Given a set of functions $\mathcal{G}$, if $\mathcal{G}$ contains a single function then the ***connectivity level*** $\ell_{\mathrm{con}}(\mathcal{G})$ is 0; otherwise, it is the minimum $\ell \geq 0$, such that the collection of sets $f_{\preceq \ell}$ for $f \in \mathcal{G}$ is connected, see Definition 6.2.5.

Remark 6.2.8 *It follows from Definition 6.2.7 that at level $\ell = \ell_{\mathrm{con}}(\mathcal{G})$, each of the sets $f_{\preceq \ell}$ for $f \in \mathcal{G}$, are nonempty and connected, and further their union $\mathcal{G}_{\preceq \ell}$ is also connected. This can be relaxed to require that the intersection graph of the sets $f_{\preceq \ell}$ for $f \in \mathcal{G}$ is connected (this also implies they are nonempty). Notice that, if at level ℓ, the sublevel sets are connected, then the relaxed definition is equivalent to Definition 6.2.7. However, the relaxed definition introduces more technical baggage, and for all the interesting applications we have, the sublevel sets $f_{\preceq y}$ are connected at all levels y at which they are nonempty. Therefore, in the interest of brevity, and to keep the presentation simple, we mandate that the sublevel sets be connected at ℓ. Furthermore, we assume that the sublevel sets are connected whenever nonempty.*

Definition 6.2.9 Given a set of functions $\mathcal{G}$ and $\delta \geq 0, y_0 \geq 0$, a ***$(\delta, y_0)$-sketch*** for $\mathcal{G}$ is a (hopefully small) subset $\mathcal{H} \subseteq \mathcal{G}$, such that $\mathcal{G}_{\preceq y} \subseteq \mathcal{H}_{\preceq (1+\delta)y}$, for all $y \geq y_0$.

It is easy to see that for any $\mathcal{G}, \delta \geq 0, y_0 \geq 0$, if $\mathcal{H} \subseteq \mathcal{G}$ is a (δ, y_0)-sketch, then for any $\delta' \geq \delta, y_0' \geq y_0, \mathcal{H}' \supseteq \mathcal{H}$ it is true that $\mathcal{H}'$ is a (δ', y_0')-sketch for $\mathcal{G}$. Trivially, for any $\delta \geq 0, y_0 \geq 0$, it is true that $\mathcal{H} = \mathcal{G}$ is a (δ, y_0)-sketch.

6.2.4 Conditions on the functions

A infinite family of functions $\mathfrak{F}$ from $\mathbb{R}^d$ to $\mathbb{R}^+$, is ***dependable***, if there are absolute constants, $\zeta > 0$ – the ***growth constant***, and a positive integer c_{sk} – the ***sketch constant***, depending on $\mathfrak{F}$, such that the following conditions are satisfied for any $f \in \mathfrak{F}$:

(P1) **Compactness.** For any $y \geq 0$ the sublevel set $(f)_{\preceq y}$ is compact.

(P2) **Bounded growth.** There is a function $\lambda_f : \mathbb{R}^+ \to \mathbb{R}^+$, called the ***growth function***, such that for any $y \geq 0$ and $\varepsilon > 0$, if $f_{\preceq y} \neq \emptyset$, then $\lambda_f(y) \geq \mathsf{diam}(f_{\preceq y})/\zeta$, and such that if $\mathsf{q} \in \mathbb{R}^d$ with $\mathsf{d}(\mathsf{q}, f_{\preceq y}) \leq \varepsilon \lambda_f(y)$, then $f(\mathsf{q}) \leq (1+\varepsilon)y$. This is equivalent to, $f_{\preceq y} \oplus \mathsf{ball}(0, \varepsilon\lambda_f(y)) \subseteq f_{\preceq (1+\varepsilon)y}$, where $\mathsf{ball}(\mathsf{u}, r)$ is the ball of radius r centered at u.

(P3) **Existence of a sketch.** Given $\delta > 0$ and a finite subset $\mathcal{G} \subseteq \mathfrak{F}$, there is a $\mathcal{H} \subseteq \mathcal{G}$ with $|\mathcal{H}| = O\Big(1/\delta^{\mathsf{c}_{\mathrm{sk}}}\Big)$ and $y_0 = O\big(\ell_{\mathrm{con}}(\mathcal{G}) \cdot \big(|\mathcal{G}|/\delta\big)^{\mathsf{c}_{\mathrm{sk}}}\big)$ such that, $\mathcal{H}$ is a (δ, y_0)-sketch for $\mathcal{G}$.

We also require some straightforward properties from the computation model:

(C1) For all $\mathsf{q} \in \mathbb{R}^d$, the value $f(\mathsf{q}) = \mathrm{d}(\mathsf{q}, f)$ is computable in $O(1)$ time.

(C2) For any $y \geq 0, r > 0$, the set of grid cells approximating the sublevel set $f_{\preceq r}$ of f, that is $f_{\preceq y, \approx r} = \mathsf{G}_{\approx r}(f_{\preceq y})$ (see Definition 2.5.3), is computable in linear time in its size.

(C3) For any $f, g \in \mathfrak{F}$, the distance $\mathrm{d}(f, g)$ is computable in $O(1)$ time.

We also assume that the growth function $\lambda_f(y)$, from Condition (P2), can be computed quickly, i.e., in $O(1)$ time.

Remark *(A) For all our examples, we will specify a set of n functions $\mathcal{F} = \{f_1, \ldots, f_n\}$, from the implicit infinite family $\mathfrak{F}$. In this case the growth constant and the sketch constant would be absolute constants not depending on n, or the chosen set of functions $\mathcal{F}$. Henceforth, we will use the term family (or set, or class) of functions, for our set of n functions $\mathcal{F}$: the infinite family $\mathfrak{F}$ is only implicitly defined.*

(B) In the following, many of the quantities used above, like the resolution r, or distance involved, would be computed approximately, as exact computation is both computationally expensive and not necessary to make things work.

Remark 6.2.11 *We will use Condition (C2) for a given y only for $r = \Omega(\varepsilon \lambda_f(y))$. That is, we will use a grid on the sublevel set at a resolution typically ε times its growth function value at its level, which by Condition (C2) is also $\Omega(\varepsilon \mathsf{diam}(f_{\preceq y}))$. Here $\varepsilon \in (0, 1)$ is a prespecified approximation parameter. As such, the number of grid cells in the approximation $f_{\preceq y, \approx r}$, for a single function f level set, is $O\big(1/\varepsilon^d\big)$. The value of r being used can be somewhat imprecise as long as it is small enough.*

Properties

In the following, let $\mathcal{F}$ be a family of functions, that satisfies the conditions above. The functions under consideration have the basic properties listed below. Since these properties are straightforward but their proofs are somewhat tedious, we present their proofs in Appendix A.

(L1) 0-level is empty or a point. For any $f \in \mathcal{F}$, either $f_{\preceq 0} = \emptyset$ or $f_{\preceq 0}$ consists of a single point, see Lemma A.0.1.

(L2) Connectivity level for more than one function is non-zero. If $\ell_{\mathrm{con}}(\mathcal{F}) = 0$ for any non-empty set $\mathcal{F}$, then $|\mathcal{F}| = 1$. See Definition 6.2.7 and Observation A.0.2.

(L3) NEAR CONVEXITY. Let $f \in \mathcal{F}$ and $y \geq 0$. For any points $\mathsf{u}, \mathsf{v} \in f_{\preceq y}$, we have $\mathsf{uv} \subseteq \mathcal{F}_{\preceq (1+\zeta/2)y}$, where uv denotes the segment joining u to v, see Lemma A.0.3.

(L4) CONNECTIVITY LEVEL IS PRESERVED UNDER SKETCHING. For any $\mathcal{G} \subseteq \mathcal{F}$, $\delta \geq 0$ and $y \geq 0$, such that $\mathcal{G}$ is a (δ, y)-sketch for $\mathcal{F}$, we have that, $\ell_{\mathrm{con}}(\mathcal{G}) \leq (1+\delta)(1+\zeta/2)\max(y, \ell_{\mathrm{con}}(\mathcal{F}))$, see Lemma A.0.5.

(L5) DISTANCES ARE PRESERVED UNDER SKETCHING. Let $\mathcal{G} \subseteq \mathcal{F}$, such that $\mathcal{G}$ is a (δ, y_0)-sketch for $\mathcal{F}$, for some $\delta \geq 0$ and $y_0 \geq 0$. Let q be a point such that $\mathbb{d}(\mathsf{q}, \mathcal{F}) \geq y_0$. Then we have that $\mathbb{d}(\mathsf{q}, \mathcal{G}) \leq (1+\delta)\mathbb{d}(\mathsf{q}, \mathcal{F})$, see Lemma A.0.6.

Remark 6.2.12 (Computing the sketch) *We implicitly assume that the above relevant quantities can be computed efficiently. For example, given some $\delta > 0$, and y_0 as per the bound in condition (P3), a (δ, y_0)-sketch can be computed in time $O(|\mathcal{G}|/\delta^{\mathsf{c}_{\mathrm{sk}}})$ time.*

6.2.5 Summary of results

Our main result is the following, the details of which are presented in Section 6.3.

Theorem 6.2.13 *Let $\mathcal{F}$ be a set of n functions in $\mathbb{R}^d$ that complies with the assumptions of Section 6.2.4, and has sketch constant $\mathsf{c}_{\mathrm{sk}} \geq d+1$. Then, one can build a data-structure to answer ANN for this set of functions, with the following properties:*

(A) The query time is $O(\log n + 1/\varepsilon^{\mathsf{c}_{\mathrm{sk}}})$.

(B) The preprocessing time is $O\big(n\varepsilon^{-2\mathsf{c}_{\mathrm{sk}}} \log^{2\mathsf{c}_{\mathrm{sk}}+1} n\big)$.

(C) The space used is $O\big(n\varepsilon^{-d-1-\mathsf{c}_{\mathrm{sk}}} \log^{\mathsf{c}_{\mathrm{sk}}+1} n\big)$.

One can transform the data-structure into an AVD, and in the process improve the query time (the space requirement slightly deteriorates). See Section 6.3 for details.

Corollary 6.2.14 *Let $\mathcal{F}$ be a set of n functions in $\mathbb{R}^d$ that complies with the assumptions of Section 6.2.4, and has sketch constant $\mathsf{c}_{\mathrm{sk}} \geq d+1$. Moreover, suppose that the family of functions satisfies the well-behaveness condition, see Definition 6.2.6, with wellness constant γ. Then, one can build a data-structure to answer ANN for this set of functions, with the following properties:*

(A) The improved query time is $O(\log(n/\varepsilon))$.

(B) The preprocessing time is $O\big(n\varepsilon^{-2\mathsf{c}_{\mathrm{sk}}} \log^{2\mathsf{c}_{\mathrm{sk}}+1} n + n\varepsilon^{-\gamma d\mathsf{c}_{\mathrm{sk}}}\big)$.

(C) The space used is $S = O\big(n\varepsilon^{-d-1-\mathsf{c}_{\mathrm{sk}}} \log^{\mathsf{c}_{\mathrm{sk}}+1} n + n\varepsilon^{-\gamma d\mathsf{c}_{\mathrm{sk}}}\big)$.

In particular, we can compute an AVD of complexity $O(S)$, for the given functions. That is, one can compute a space decomposition, such that every region has a single function associated with it, and for any

point in this region, this function is a $(1+\varepsilon)$-ANN among the functions of $\mathcal{F}$. Here, a region is either a cube, or the set difference of two cubes, or the intersection of such a region with the region which is a cell in the projection of the lower envelope of some of the functions.

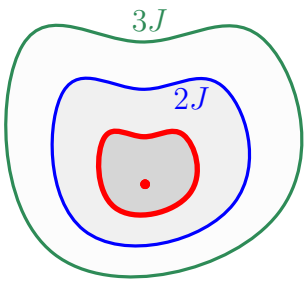
Distance functions for which the framework applies

Multiplicative distance functions with additive offsets. For $i = 1, \ldots, n$, we are given a point p_i, with an associated weight $w_i > 0$, and an ***offset*** $\alpha_i \geq 0$. The ***multiplicative distance with offset*** induced by the ith point is $f_i(\mathsf{q}) = w_i \|\mathsf{q} - \mathsf{p}_i\| + \alpha_i$. In Section 6.4.1 we prove that these distance functions satisfy the conditions of Section 6.2.4, and in particular we get the following result.

Theorem 6.2.15 *Consider a set P of n points in $\mathbb{R}^d$, where the ith point p_i has additive weight $\alpha_i \geq 0$ and multiplicative weight $w_i > 0$, for $i = 1, \ldots, n$. Such a point induces the additive/multiplicative distance function $f_i(\mathsf{q}) = w_i \|\mathsf{q} - \mathsf{p}_i\| + \alpha_i$. Then one can compute a $(1+\varepsilon)$-AVD for these distance functions, with near-linear space complexity, and logarithmic query time. See Theorem 6.2.13 and Corollary 6.2.14, for the exact bounds, where we can use the sketch constant $\mathsf{c}_{\mathrm{sk}} = d+1$, and the wellness constant $\gamma = 1$.*

Scaling distance. Somewhat imprecisely, a connected body J centered at a point ρ is α-rounded fat if it is α-fat (that is, there is radius r such that $\mathsf{ball}(\rho, r) \subseteq J \subseteq \mathsf{ball}(\rho, \alpha r)$), and from any point p on the boundary of J the "cone" $\mathcal{CH}(\mathsf{ball}(\rho, r) \cup \mathsf{p})$ is contained inside J (i.e., every boundary point sees a large fraction of the "center" of the object). We also assume that the boundary of each object J has constant descriptive complexity. Note that such an object is star-shaped with ρ as its center. See Definition 6.4.5 for the formal definition.

For such an object J, its ***scaling distance*** to a point q, is the minimum t, such that $\mathsf{q} \in tJ$ (where the scaling of J is done around its center ρ), see figure on the right. Given n α-rounded fat objects, it is natural to ask for the Voronoi diagram induced by their scaling distance.



The distance functions induced by such a set of objects complies with the framework of Section 6.2.4, see Section 6.4.2 for details. As such, we get the following result.

Theorem 6.2.16 *Consider a set $\mathcal{J}$ of n α-rounded fat objects in $\mathbb{R}^d$, for some constant α. Then one can compute a ANN data-structure, for the scaling distance functions induced by $\mathcal{J}$, with near-linear space complexity, and logarithmic query time. See Theorem 6.2.13 for the exact bounds, where the sketch constant is $\mathsf{c}_{\mathrm{sk}} = d+1$.*

Nearest furthest-neighbor. For a set of points $\mathsf{S} \subseteq \mathbb{R}^d$ and a point q, the ***furthest-neighbor distance*** of q from S, is $\mathsf{f}_{\mathsf{S}}(\mathsf{q}) = \max_{\mathsf{s}\in\mathsf{S}} \|\mathsf{s} - \mathsf{q}\|$; that is, it is the furthest one might have to travel from q to arrive to a point of S. For example, S might be the set of locations of facilities, where it is known that one of them is always open, and one is interested in the worst case distance a client has to travel to reach an open facility. The function $\mathsf{f}_{\mathsf{S}}(\cdot)$ induces a partition of space into the ***furthest-neighbor Voronoi*** diagram, and while its worst case combinatorial complexity is similar to the regular Voronoi diagram, it can be approximated using a constant sized representation (in constant dimensions), see [Har99].

Given n sets of points $\mathsf{P}_1, \ldots, \mathsf{P}_n$ in $\mathbb{R}^d$, we are interested in the distance function $\mathsf{f}(\mathsf{q}) = \min_i \mathsf{f}_i(\mathsf{q})$, where $\mathsf{f}_i(\mathsf{q}) = \mathsf{f}_{\mathsf{P}_i}(\mathsf{q})$. This quantity arises naturally when one tries to model uncertainty [AAH+13]; indeed, let P_i be the set of possible locations of the ith point (i.e., the location of the ith point is chosen randomly, somehow, from the set P_i). Thus, $\mathsf{f}_i(\mathsf{q})$ is the worst case distance to the ith point, and $\mathsf{f}(\mathsf{q})$ is the worst-case nearest-neighbor distance to the random point-set generated by picking the ith point from P_i, for $i = 1, \ldots, n$. We refer to $\mathsf{f}(\cdot)$ as the ***nearest furthest-neighbor*** distance, and we are interested in approximating it.

We prove in Section 6.4.3 that the distance functions $\mathsf{f}_1, \ldots, \mathsf{f}_n$ satisfy the conditions of the framework, and we get the following result.

Theorem 6.2.17 *Given n point sets $\mathsf{P}_1, \ldots, \mathsf{P}_n$ in $\mathbb{R}^d$ with a total of m points, and a parameter $\varepsilon > 0$, one can preprocess these points into a data-structure for answering $(1+\varepsilon)$-ANFN (approximate nearest further-neighbor) queries, to get the following performance:*

(A) The query time is $O\big(\log n + 1/\varepsilon^{d+1}\big)$.

(B) The preprocessing time and space used is $O\left(n\left(\frac{\log n}{\varepsilon}\right)^{O(d)}\right)$.

Note that the space and query time used, depend only on n, and not on the input size.

6.3 Constructing the AVD

The input is a set $\mathcal{F}$ of n functions, satisfying the conditions of Section 6.2.4, and a number $0 < \varepsilon \leq 1$. We preprocess $\mathcal{F}$, such that given a query point q, one can compute a $f \in \mathcal{F}$, where $\mathbb{d}(\mathsf{q}, \mathcal{F}) \leq \mathbb{d}(\mathsf{q}, f) \leq (1+\varepsilon)\mathbb{d}(\mathsf{q}, \mathcal{F})$.

6.3.1 Building blocks

Near-neighbor

Given a set of functions $\mathcal{G}$, a real number $\alpha \geq 0$, and a parameter $\varepsilon > 0$, a ***near-neighbor*** data-structure $\mathcal{D}_{near} = \mathcal{D}_{\mathrm{nr}}(\mathcal{G}, \varepsilon, \alpha)$ can decide (approximately) if a point has distance larger or smaller than α from $\mathcal{G}$.

Formally, for a query point q, a near-neighbor query answers yes if $\mathrm{d}(\mathsf{q}, \mathcal{G}) \leq \alpha$, and no if $\mathrm{d}(\mathsf{q}, \mathcal{G}) > (1+\varepsilon)\alpha$. It can return either answer if $\mathrm{d}(\mathsf{q}, \mathcal{G}) \in \Big(\alpha, (1+\varepsilon)\alpha\Big]$. If it returns yes, then it also returns a function $f \in \mathcal{G}$ such that $\mathrm{d}(\mathsf{q}, f) \leq (1+\varepsilon)\alpha$. The query time of this data-structure is denoted by $T_{\leq}(m)$, where $m = |\mathcal{G}|$.

Lemma 6.3.1 *Given a set of m functions $\mathcal{G} \subseteq \mathcal{F}$, $\alpha > 0$ and $\varepsilon > 0$. One can construct a data-structure (which is a compressed quadtree), of size $O\big(m/\varepsilon^d\big)$, in $O\big(m\varepsilon^{-d}\log(m/\varepsilon)\big)$ time, such that given any query point $\mathsf{q} \in \mathbb{R}^d$, one can answer a $(1+\varepsilon)$-approximate near-neighbor query for the distance α, in time $T_{\leq}(m) = O(\log(m/\varepsilon))$.*

Proof: For each $f \in \mathcal{G}$, consider the canonical grid set $\mathsf{G}_{\approx r_f}(f_{\preceq\alpha})$, where $r_f = \varepsilon\lambda_f(\alpha) \geq \varepsilon\mathsf{diam}(f_{\preceq\alpha})/\zeta$, where $\lambda_f(\cdot)$ and ζ, are the growth function and the growth constant, respectively, see (P2). The sublevel set of interest is $\mathcal{G}_{\preceq\alpha}$, and its approximation is $\mathcal{C} = \bigcup_{f\in\mathcal{G}} \mathsf{G}_{\approx r_f}(f_{\preceq\alpha})$, as the bounded growth condition (P2) implies that $f_{\preceq\alpha} \subseteq \mathsf{G}_{\approx r_f}(f_{\preceq\alpha}) \subseteq f_{\preceq(1+\varepsilon)\alpha}$. The set of canonical cubes $\mathcal{C}$ can be stored in a compressed quadtree $\mathcal{T}$, and given a query point q, we can decide if it is covered by some cube of $\mathcal{C}$, by performing a point location query in $\mathcal{T}$.

By Remark 6.2.11, $\big|\mathsf{G}_{\approx r_f}(f_{\preceq\alpha})\big| = O\big(\varepsilon^{-d}\big)$. As such, the total number of canonical cubes in $\mathcal{C}$ is $O\big(m/\varepsilon^d\big)$, and the compressed quadtree for storing them can be computed in $O\big(m\varepsilon^{-d}\log(m/\varepsilon)\big)$ time [Har11].

We mark a cell of the resulting quadtree by the function whose sublevel set it arose from (ties can be resolved arbitrarily). During query, if q is found in one of the cells we return yes and the function associated with the cell, otherwise we return no.

If we have that $\mathrm{d}(\mathsf{q}, \mathcal{G}) \leq \alpha$, then the query point q will be found in one of the marked cells, since they cover $\mathcal{G}_{\preceq\alpha}$. As such, the query will return yes. Moreover, if the query does return a yes, then it belongs to a cube of $\mathcal{C}$ that is completely covered by $\mathcal{G}_{\preceq(1+\varepsilon)\alpha}$, as desired. ■

Interval data-structure

Given a set of functions $\mathcal{G}$, real numbers $0 < \alpha \leq \beta$, and $\varepsilon > 0$, the ***interval data-structure*** returns for a query point q, one of the following:

(A) If $\mathrm{d}(\mathsf{q}, \mathcal{G}) \in \big[\alpha, \beta\big]$, then it returns a function $g \in \mathcal{G}$ such that $\mathrm{d}(\mathsf{q}, g) \leq (1+\varepsilon)\mathrm{d}(\mathsf{q}, \mathcal{G})$. It might also return such a function for values outside this interval.

(B) "$\mathrm{d}(\mathsf{q}, \mathcal{G}) < \alpha$". In this case it returns a function $g \in \mathcal{G}$, such that $\mathrm{d}(\mathsf{q}, g) < \alpha$.

(C) "$\mathrm{d}(\mathsf{q}, \mathcal{G}) > \beta$".

The time to perform an interval query is denoted by $T_{\mathrm{I}}(m, \alpha, \beta)$.

Lemma 6.3.2 *Given a set of m functions $\mathcal{G}$, an interval $[\alpha, \beta]$, and an approximation parameter $0 < \tau \leq 4$, one can construct an interval data-structure of size $O\Big(m\tau^{-d-1} \log \frac{4\beta}{\alpha}\Big)$, in time $O\big(m\tau^{-d-1} \log \frac{4\beta}{\alpha} \log \frac{m}{\tau}\big)$, such that given a query point q one can answer $(1+\tau)$-ANN query for the distances in the interval $[\alpha, \beta]$, in time $T_{\mathrm{I}}(m, \alpha, \beta, f) = O\left(\log \dfrac{m \log(4\beta/\alpha)}{\tau}\right)$.*

Proof: Using Lemma 6.3.1, build a $(1+\tau/4)$ near-neighbor data-structure $\mathcal{D}_i$ for $\mathcal{G}$, for distance $r_i = (\alpha/2)(1+\tau/4)^i$, for $i = 0, \ldots, L = \Big\lceil \log_{1+\tau/4}(4\beta/\alpha) \Big\rceil = O\big(\tau^{-1} \log(4\beta/\alpha)\big)$. Clearly, an interval query can be answered in three stages:

(A) Perform a point-location query in $\mathcal{D}_0$. If the answer is yes then $\mathrm{d}(\mathsf{q}, \mathcal{G}) < \alpha$. We can also return a function $g \in \mathcal{G}$ with $\mathrm{d}(\mathsf{q}, g) < \alpha$.

(B) Similarly, perform a point-location query in $\mathcal{D}_L$. If the answer is no then $\mathrm{d}(\mathsf{q}, \mathcal{G}) > \beta$ and we are done.

(C) It must be that $\mathrm{d}(\mathsf{q}, \mathcal{G}) \in \big[r_i, r_{i+1}\big]$ for some i. Find this i by performing a binary search on the data-structures $\mathcal{D}_0, \ldots, \mathcal{D}_L$, for the first i such that $\mathcal{D}_i$ returns no, but $\mathcal{D}_{i+1}$ returns yes. Clearly, $\mathcal{D}_{i+1}$ provides us with the desired $(1+\tau/4)^2$-ANN to the query point.

To get the improved query time, observe that we can overlay these compressed quadtrees $\mathcal{D}_0, \ldots, \mathcal{D}_L$, into a single quadtree. For every leaf (or compressed node) of this quadtree, we compute the original node with the lowest value covering this node. Clearly, finding the desired distance can now be resolved by a single point-location query in this overlay of quadtrees. The total size of these quadtrees is $S = O\big(Lm/\tau^d\big)$, and the total time to compute these quadtrees is $T_1 = O\big(L\big(m/\tau^d\big) \log(m/\tau)\big)$, and the time to compute their overlay is $O(S \log L)$. The time to perform a point-location query in the overlayed quadtree is $O(\log S)$. ■

Lemma 6.3.2 readily implies that if somehow a priori we know that the nearest-neighbor distance lies in an interval of values of polynomial spread, then we would get the desired data-structure by just using Lemma 6.3.2. To overcome this unbounded spread problem, we would first argue that, under our assumptions, there are only linear number of intervals where interesting things happen to the distance function.

Connected components of the sublevel sets

Given a finite set X and a partition into disjoint sets $X = X_1 \cup \cdots \cup X_k$, let this partition be denoted by $\langle X_1, \ldots, X_k \rangle_X$. For $i = 1, \ldots, k$, each X_i is a ***part*** of the partition.

Definition 6.3.3 For two partitions $P_A = \langle A_1, \ldots, A_k \rangle_X$ and $P_B = \langle B_1, \ldots, B_l \rangle_X$ of the same set X, P_B is a ***refinement*** of P_A, denoted by $P_B \sqsubseteq P_A$, if for any B_i there exists a set A_{j_i}, such that $B_i \subseteq A_{j_i}$. In the other direction, P_A is a ***coarsening*** of P_B.

Given partitions $\Pi = \langle X_1, \ldots, X_k \rangle_X \sqsubseteq \Xi = \langle X'_1, \ldots, X'_{k'} \rangle_X$, let $\phi(\Pi, \Xi, i)$ be the function that return the set of indices of sets in Π whose union is $X'_i \in \Xi$.

Observation 6.3.4 *(A) Given partitions Π and Ξ of a finite set X, if $\Pi \sqsubseteq \Xi$ then $|\Xi| \leq |\Pi|$.*

(B) Given partitions $\Pi \sqsubseteq \Xi$ of a set X with n elements. The partition function $\phi(\Pi, \Xi, \cdot)$ can be computed in $O(n)$ time. For any $1 \leq i \leq |\Xi|$, the set $\phi(\Pi, \Xi, i)$ can be returned in $O\big(|\phi(\Pi, \Xi, i)|\big)$ time, and its size can be computed in $O(1)$ time.

Definition 6.3.5 Given partitions $\Pi = \langle X_1, \ldots, X_k \rangle_X \sqsubseteq \Xi = \langle X'_1, \ldots, X'_{k'} \rangle_X$ of a set X, for any part X'_i of Ξ, $\Pi[X'_i]$ denotes the ***induced partition*** of X'_i by Π, where $\Pi[X'_i] = \langle X_{i_1}, \ldots, X_{i_r} \rangle_{X'_i}$, and $\{i_1, \ldots, i_r\} = \phi(\Pi, \Xi, i)$.

Definition 6.3.6 For $\mathcal{G} \subseteq \mathcal{F}$ and $\ell > 0$, consider the intersection graph of the sets $f_{\preceq \ell}$, for all $f \in \mathcal{G}$. Each connected component is a ***cluster*** of $\mathcal{G}$ at level ℓ. And the partition of $\mathcal{G}$ by these clusters, denoted by $\mathsf{C}(\mathcal{G}, \ell)$, is the ***$\ell$-clustering*** of $\mathcal{G}$.

The values ℓ at which the ℓ-clustering of $\mathcal{F}$ changes, are, intuitively, the critical values when the sublevel set of $\mathcal{F}$ changes and which influence the AVD. These values are critical in trying to decompose the nearest-neighbor search on $\mathcal{F}$, into a search on smaller sets.

Observation 6.3.7 *If $0 \leq a \leq b$ then $\mathsf{C}(\mathcal{G}, a) \sqsubseteq \mathsf{C}(\mathcal{G}, b)$.*

The following lemma testifies that we can approximate the ℓ-clustering quickly, for any number ℓ.

Lemma 6.3.8 *Given $\mathcal{G} \subseteq \mathcal{F}$, $\ell \geq 0$, and $\varepsilon > 0$, one can compute, in $O\big(\frac{m}{\varepsilon^d} \log(m/\varepsilon)\big)$ time, a partition $\Psi = \Psi_\varepsilon(\mathcal{G}, \ell)$, such that $\mathsf{C}(\mathcal{G}, \ell) \sqsubseteq \Psi \sqsubseteq \mathsf{C}(\mathcal{G}, (1+\varepsilon)\ell)$, where $m = |\mathcal{G}|$.*

Proof: For each $f \in \mathcal{G}$, tile the sublevel sets $(f)_{\preceq \ell}$ by canonical cubes of small enough diameter, such that bounded growth condition (P2) assures all cubes are inside $(f)_{\preceq (1+\varepsilon)\ell}$. To this end, for $f \in \mathcal{G}$, set $r_f = \varepsilon \lambda_f(\ell) \geq \varepsilon \mathsf{diam}(f_{\preceq \ell})/\zeta$, and compute the set $\mathcal{C}_f = \bigcup_{f \in \mathcal{G}} f_{\preceq \ell, \approx r_f}$, see Definition 6.2.1. It is easy to verify that we have that

$$(\mathcal{G})_{\preceq \ell} \subseteq \cup \mathcal{C}_f \subseteq (\mathcal{G})_{\preceq (1+\varepsilon)\ell}. \tag{6.2}$$

By assumption, we have that $|\mathcal{C}_f| = O(1/\varepsilon^d)$, and the total number of canonical cubes in all the sets $\mathcal{C}_f$ for $f \in \mathcal{G}$ is $O(m/\varepsilon^d)$. We throw all these canonical cubes into a compressed quadtree, this takes $O\big((m/\varepsilon^d) \log(m/\varepsilon)\big)$ time. Here, every node of the compressed quadtree is marked if it belongs to some of these sets, and if so, to which of the sets. Two sets $\cup \mathcal{C}_f$ and $\cup \mathcal{C}_g$ intersect, if and only if there are two

canonical cubes, in these two sets, such that they overlap; that is, one of them is a sub-cube of the other. Initialize a union-find data-structure, and traverse the compressed quadtree using **DFS**, keeping track of the current connected component, and performing a union operation whenever encountering a marked node (i.e., all the canonical nodes associated with it, are unionized into the current connected component). Finally, we perform a union operation for all the cells in $\mathcal{C}_f$, for all $f \in \mathcal{G}$. Clearly, this results in the desired connected components of the intersection graph of $\cup \mathcal{C}_f$ (note that we consider two sets as intersecting only if their interiors intersect). Translating each such connected set of canonical cubes back to the functions that gave rise to them, results in the desired partition. ■

The partition Ψ computed by Lemma 6.3.8 is monotone, that is, for $\ell \leq \ell'$ and $\varepsilon \leq \varepsilon'$, we have $\Psi_\varepsilon(\mathcal{G}, \ell) \sqsubseteq \Psi_{\varepsilon'}(\mathcal{G}, \ell')$. Moreover, for each cluster $C \in \Psi_\varepsilon(\mathcal{G}, \ell)$, we have that $\ell_{\text{con}}(C) \leq (1+\varepsilon)\ell$.

Computing a splitting distance

Definition 6.3.9 Given a partition $\Psi = \Psi_1(\mathcal{G}, \ell)$ of $\mathcal{G}$, with $m = |\Psi|$ clusters, a distance x is a ***splitting distance*** if $m/4 \leq |\Psi_1(\mathcal{G}, x/4)|$, $\Psi \sqsubseteq \Psi_1(\mathcal{G}, x/4)$, and $|\Psi_1(\mathcal{G}, x)| \leq (7/8)m$.

Lemma 6.3.10 *Given a partition $\Psi = \Psi_1(\mathcal{G}, \ell)$ of $\mathcal{G}$, one can compute a splitting distance for it, in expected $O(n(\log n + t))$ time, where $n = |\mathcal{G}|$, and t is the maximum cluster size in Ψ.*

Proof: For each cluster $C \in \Psi$, let r_C be its distance from all the functions in $\mathcal{G} \setminus C$; that is $r_C = \min_{f \in C} \min_{g \in \mathcal{G} \setminus C} \mathrm{d}(f, g)$. Note that $r_C \geq \ell$. Now, let $r_1 \leq r_2 \leq \cdots \leq r_m$, be these distances for the m clusters of Ψ. We randomly pick a cluster $C \in \Psi$ and compute r_C for it, by brute force – computing the distance of each function of C with the functions of $\mathcal{G} \setminus C$.

Let $\ell' = \max(r_C, 4\ell)$. Let i be the rank of r_C, among $r_1, \ldots, r_m$. With probability $1/2$, we have that $m/4 \leq i \leq (3/4)m$. If so we have that:

(A) All the clusters that correspond to $r_i, \ldots, r_m$, are singletons in the partition $\Psi_1(\mathcal{G}, \ell'/4)$, as the distance of each one of these clusters, to the nearest cluster other than itself, is larger than $\ell'/4$. We conclude that $|\Psi_1(\mathcal{G}, \ell'/4)| \geq m/4$.

(B) All the clusters of Ψ that correspond to $r_1, \ldots, r_i$, are contained inside a larger cluster of $\Psi_1(\mathcal{G}, \ell')$ (i.e., they were merged with some other cluster). But then, the number of clusters in $\Psi_1(\mathcal{G}, \ell')$ is at most $(7/8)m$. Indeed, put an edge between such a cluster, to the cluster realizing the smallest distance with it. This graph has at least $m' \geq m/4$ edges, and it is easy to see that each component of size at least 2 in the underlying undirected graph, has the same number of edges as vertices. As such the number of singleton components is at most $m - m'$, while the number of components

```
Search( 𝒢, ϒ, q )
    // 𝒢:  set of functions, ϒ = Ψ_1(𝒢, ℓ) for some value ℓ, q:  query point.
    if |ϒ| ≤ 8 then
        return d(q, 𝒢) = min_{f∈𝒢} d(q, f)            (*)
    x ← compute a splitting distance of ϒ, see Lemma 6.3.10

    // Perform an interval ANN query on the interval [x/8, Nx], see Lemma 6.3.2.
    if d(q, 𝒢) ∈ [x/8, Nx] or (1 + ε/4)-ANN found then
        return nearest function found by (1 + ε/4)-ANN interval query.
    if d(q, 𝒢) < x/8 then
        f ← 2-approximate NN query on 𝒢, with distance x/8, see Lemma 6.3.1.
        Find cluster C ∈ Ψ_1(𝒢, x/4), such that f ∈ C, see Lemma 6.3.8.
        return Search( C, ϒ[C], q )

    // Must be that d(q, 𝒢) > Nx
    return Search( 𝒢, Ψ_1(𝒢, x), q )              (**)
```

Figure 6.1: Search algorithm: We are given a query point q, and an approximation parameter $\varepsilon > 0$. The quantity N is a parameter to be specified shortly. Initially, we call this procedure on the set of functions $\mathcal{F}$ with Υ being the partition of $\mathcal{F}$ into singletons (i.e., $\ell = 0$).

of size at least 2, is at most $m'/2$. It follows that the total number of components is at most $m - m'/2 \leq 7m/8$. Since each such component corresponds to a cluster in $\Psi_1(\mathcal{G}, \ell')$, the claim is proved.

Now, compute $\Psi_1(\mathcal{G}, \ell')$ and $\Psi_1(\mathcal{G}, \ell'/4)$, using Lemma 6.3.8. It is clear that, $\Psi \sqsubseteq \Psi_1(\mathcal{G}, \ell'/4)$. With probability at least half, they have the desired sizes, and we are done. Otherwise, we repeat the process. In each iteration we spend $O(n(\log n + t))$ time, and the probability for success is half. As such, in expectation, the number of rounds needed is constant. ■

6.3.2 The search procedure

An initial "naive" implementation

The search procedure is presented in Figure 6.1.

Lemma 6.3.11 Search$(\mathcal{G}, \Upsilon, \mathsf{q})$ *returns a function* $f \in \mathcal{G}$*, such that* $\mathrm{d}(\mathsf{q}, f) \leq (1+\varepsilon)\mathrm{d}(\mathsf{q}, \mathcal{G})$*. The depth of the recursion of* Search *is* $\mathsf{h} = O(\log n)$*, where* $n = |\mathcal{G}|$*.*

Proof: The proof is by induction on the size of Υ. If $|\Upsilon| \leq 8$, then the function realizing $\mathrm{d}(\mathsf{q}, \mathcal{G})$ is returned, and the claim is true.

Let x be the computed splitting distance of Υ. Next, the procedure performs an $(1 + \varepsilon/4)$-approximate interval nearest-neighbor query for q on the range $[x/8, Nx]$. If this computed the approximate nearest-neighbor, then we are done.

Otherwise, it must be that either $\mathrm{d}(\mathsf{q}, \mathcal{G}) < x/8$ or $\mathrm{d}(\mathsf{q}, \mathcal{G}) > Nx$, and significantly, we know which of the two options it is:

(A) If $\mathrm{d}(\mathsf{q}, \mathcal{G}) < x/8$, then doing an approximate near-neighbor query on $\mathcal{G}$, and distance $x/8$, returns a function $f \in \mathcal{G}$ such that $\mathrm{d}(\mathsf{q}, f) < x/4$. Clearly, the nearest-neighbor to q must be in the cluster containing f in the partition $\Psi_1(\mathcal{G}, x/4)$, and **Search** recurses on this cluster. Now, by induction, the returned ANN is correct.

Since x is a splitting distance of Υ, see Definition 6.3.9, we have $|\Upsilon|/4 \leq |\Psi_1(\mathcal{G}, x/4)|$ and $\Upsilon \sqsubseteq \Psi_1(\mathcal{G}, x/4)$. As such, since C is one of the clusters of $\Psi_1(\mathcal{G}, x/4)$, the induced partition of C by Υ (i.e., $\Upsilon[C]$), can have at most $(1 - 1/4)|\Upsilon| + 1 \leq (7/8)|\Upsilon|$ clusters.

(B) Otherwise, we have $\mathrm{d}(\mathsf{q}, \mathcal{G}) > Nx$. Since x is a splitting distance, we have that $|\Psi_1(\mathcal{G}, x)| \leq (7/8)|\Upsilon|$, see Definition 6.3.9. We recurse on $\mathcal{G}$, and a partition that has fewer clusters, and by induction, the returned answer is correct.

In each step of the recursion, the number of sets in the partition shrunk by at least a fraction of $7/8$. As such, after a logarithmic number of recursive calls, the procedure is done. ■

But where is the beef? Modifying Search to provide fast query time

The reader might wonder how we are going to get an efficient search algorithm out of **Search**, as in the case that Υ has a small number of clusters (i.e., 8), still requires us to perform a scan on all the functions in these clusters, and compute their distance from the query point q. It turns out, that we can assume that each cluster of Υ has size bounded by $O(1/\varepsilon^{\mathsf{c}_{\mathrm{sk}}})$, i.e., it is really a sketch (of some set of functions). Initially, since all the clusters are singletons, this is clearly true. In future iterations, when clusters merge – and this happens during (**), as we shall see, it is okay to pass the sketches as proxies for the actual cluster, bounding the size of each cluster. In particular, the query time is $O(1/\varepsilon^{\mathsf{c}_{\mathrm{sk}}} + \log^2 n)$. Indeed, an interval query takes $O(\log n)$ time, and there $O(\log n)$ such queries. The final query on the sketch takes time proportional to the sketch size which is $O(1/\varepsilon^{\mathsf{c}_{\mathrm{sk}}})$.

As such, the major challenge is not making the query process fast, but rather building the search structure quickly, and arguing that it requires little space.

Sketching a sketch

To improve the efficiency of the preprocessing for **Search**, we are going to use sketches more aggressively. Specifically, for each of the clusters of Υ, we can compute their δ-sketches, for $\delta = \varepsilon/(8\mathsf{h}) = O(\varepsilon/\log n)$, see Lemma 6.3.11. From this point on, when we manipulate this cluster, we do it on its sketch. To make

this work set $N = n^{4\mathsf{c}_{\mathrm{sk}}}$. Indeed, for any set of functions $\mathcal{G}$ under consideration, their δ-sketch is active at a distance $O(\ell_{\mathrm{con}}(\mathcal{G})(|\mathcal{G}|/\delta)^{\mathsf{c}_{\mathrm{sk}}}) = O(\ell_{\mathrm{con}}(\mathcal{G})(n/\delta)^{\mathsf{c}_{\mathrm{sk}}})$, see (P3). Now assuming $\delta = O(\varepsilon/\log n)$, and ε at least $1/\log n$ (otherwise our algorithm is no worse than brute-force search, essentially), as well as n sufficiently large, it follows that a sketch is active at $N = n^{4\mathsf{c}_{\mathrm{sk}}}$ times $\ell_{\mathrm{con}}(\mathcal{G})$.

The only place in the algorithm where we need to compute the sketches, is in (**) in Figure 6.1. Specifically, we compute $\Psi_1(\mathcal{G}, x)$, and for each new cluster $C \in \Psi_1(\mathcal{G}, x)$, we combine all the sketches of the clusters $D \in \Upsilon$ such that $D \subseteq C$, into a single set of functions. We then compute a δ-sketch for this set, and this sketch is this cluster from this point on. In particular, the recursive calls to **Search** would send the sketches of the clusters, and not the clusters themselves. Conceptually, the recursive call would also pass the minimum distance where the sketches are active – it is easy to verify that we use these sketches only at distances that are large enough to be allowable (i.e., the sketches represent the functions they correspond to, well in these distances).

Importantly, whenever we compute such a new set, we do so for a distance that is bigger by a polynomial factor (i.e., N) than the connectivity level of the clusters being merged. Indeed, observe that $\mathrm{d}(\mathsf{q}, \mathcal{G}) > Nx$ and Nx is N times bigger than x, which is an upper bound on the connectivity level of the clusters being merged.

As such, all these sketches are valid, and can be used at this distance (or any larger distance). Of course, the quality of the sketch deteriorates. In particular, since the depth of recursion is h, the worst quality of any of the sketches created in this process is at most $(1+\delta)^{\mathsf{h}} \leq 1 + \varepsilon/4$.

Significantly, before using such a sketch, we would shrink it by computing a $\varepsilon/8$-sketch of it. This would reduce the sketch size to $O(1/\varepsilon^{\mathsf{c}_{\mathrm{sk}}})$. Note however, that this still does not help us as far as recursion - we must pass the larger δ-sketches in the recursive call of (**).

This completes the description of the search procedure. It is still unclear how to precompute all the data-structures required during the search. To do that, we need to better understand what the search process does.

6.3.3 The connectivity tree, and the preprocessing

Given a set of functions $\mathcal{F}$, consider the tree tracking the connected components of the sublevel sets of the functions, as the level changes continuously. Formally, initially we start with n singletons (which are the leaves of the tree) that are labeled with the value zero, and we store them in a set $\mathcal{F}$ of active nodes. Now, we compute for each pair of sets of functions $X, Y \in \mathcal{F}$ the distance $\mathrm{d}(X, Y)$, and let X', Y' be the pair realizing the minimum of this quantity. Merge the two sets into a new set $Z = X' \cup Y'$, create a new node

for this set having the node for X' and Y' as children, and set its label to be $\mathbb{d}(X', Y')$. Finally, remove X' and Y' from $\mathcal{F}$ and insert Z into it. Repeat till there is a single element in $\mathcal{F}$. Clearly, the result is a tree that tracks the connected components, during the execution of Kruskal's algorithm for MST, for the graph on the functions where distances are defined using $\mathbb{d}(\cdot, \cdot)$.

To make the presentation consistent, let $\mathbb{d}_{\approx}(X, Y)$ be the minimum x, such that $\Psi_1(X \cup Y, x)$ is connected (see Lemma 6.3.8), and x is a power of two. Computing $\mathbb{d}_{\approx}(X, Y)$ can be done by computing $\mathbb{d}_{\approx}(f, g)$ for each pair of functions separately. This in turn, can be done by first computing $\alpha = \mathbb{d}(f, g)$, and observing that x is between $\alpha/2$ and α. In particular, since x must be a power of two, so there are at most 2 candidate values to consider, and which is the right one can be decided using Lemma 6.3.8.

So, in the above, we use $\mathbb{d}_{\approx}(\cdot, \cdot)$ instead of $\mathbb{d}(\cdot, \cdot)$, and let $\mathcal{H}$ be the resulting tree. For a value ℓ, let $\mathrm{L}_{\mathcal{H}}(\ell)$ be the set of nodes such that their label is smaller than ℓ, but their parent label is larger than ℓ. It is easy to verify that $\mathrm{L}_{\mathcal{H}}(\ell)$ corresponds to $\Psi = \Psi_1(\mathcal{F}, \ell)$; indeed, every cluster $C \in \Psi$ corresponds to a node $u \in \mathrm{L}_{\mathcal{H}}(\ell)$, such that the set of functions stored in the leaves of the subtree of u, denoted by $\mathrm{F}(u)$ is C. The following can be easily proved by induction. Recall that we are using the approximate distances $\mathbb{d}_{\approx}(X, Y)$ between sets, and all splitting distances, as well as $N = n^{4\mathsf{c}_{\mathrm{sk}}}$ are set to be the closest power of 2.

Lemma 6.3.12 *Consider a recursive call* **Search**$(\mathcal{G}, \Upsilon, \mathsf{q})$*, see Figure 6.1, made during the search algorithm execution. Then there exists a node u, and a value ℓ, such that (i) $\mathcal{G} = \mathrm{F}(u)$ and, (ii) $\Upsilon = \left\{ \mathrm{F}(v) \;\middle|\; v \in \mathrm{L}_{\mathcal{H}}(\ell)\text{, and } v \text{ is in the subtree of } u \right\}$. That is, a recursive call of* **Search** *corresponds to a subtree of $\mathcal{H}$.*

Of course, not all possible subtrees are candidates to be such a recursive call. In particular, **Search** can now be interpreted as working on a subtree T of $\mathcal{H}$, as follows:

(A) If T is a single node u, then find the closest function in $\mathrm{F}(u)$ to q. Using the sketch this can be done quickly.

(B) Otherwise, compute a distance x, such that the number of nodes in the level $\mathrm{L}_T(x)$ is roughly half the number of leaves of T.

(C) Using interval data-structure determine if the distance $\mathbb{d}(\mathsf{q}, \mathrm{F}(T))$ is in the range $[x/8, Nx]$. If so, we found the desired ANN.

(D) If $\mathbb{d}(\mathsf{q}, \mathrm{F}(T)) > Nx$ then continue recursively on portion of T above $\mathrm{L}_T(x)$.

(E) If $\mathbb{d}(\mathsf{q}, \mathrm{F}(T)) < x/8$ then we know the node $u \in \mathrm{L}_T(x/4)$ such that the ANN belongs to $\mathrm{F}(u)$. Continue the search recursively on the subtree of T rooted at u.

That is, **Search** breaks T into subtrees, and continues the search recursively on one of the subtrees.

Significantly, the size of every such subtree is at most a constant fraction of the size of T, and every edge of T belongs to a single such subtree.

The preprocessing now works by precomputing all the data-structures required by **Search**. Of course, the most natural approach would be to precompute $\mathcal{H}$, and build the search tree by simulating the above recursion on $\mathcal{H}$. Fortunately, this is not necessary, and we only use the above $\mathcal{H}$ in analyzing the preprocessing running time.

In particular, given a subtree T with m edges, the corresponding partition Υ would have at most m sets. Each such set would have a δ-sketch, and we compute a $\varepsilon/8$-sketch for each one of these sketches. Namely, the input size here is $M = O(m/\delta^{\mathsf{c}_{\mathrm{sk}}})$. Computing the $\varepsilon/8$-sketches for each one of these sketches takes $U_1 = O(M/\varepsilon^{\mathsf{c}_{\mathrm{sk}}}) = O\Big(m(\varepsilon\delta)^{-\mathsf{c}_{\mathrm{sk}}}\Big)$ time, see Remark 6.2.12. Computing the splitting distance, using Lemma 6.3.10, takes $U_2 = O(M(\log M + 1/\delta^{\mathsf{c}_{\mathrm{sk}}})) = O\big(m \log n\delta^{-\mathsf{c}_{\mathrm{sk}}} + m\delta^{-2\mathsf{c}_{\mathrm{sk}}}\big)$ time. Computing the interval data-structure Lemma 6.3.2 takes $U_3 = O\big(M\varepsilon^{-d-1}\log n \log M\big) = O\big(m\log^2 n\delta^{-\mathsf{c}_{\mathrm{sk}}}\varepsilon^{-d-1}\big)$ time, and requires $S_1 = O\big(M\varepsilon^{-d-1}\log n\big) = O\big(m\log n\delta^{-\mathsf{c}_{\mathrm{sk}}}\varepsilon^{-d-1}\big)$ space. This breaks T into edge disjoint subtrees $T_1, \ldots, T_t$, and we compute the search data-structure for each one of them separately (each one of these subtrees is smaller by a constant fraction of the original tree). Finally, we need to compute the δ-sketches for the clusters sent to the appropriate recursive calls, and this takes $U_4 = O(M/\delta^{\mathsf{c}_{\mathrm{sk}}}) = O\big(m\delta^{-2\mathsf{c}_{\mathrm{sk}}}\big)$ time, by Remark 6.2.12.

Every edge of the tree T gets charged for the amount of work spent in building the top level data-structure. That is, the top level amortized work each edge of T has to pay is

$$\begin{aligned} W &= O\Big(\left(U_1 + U_2 + U_3 + U_4\right)/m\Big) = O\Big((\varepsilon\delta)^{-\mathsf{c}_{\mathrm{sk}}} + \delta^{-\mathsf{c}_{\mathrm{sk}}}\log n + \delta^{-2\mathsf{c}_{\mathrm{sk}}} + \varepsilon^{-d-1}\delta^{-\mathsf{c}_{\mathrm{sk}}}\log^2 n + \delta^{-2\mathsf{c}_{\mathrm{sk}}}\Big) \\ &= O\Big(\varepsilon^{-\max(d+1+\mathsf{c}_{\mathrm{sk}},2\mathsf{c}_{\mathrm{sk}})}\log^{\max(\mathsf{c}_{\mathrm{sk}}+2,2\mathsf{c}_{\mathrm{sk}})} n\Big). \end{aligned}$$

Now, it is valid to assume a larger value for the sketch constant c_{sk}, than its optimal value as per its definition by (P3). As such we assume that $\mathsf{c}_{\mathrm{sk}} \geq (d+1) \geq 2$, and we can simplify the above expression to $O(\varepsilon^{-2\mathsf{c}_{\mathrm{sk}}}\log^{2\mathsf{c}_{\mathrm{sk}}} n)$. Since an edge of T gets charged at most $O(\log n)$ times by this recursive construction, we conclude that the total preprocessing time is $O\big(n\varepsilon^{-2\mathsf{c}_{\mathrm{sk}}}\log^{2\mathsf{c}_{\mathrm{sk}}+1} n\big)$.

By the same argument, each edge requires $O(\log n \cdot (S_1/m)) = O\big(\varepsilon^{-d-1-\mathsf{c}_{\mathrm{sk}}}\log^{\mathsf{c}_{\mathrm{sk}}+1} n\big)$ space. Moreover, the space used to store the $\varepsilon/8$-sketches at the leaf nodes is just $O(n\varepsilon^{-\mathsf{c}_{\mathrm{sk}}})$. As such, the overall space used by the data-structure is $\big(n\varepsilon^{-d-1-\mathsf{c}_{\mathrm{sk}}}\log^{\mathsf{c}_{\mathrm{sk}}+1} n\big)$. As for the query time, it boils down to $O(\log n)$ interval queries, and then scanning one $O(\varepsilon)$-sketch. As such, this takes $O\big(\log^2 n + 1/\varepsilon^{\mathsf{c}_{\mathrm{sk}}}\big)$ time.

6.3.4 The result

Proof of 1 Theorem 6.2.13: *The query time stated above is $O\big(\log^2 n + 1/\varepsilon^{\mathsf{c}_{\mathrm{sk}}}\big)$. To get the improved query time, we observe that* **Search** *performs a sequence of point-location queries in a sequence of interval near-neighbor data-structures (i.e., compressed quadtrees), and then it scans a set of functions of size $O(1/\varepsilon^{\mathsf{c}_{\mathrm{sk}}})$ to find the ANN. We take all these quadtrees spread through our data-structure, and assign them priority, where a quadtree $\mathcal{T}_1$ has higher priority than a compressed quadtree $\mathcal{T}_2$, if $\mathcal{T}_1$ is queried after $\mathcal{T}_2$, for any search query. Such an ordering can be assigned, as, conceptually, the search proceeds using queries on the nodes of a tree, down from the root. This defines an acyclic ordering on these compressed quadtrees. Overlaying all these compressed quadtrees together, one needs to return for the query point, the leaf of the highest priority quadtree that contains the query point. This can be easily done by scanning the compressed quadtree, and for every leaf, computing the highest priority leaf that contains it (observe, that here we are overlaying only the nodes in the compressed quadtrees that are marked by some sublevel set – nodes that are empty are ignored). Furthermore, since not all query points can be dealt with by one of the interval near-neighbor data-structures, we must store sketches of size $O(1/\varepsilon^{\mathsf{c}_{\mathrm{sk}}})$, for certain regions of $[0,1]^d$ – such regions are not covered by any of the cubes arising from the interval near-neighbor data-structures.*

A tedious, but straightforward induction argument, implies that doing a point-location query in the resulting quadtree, is equivalent to running the search procedure, as described above. Once we found the leaf that contains the query point, we scan the sketch associated with this cell, and return the computed nearest-neighbor.

Proof of 2 Corollary 6.2.14: *We build the data-structure of Theorem 6.2.13, except that instead of linearly scanning the sketch at a leaf node, during the query time, we preprocess each such sketch (which is set of functions of size $O(1/\varepsilon^{\mathsf{c}_{\mathrm{sk}}})$), for an exact point-location query; that is, we compute the lower envelope of the sketch and preprocess it for vertical ray shooting. This can be done because we are assuming that our function family satisfies the well-behaveness condition, see Definition 6.2.6. This would require $O\big(1/\varepsilon^{\gamma d \mathsf{c}_{\mathrm{sk}}}\big)$ space and time, and the linear scanning that takes $O(1/\varepsilon^{\mathsf{c}_{\mathrm{sk}}})$ time, now is replaced by a point-location query that takes $O(\log 1/\varepsilon)$, as desired. The total query time is thus $O(\log n + \log 1/\varepsilon) = O(\log n/\varepsilon)$.*

Using a similar argument, the space requirement is $O\big(n\varepsilon^{-d-1-\mathsf{c}_{\mathrm{sk}}} \log^{\mathsf{c}_{\mathrm{sk}}+1} n + n\varepsilon^{-\gamma d \mathsf{c}_{\mathrm{sk}}}\big)$, where the first part comes from Theorem 6.2.13, and for the second part we notice that we replaced the space used for sketches, $O(n\varepsilon^{-\mathsf{c}_{\mathrm{sk}}})$ by $O(n\varepsilon^{-\gamma d \mathsf{c}_{\mathrm{sk}}})$, the space required to store the point-location data structures for the projection of the lower envelope of the sketch onto $\mathbb{R}^d$, each of which is a partition of space, intersected with the region in which the sketch is to be applied – such a region is one arising from the compressed quadtree of Theorem 6.2.13, and is either a cube or the set difference of two cubes.

6.4 Applications

We present some concrete classes of functions that satisfy our framework, and for which we construct AVD's efficiently.

6.4.1 Multiplicative distance functions with additive offsets

As a warm-up we present the simpler case of additively offset multiplicative distance functions. The results of this section are almost subsumed by more general results in Section 6.4.2. Here the sublevel sets look like expanding balls, but there is a time lag before the balls even come into existence, i.e., sublevel sets are empty up-to a certain level – this corresponds to the additive offsets. In Section 6.4.2 the sublevel sets are more general fat bodies but there is no additive offset. The results in the present section essentially give an AVD construction of approximate weighted Voronoi diagrams. More formally, we are given a set of points $\mathsf{P} = \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}$. For $i = 1, \ldots, n$, the point p_i has weight $w_i > 0$, and a constant $\alpha_i \geq 0$, associated with it. We define $f_i(\mathsf{q}) = w_i \|\mathsf{q} - \mathsf{p}_i\| + \alpha_i$. Let $\mathcal{F} = \{f_1, \ldots, f_n\}$. We have, $(f_i)_{\preceq y} = \emptyset$ for $y < \alpha_i$, and $(f_i)_{\preceq y} = \mathsf{ball}\Big(\mathsf{p}_i, \frac{y-\alpha_i}{w_i}\Big)$ for $y \geq \alpha_i$. Checking conditions (C1) and (C2) is trivial. As for (C3) we have the following easy lemma,

Lemma 6.4.1 *For any $1 \leq i, j \leq n$ we have $\mathrm{d}(f_i, f_j) = \max\Big(\alpha_i,\, \alpha_j,\, \|\mathsf{p}_i - \mathsf{p}_j\| \frac{w_i w_j}{w_i + w_j} + \frac{\alpha_i w_j + \alpha_j w_i}{w_i + w_j}\Big)$.*

Proof: The ith distance function is $f_i(\mathsf{q}) = w_i \|\mathsf{q} - \mathsf{p}_i\| + \alpha_i$. As such, for $y < \max(\alpha_i, \alpha_j)$, either $(f_i)_{\preceq y} = \emptyset$, or $(f_j)_{\preceq y} = \emptyset$, and $(f_i)_{\preceq y} \cap (f_j)_{\preceq y} = \emptyset$. For $y \geq \max(\alpha_i, \alpha_j)$, we have $f_i(\mathsf{q}) \leq y \iff w_i \|\mathsf{q} - \mathsf{p}_i\| + \alpha_i \leq y \iff \|\mathsf{q} - \mathsf{p}_i\| \leq \frac{y-\alpha_i}{w_i}$, which implies that $\mathsf{q} \in \mathsf{ball}\Big(\mathsf{p}_i, \frac{y-\alpha_i}{w_i}\Big)$; that is, we have $(f_i)_{\preceq y} = \mathsf{ball}\Big(\mathsf{p}_i, \frac{y-\alpha_i}{w_i}\Big)$, and $(f_j)_{\preceq y} = \mathsf{ball}\Big(\mathsf{p}_j, \frac{y-\alpha_j}{w_j}\Big)$.

Now, if $\mathsf{p}_i = \mathsf{p}_j$, then the distance between the two functions is the minimal value such that their sublevel sets are not empty, and this is $\max(\alpha_i, \alpha_j)$. In particular, $\frac{\alpha_i w_j + \alpha_j w_i}{w_i + w_j} \leq \max(\alpha_i, \alpha_j)$, and we also have $\max\Big(\alpha_i, \alpha_j, \|\mathsf{p}_i - \mathsf{p}_j\| \frac{w_i w_j}{w_i + w_j} + \frac{\alpha_i w_j + \alpha_j w_i}{w_i + w_j}\Big) = \max(\alpha_i, \alpha_j,)$, as desired.

If $\mathsf{p}_i \neq \mathsf{p}_j$ the sublevel sets intersect for the first time when the balls $\mathsf{ball}\Big(\mathsf{p}_i, \frac{y-\alpha_i}{w_i}\Big)$ and $\mathsf{ball}\Big(\mathsf{p}_j, \frac{y-\alpha_j}{w_j}\Big)$ touch at a point that belongs to the segment $\mathsf{p}_i\mathsf{p}_j$. Clearly then we have, $\|\mathsf{p}_i - \mathsf{p}_j\| = \frac{y-\alpha_i}{w_i} + \frac{y-\alpha_j}{w_j} \implies w_i w_j \|\mathsf{p}_i - \mathsf{p}_j\| = w_j(y - \alpha_i) + w_i(y - \alpha_j) \implies (w_i + w_j)\, y = w_i w_j \|\mathsf{p}_i - \mathsf{p}_j\| + w_j \alpha_i + w_i \alpha_j \implies y = \|\mathsf{p}_i - \mathsf{p}_j\| \frac{w_i w_j}{w_i + w_j} + \frac{\alpha_i w_j + \alpha_j w_i}{w_i + w_j}$. ∎

Lemma 6.4.2 *Given $1 \leq i, j \leq n$, such that $w_i \leq w_j$. Suppose $y \geq \max(\alpha_i, \alpha_j)$. Then, for any $\delta \geq 0$, we have $(f_j)_{\preceq y} \subseteq (f_i)_{\preceq (1+\delta) y}$ if and only if $y \geq \frac{\|\mathsf{p}_i - \mathsf{p}_j\| + \alpha_i/w_i - \alpha_j/w_j}{(1+\delta)/w_i - 1/w_j}$.*

Proof: For $y \geq \max(\alpha_i, \alpha_j)$, we have that $(f_i)_{\preceq y} = \mathsf{ball}\Big(\mathsf{p}_i, \frac{y-\alpha_i}{w_i}\Big)$, and $(f_j)_{\preceq y} = \mathsf{ball}\Big(\mathsf{p}_j, \frac{y-\alpha_j}{w_j}\Big)$. If $\mathsf{p}_i = \mathsf{p}_j$, then for any y such that $\frac{(1+\delta)y-\alpha_i}{w_i} \geq \frac{y-\alpha_j}{w_j}$, we will have that $(f_j)_{\preceq y} \subseteq (f_i)_{\preceq (1+\delta)y}$. Clearly, this condition is also necessary. It is easy to verify that this is equivalent to the desired expression.

Consider the case $\mathsf{p}_i \neq \mathsf{p}_j$. For any $\mathsf{u} \in \mathsf{ball}\Big(\mathsf{p}_j, \frac{y-\alpha_j}{w_j}\Big)$ we have $\|\mathsf{u} - \mathsf{p}_i\| \leq \|\mathsf{p}_i - \mathsf{p}_j\| + \|\mathsf{p}_j - \mathsf{u}\| \leq \|\mathsf{p}_i - \mathsf{p}_j\| + \frac{y-\alpha_j}{w_j}$, by the triangle inequality. Therefore, if $\frac{(1+\delta)y-\alpha_i}{w_i} \geq \|\mathsf{p}_i - \mathsf{p}_j\| + \frac{y-\alpha_j}{w_j}$, then $\mathsf{ball}\Big(\mathsf{p}_j, \frac{y-\alpha_j}{w_j}\Big) \subseteq \mathsf{ball}\Big(\mathsf{p}_i, \frac{(1+\delta)y-\alpha_i}{w_i}\Big)$. This is exactly the stated condition. Indeed, by rearrangement, $y\big((1+\delta)/w_i - 1/w_j\big) \geq \|\mathsf{p}_i - \mathsf{p}_j\| + \alpha_i/w_i - \alpha_j/w_j$.

As for the other direction, note that $\mathsf{ball}\Big(\mathsf{p}_j, \frac{y-\alpha_j}{w_j}\Big)$ has a boundary point at distance $\frac{y-\alpha_j}{w_j}$ from p_j on the directed line from p_i to p_j on the other side of p_j as p_i, while $\mathsf{ball}\Big(\mathsf{p}_i, \frac{(1+\delta)y-\alpha_i}{w_i}\Big)$ has the intercept of $\frac{(1+\delta)y-\alpha_i}{w_i} - \|\mathsf{p}_i - \mathsf{p}_j\|$. For the condition to hold it must be true that $\frac{(1+\delta)y-\alpha_i}{w_i} - \|\mathsf{p}_i - \mathsf{p}_j\| \geq \frac{y-\alpha_j}{w_j}$, which is also the stated condition. ■

It is easy to see that compactness (P1) and bounded growth (P2) hold for the set of functions $\mathcal{F}$ (for (P2) we can take the growth function $\lambda_{(f_i)}(y) = (y - \alpha_i)/w_i$ for $y \geq \alpha_i$ and the growth constant ζ to be 2). The following lemma proves the sketch property (P3).

Lemma 6.4.3 *For any $\mathcal{G} \subseteq \mathcal{F}$ and $\delta > 0$ there is a (δ, y_0)-sketch $\mathcal{H} \subseteq \mathcal{G}$, with $|\mathcal{H}| = 1$ and $y_0 = 3\ell_{\mathrm{con}}(\mathcal{G})\,|\mathcal{G}|\,/\delta$.*

Proof: If $|\mathcal{G}| = 1$, set $\mathcal{H} = \mathcal{G}$. Otherwise, let $\ell = \ell_{\mathrm{con}}(\mathcal{G})$ for brevity. Observe that $\ell \geq \max\limits_{i:\, f_i \in \mathcal{G}} \alpha_i$, as otherwise some $(f_i)_{\preceq \ell} = \emptyset$, and it cannot be part of a connected collection of sets. Let $|\mathcal{G}| = m \geq 2$, and let $\mathcal{G} = \{f_1, \ldots, f_m\}$, and assume that $w_1 \leq w_i$, for $i = 1, \ldots, m$. Set $\mathcal{H} = \{f_1\}$ – the function with the minimum associated weight. Since $\ell \geq \alpha_i$, we have $(f_i)_{\preceq \ell}$ is the ball $\mathsf{ball}\Big(\mathsf{p}_i, \frac{\ell-\alpha_i}{w_i}\Big)$, for $i = 1, \ldots, m$. Since $\mathcal{G}_{\preceq \ell}$ is connected, it must be true that, for a fixed $j, 2 \leq j \leq m$, there exist a sequence of distinct indices, $1 = i_1, i_2, \ldots, i_{k-1}, i_k = j$, such that $\mathsf{ball}\Big(\mathsf{p}_{i_r}, \frac{\ell-\alpha_{i_r}}{w_{i_r}}\Big) \cap \mathsf{ball}\Big(\mathsf{p}_{i_{r+1}}, \frac{\ell-\alpha_{i_{r+1}}}{w_{i_{r+1}}}\Big) \neq \emptyset$, for $r = 1, \ldots, k-1$. By Lemma 6.4.1, we have $\ell \geq \frac{\|\mathsf{p}_{i_r} - \mathsf{p}_{i_{r+1}}\| + \alpha_{i_r}/w_{i_r} + \alpha_{i_{r+1}}/w_{i_{r+1}}}{1/w_{i_r} + 1/w_{i_{r+1}}}$. Rearranging, $\big\|\mathsf{p}_{i_r} - \mathsf{p}_{i_{r+1}}\big\| \leq \ell \cdot \left(\frac{1}{w_{i_r}} + \frac{1}{w_{i_{r+1}}}\right) - \left(\frac{\alpha_{i_r}}{w_{i_r}} + \frac{\alpha_{i_{r+1}}}{w_{i_{r+1}}}\right) \leq \frac{2\ell}{w_1}$, as $w_1 \leq w_i$, for $i = 1, \ldots, m$. It follows by the triangle inequality and the above, that $\|\mathsf{p}_{i_1} - \mathsf{p}_{i_k}\| \leq \sum_{r=1}^{k-1} \big\|\mathsf{p}_{i_r} - \mathsf{p}_{i_{r+1}}\big\| \leq 2(k-1)\ell/w_1 \leq 2m\ell/w_1$. Thus we have, $\|\mathsf{p}_1 - \mathsf{p}_j\| \leq 2m\ell/w_1$, for $j = 1, \ldots, m$. Let $y_0 = 3\ell\,|\mathcal{G}|/\delta = 3m\ell/\delta$. Then, for $y \geq y_0$, we have that, $y \geq \frac{3m\ell}{\delta} = \frac{2m\ell/w_1 + m\ell/w_1}{\delta/w_1} \geq \frac{2m\ell/w_1 + \ell/w_1}{\delta/w_1}$, for $m \geq 2$. The above implies that, for $y \geq y_0$, we have $y \geq \frac{\|\mathsf{p}_1 - \mathsf{p}_j\| + \ell/w_1}{\delta/w_1} \geq \frac{\|\mathsf{p}_1 - \mathsf{p}_j\| + \alpha_1/w_1}{\delta/w_1}$, since $\ell \geq \alpha_1$. It follows that for $y \geq y_0$, $y \geq \frac{\|\mathsf{p}_1 - \mathsf{p}_j\| + \alpha_1/w_1}{\delta/w_1} \geq \frac{\|\mathsf{p}_1 - \mathsf{p}_j\| + \alpha_1/w_1 - \alpha_j/w_j}{\delta/w_1 + (1/w_1 - 1/w_j)} = \frac{\|\mathsf{p}_1 - \mathsf{p}_j\| + \alpha_1/w_1 - \alpha_j/w_j}{(1+\delta)/w_1 - 1/w_j}$, as $w_1 \leq w_j$, for $j = 1, \ldots, m$. Thus, by Lemma 6.4.2, $\mathsf{ball}\Big(\mathsf{p}_j, \frac{y-\alpha_j}{w_j}\Big) \subseteq \mathsf{ball}\Big(\mathsf{p}_1, \frac{(1+\delta)y-\alpha_1}{w_1}\Big)$, for $y \geq y_0$, and therefore, by definition, $\mathcal{H}$ is a (δ, y_0)-sketch for $\mathcal{G}$. ■

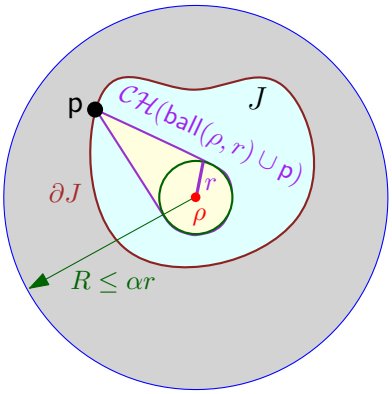


Figure 6.2: Being α-rounded fat.

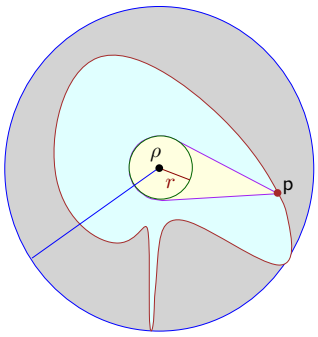


Figure 6.3: A α-fat star shaped region that is not α-rounded fat. Clearly, the gap between the rounded fatness parameter and the regular fatness parameter can be made to be arbitrarily large.

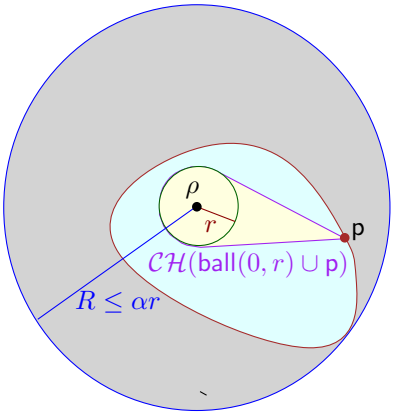


Figure 6.4: A α-fat convex body is α-rounded fat (for the same center point ρ).

From the above, and the requirement that the sketch constant c_{sk} be at least $d+1$ for our bounds to hold, it follows that we can choose the sketch constant $\mathsf{c}_{\mathrm{sk}} = \max(1, d+1) = d+1$. Moreover, as the graphs of the functions are cones, we can easily see that this function family satisfies the well-behaveness condition, see Definition 6.2.6, and that we can choose the wellness constant $\gamma = 1$. Thus, we get Theorem 6.2.15.

6.4.2 Scaling distance – generalized polytope distances

Let $J \subseteq \mathbb{R}^d$ be a compact set containing a "center" point ρ in its interior. Then J is ***star shaped***, if for any point $\mathsf{v} \in J$, the entire segment $\rho\mathsf{v}$ is also in J. Naturally, any convex body J with any center ρ in the interior of J, is star shaped. The ***t-scaling*** of J with a center ρ, is the set, $tJ = \left\{ t(\mathsf{v} - \rho) + \rho \;\middle|\; \mathsf{v} \in J \right\}$.

Given a star shaped object J with a center ρ, the ***scaling distance*** of a point q from J is the minimum t, such that $\mathsf{p} \in tJ$, and let $f_J(\mathsf{q})$ denote this distance function. Note that, for any $y \geq 0$, the sublevel set $(f_J)_{\leq y}$, is the y-scaling of J, that is $(f_J)_{\leq y} = yJ$. Note that for a point $\mathsf{p} \in \mathbb{R}^d$, if we take $J = \mathsf{ball}(\mathsf{p}, 1)$ with center p, then $f_J(\mathsf{q}) = \|\mathsf{q} - \mathsf{p}\|$. That is, this distance notion is a strict extension of the Euclidean distance. Here, we assume that an object J contains the origin in its interior, and the origin is the designated center, unless otherwise stated.

Definition 6.4.4 A star shaped object $J \subseteq \mathbb{R}^d$ centered at ρ is ***α-fat*** if there is a number r such that, $\mathsf{ball}(\rho, r) \subseteq J \subseteq \mathsf{ball}(\rho, \alpha r)$.

Definition 6.4.5 Let J be a star shaped object centered at ρ. The object J is ***α-rounded fat***, if there is a radius r such that, (i) $\mathsf{ball}(\rho, r) \subseteq J \subseteq \mathsf{ball}(\rho, \alpha r)$, and (ii) For every point p in the boundary of J, the cone $\mathcal{CH}(\mathsf{ball}(\rho, r) \cup \mathsf{p})$, lies within J, see Figure 6.2.

By definition, any α-rounded fat object is also α-fat. However, it is not true that a α-fat object is necessarily rounded fat, see Figure 6.3. The following useful result is easy to see, also see Figure 6.4 for an illustration.

Lemma 6.4.6 *An object J that is a α-fat and convex is also α-rounded fat.*

Given a set $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ of n star shaped objects, consider the set $\mathcal{F}$ of n scaling distance functions, where the ith function, for $i = 1, \ldots, n$, is $f_i = f_{J_i}$. We assume that the boundary of each object J_i, has constant complexity.

We next argue that $\mathcal{F}$ complies with the framework of Section 6.2.4. Using standard techniques, we can compute the quantities required in conditions (C1)–(C3), as well as the diameter of the sublevel set, as $\mathsf{diam}(yJ_i) = y\mathsf{diam}(J_i)$. Also, trivially we have that condition (P1) is satisfied, as the sublevel sets are dilations of the J_i, and are thus compact by definition. The next few lemmas establish that both bounded growth (P2), and the sketch property (P3), are also true, if the objects are also α-rounded fat, for some constant α.

Lemma 6.4.7 *Given $\alpha > 0$, and an object J that is α-rounded fat. Then, for any $c \geq 2\alpha$, $y \geq 0$, and $\varepsilon > 0$, we have that $yJ \oplus \mathsf{ball}(0, (\varepsilon/c)\mathsf{diam}(yJ)) \subseteq (1+\varepsilon)yJ$; that is, $(f_J)_{\preceq y} \oplus \mathsf{ball}\Big(0, (\varepsilon/c)\mathsf{diam}\Big((f_J)_{\preceq y}\Big)\Big) \subseteq (f_J)_{\preceq (1+\varepsilon)y}$.*

Proof: Since $(f_J)_{\preceq y} = yJ$ it is enough to show that $yJ \oplus \mathsf{ball}\big(0, (\varepsilon/c)\mathsf{diam}(yJ)\big) \subseteq (1+\varepsilon)yJ$. Let r be the radius guaranteed by Definition 6.4.5 for J. Clearly $\mathsf{diam}(yJ) = y\mathsf{diam}(J) \leq 2y\alpha r$. Now, for every $\mathsf{p} \in \partial yJ$, we have that $\mathsf{p} + \mathsf{ball}(0, (\varepsilon/c)\mathsf{diam}(yJ)) \subseteq (1+\varepsilon)yJ$. It is sufficient to show that $\mathsf{ball}(\mathsf{p}, (2\varepsilon y\alpha r/c)) \subseteq (1+\varepsilon)yJ$, for $\mathsf{p} \in \partial yJ$. Clearly $\mathsf{p}' = (1+\varepsilon)\mathsf{p} \in \partial(1+\varepsilon)yJ$. Since the cone, $\mathcal{CH}(\mathsf{ball}(\rho, (1+\varepsilon)yr) \cup \mathsf{p}')$ is in $(1+\varepsilon)yJ$, it is clear that the ball of radius, $x = \|\mathsf{p}' - \mathsf{p}\| \dfrac{(1+\varepsilon)yr}{\|\mathsf{p}'\|} = \|\mathsf{p}' - \mathsf{p}\| \dfrac{yr}{\|\mathsf{p}\|}$ is completely within $(1+\varepsilon)yJ$, see Figure 6.5. Now, $\|\mathsf{p} - \mathsf{p}'\| = \varepsilon \|\mathsf{p}\|$, so $x = \varepsilon yr$. For $c \geq 2\alpha$, the result follows. ∎

By the above lemma, we can take the growth function $\lambda_{f_{J_i}}(y) = \mathsf{diam}\Big((f_{J_i})_{\preceq y}\Big) = y\mathsf{diam}(J_i)$, and the growth constant, see (P2), for the set of functions f_{J_i}, to be $\zeta = c = 2\alpha$. If the object J is α-fat, but not α'-rounded fat for any constant $\alpha' > 0$, then it may be that its scaling distance function grows arbitrarily quickly, and thus is not a valid distance function for our framework, see Figure 6.6. It is not hard to see that Lemma 6.4.7 implies that bounded growth (P2) is satisfied for all the functions $f_1, \ldots, f_n$, when the objects under consideration $J_1, \ldots, J_n$, are α-rounded fat.

There is a small sketch and it can be computed quickly

To show that a small sketch exists (i.e., (P3) holds) is slightly harder and is tackled next.

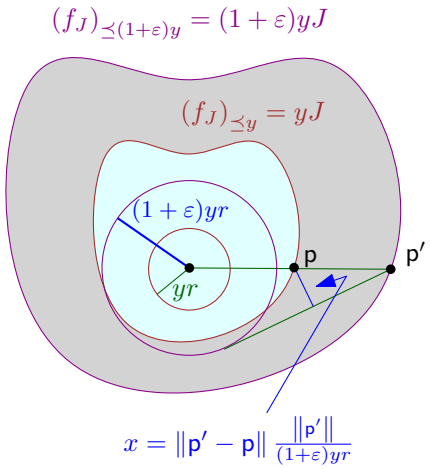


Figure 6.5: The $(1+\varepsilon)$ expansion of yJ contains $\mathsf{ball}(\mathsf{p}, x)$.

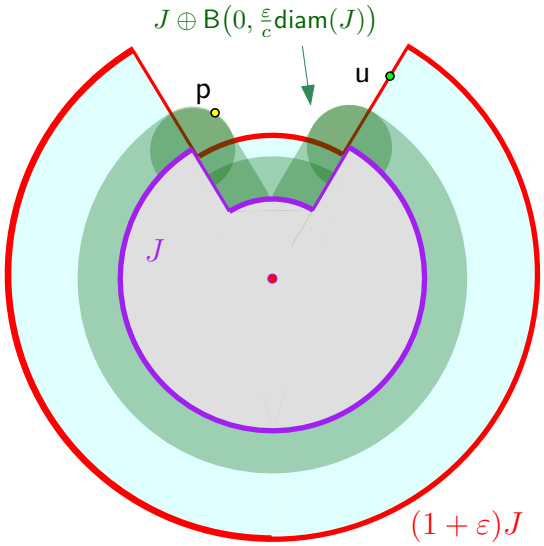


Figure 6.6: The object J is α-fat but not α'-rounded fat. In particular, the point p is in $J \oplus \mathsf{ball}(0, (\varepsilon/c)\mathsf{diam}(J))$ but not in $(1+\varepsilon)J$, and the scaling distance function is not well defined at u.

Lemma 6.4.8 *Let $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ be a set of n α-rounded fat objects centered at the origin, where $\alpha \geq 1$ is a fixed constant. Then, for any $\delta > 0$, there is a subset $\mathcal{I} \subseteq \{1, 2, \ldots, n\}$ with $|\mathcal{I}| = O(\delta^{-d})$, such that for all $y \geq 0$, we have $\bigcup_{i \in \{1,2,\ldots,n\}} yJ_i \subseteq \bigcup_{i \in \mathcal{I}} (1+\delta) yJ_i$. Moreover, for every $i \in \mathcal{I}$, we have that $\mathsf{diam}(J_i) = \Omega(\max_i \mathsf{diam}(J_i))$.*

Proof: Clearly it is sufficient to show this for $y = 1$. For $i = 1, \ldots, n$, let r_i be the radius of the "fatness" ball contained in J_i, see Definition 6.4.5. Let $\mathcal{K} = \left\{ i \;\middle|\; \alpha r_i \geq r_1 \right\}$ be the set of indices of relatively large objects (the other objects are sufficiently small and are contained in J_1). Observe that $\mathcal{K}$ is not empty as $1 \in \mathcal{K}$. We have that

$$\mathsf{diam}(J_i) \geq 2r_i \geq \frac{2}{\alpha} r_1 \geq \frac{1}{\alpha^2} \cdot 2\alpha r_1 \geq \frac{1}{\alpha^2} \max_{1 \leq i \leq n} \mathsf{diam}(J_i)\,,$$

for any $i \in \mathcal{K}$, since all the objects are centered at the origin.

Clearly, $\bigcup_{i \in [n]} J_i \subseteq \bigcup_{i \in [n]} \mathsf{ball}(0, \alpha r_i) \subseteq \mathsf{ball}(0, \alpha r_1)$. We tile the ball $\mathsf{ball}(0, \alpha r_1)$ with cubes of diameter $\delta \alpha r_1 / c'$ where $c' = c\alpha^2/2 = \alpha^3$. Let $\mathcal{C}$ denote the set of these cubes, and observe that $|\mathcal{C}| = O(\delta^{-d})$. For every cube $\mathsf{c} \in \mathcal{C}$, if it intersects $\mathsf{X} = \cup_{i \in \mathcal{K}} J_i$, then we add the index of one of the objects of $\mathcal{K}$ intersecting c to $\mathcal{I}$, and add c to $\mathcal{A}$.

Now, $\bigcup_{i \in [n]} J_i \subseteq \bigcup_{\mathsf{c} \in \mathcal{A}} \mathsf{c}$, as $\bigcup_{\mathsf{c} \in \mathcal{A}} \mathsf{c}$ covers $\mathsf{ball}(0, \alpha r_1)$. Observe that $|\mathcal{I}| = |\mathcal{A}| \leq |\mathcal{C}| = O(\delta^{-d})$. Next, we show that if c' is sufficiently large, then $\bigcup_{\mathsf{c} \in \mathcal{A}} \mathsf{c} \subseteq \bigcup_{i \in \mathcal{I}} (1+\delta) J_i$. Since $\mathsf{c} \cap J_i \neq \emptyset$, and $\mathsf{diam}(\mathsf{c}) \leq \delta \alpha r_1 / c'$,

$\mathsf{c} \subseteq J_i \oplus \mathsf{ball}(0, \delta\alpha r_1/c')$. By the choice of c' we have $\frac{\delta\alpha r_1}{c'} \leq \frac{\delta\alpha^2 r_i}{c'} \leq \frac{\delta\alpha^2 \mathsf{diam}(J_i)}{2c'} \leq \frac{\delta \mathsf{diam}(J_i)}{c}$, where $c = 2\alpha$ is the constant from Lemma 6.4.7. Then, by Lemma 6.4.7, we have

$$\mathsf{c} \subseteq J_i \oplus \mathsf{ball}\big(0, \delta\alpha r_1/c'\big) \subseteq J_i \oplus \mathsf{ball}\big(0, \delta \mathsf{diam}(J_i)/c\big) \subseteq (1+\delta)J_i,$$

which implies the claim. ■

Lemma 6.4.9 *Let $\alpha \geq 1$ be a constant, J an α-rounded fat object, $\delta > 0$, and let u be a point in $\mathbb{R}^d$, with $\|\mathsf{u}\| \leq \delta \mathsf{diam}(J)/c$, where $c = 2\alpha$. Then $J + \mathsf{u} \subseteq (1+\delta)J$.*

Proof: We have, $J + \mathsf{u} \subseteq J \oplus \mathsf{ball}(0, \|\mathsf{u}\|) \subseteq J \oplus \mathsf{ball}(0, \delta\mathsf{diam}(J)/c)$ as $\|\mathsf{u}\| \leq \delta\mathsf{diam}(J)/c$. Now, the claim follows by Lemma 6.4.7. ■

Lemma 6.4.10 *Let $\mathcal{J} = \{J_1, \ldots, J_n\}$ be a set of n α-rounded fat object in $\mathbb{R}^d$, let $\mathcal{F}$ be the set of scaling distance functions of these objects, and let P be the set of offset points (i.e., centers) of the objects of $\mathcal{J}$. Then $\ell_{\mathrm{con}}(\mathcal{F}) \geq \mathsf{diam}(\mathsf{P})/2n\alpha r$, where $r = \max_i r_i$, and r_i is the inner radius of J_i, see Definition 6.4.5.*

Proof: Let $\mathcal{F} = \big\{f_i = f_{J_i} \big| i = 1, \ldots, n\big\}$ and $\mathsf{P} = \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}$. Here, for $i = 1, \ldots, n$, p_i is the center of J_i. The claim is trivially true if $\mathsf{diam}(\mathsf{P}) = 0$, i.e., all the points p_i are the same. Let $\ell = \ell_{\mathrm{con}}(\mathcal{F})$. As $(f_i)_{\preceq \ell} = \ell J_i$, where the scaling for object J_i is done around its center p_i, it follows that $\ell\mathcal{J} = \Big\{\ell J_i \;\Big|\; J_i \in \mathcal{F}\Big\}$ is connected, see Definition 6.2.5. Since $\ell J_i \subseteq \mathsf{ball}(\mathsf{p}_i, \ell\alpha r_i) \subseteq \mathsf{ball}(\mathsf{p}_i, l\alpha r)$, it follows that $\mathcal{B} = \Big\{\mathsf{ball}(\mathsf{p}_i, \ell\alpha r) \;\Big|\; i = 1, \ldots, n\Big\}$ is also connected. Let $\mathsf{u}, \mathsf{v} \in \mathsf{P}$ be such that $\|\mathsf{u} - \mathsf{v}\| = \mathsf{diam}(\mathsf{P})$. There is a sequence of distinct $i_1, \ldots, i_k \in \{1, \ldots, n\}$, such that $\mathsf{u} = \mathsf{p}_{i_1}, \mathsf{v} = \mathsf{p}_{i_k}$, and $\mathsf{ball}(\mathsf{p}_{i_t}, \ell\alpha r) \cap \mathsf{ball}\big(\mathsf{p}_{i_{t+1}}, \ell\alpha r\big) \neq \emptyset$, and thus $\big\|\mathsf{p}_{i_t} - \mathsf{p}_{i_{t+1}}\big\| \leq 2\ell\alpha r$, for $t = 1, \ldots, k-1$. By the triangle inequality,

$$\mathsf{diam}(\mathsf{P}) = \|\mathsf{u} - \mathsf{v}\| = \|\mathsf{p}_{i_1} - \mathsf{p}_{i_k}\| \leq \sum_{t=1}^{k-1} \big\|\mathsf{p}_{i_t} - \mathsf{p}_{i_{t+1}}\big\| \leq \sum_{t=1}^{k-1} 2\ell\alpha r = 2(k-1)\ell\alpha r \leq 2n\ell\alpha r,$$

■

We can now show that condition (P3) holds for the f_{J_i}.

Lemma 6.4.11 *Consider the setting of Lemma 6.4.10. Given $\delta > 0$, there is a index set $\mathcal{I} \subseteq \{1, \ldots, n\}$, with $|\mathcal{I}| = O\big(\delta^{-d}\big)$, and $y_0 = O(\ell \cdot n/\delta)$, such that the functions $\Big\{f_j \;\Big|\; j \in \mathcal{I}\Big\}$, form a (δ, y_0)-sketch, where $\ell = \ell_{\mathrm{con}}(\mathcal{F})$.*

Proof: We provide a sketch of the proof, as the details are easy but tedious. Consider the set of objects $J_{ij} = J_i + \mathsf{p}_j - \mathsf{p}_i$, for each pair (i, j) with $1 \leq i, j \leq n$. Clearly, J_{ij} is J_i translated, so that it is centered at p_j.

By Lemma 6.4.8 there is an index set $\mathcal{I} \subseteq \{1, \ldots, n\}$, with $|\mathcal{I}| = O\left(\delta^{-d}\right)$, such that for all y and any fixed j with $1 \leq j \leq n$, we have that $\bigcup_{i \in [n]} yJ_{ij} \subseteq \bigcup_{i \in \mathcal{I}} (1+\delta/4) yJ_{ij}$. Let r_i denote the radius of the ball for J_i from Definition 6.4.5, and let $r = \max_i r_i$. By Lemma 6.4.10, we have that, $\ell \geq \mathsf{diam}(\mathsf{P})/(2n\alpha r)$. Lemma 6.4.8 finds a $\mathcal{I}$ such that for all $i \in \mathcal{I}$, $r_i \geq \Omega(r)$. A translated copy $J_{ij} = J_i + \mathsf{p}_j - \mathsf{p}_i$, is a translation of J_i by a vector $\mathsf{u} = \mathsf{p}_j - \mathsf{p}_i$. As $\ell \geq \mathsf{diam}(\mathsf{P})/(2n\alpha r)$, there is a $y_0 = O(\ell n/\delta)$ such that $\|\mathsf{p}_j - \mathsf{p}_i\| \leq \delta \mathsf{diam}(y_0 J_i)/4c$ for all $1 \leq i, j \leq n$, where $c = 2\alpha$. Thus using Lemma 6.4.9, $(1+\delta/4) y_0 J_i + (\mathsf{p}_j - \mathsf{p}_i) \subseteq (1+\delta/4)^2 y_0 J_i \subseteq (1+\delta) y_0 J_i$. Clearly this also holds for any $y \geq y_0$. Thus for $y \geq y_0$ we have $(1+\delta) yJ_i$, covers $yJ_i + (\mathsf{p}_j - \mathsf{p}_i)$ for $1 \leq i \leq n$. It is then easy to see that $\left\{ f_i \;\middle|\; i \in \mathcal{I} \right\}$, is a (δ, y_0)-sketch. ■

The result

We conclude that for α-rounded fat objects, the scaling distance functions they define falls under our framework. From the above, and the requirement that the sketch constant c_{sk} be at least $d+1$ for our bounds to hold, it follows that we can choose the sketch constant $\mathsf{c}_{\mathrm{sk}} = \max(d, d+1) = d+1$, and this concludes the proof of Theorem 6.2.16.

Note that the result in Theorem 6.2.16 covers any symmetric convex metric. Indeed, given a convex symmetric shape C centered at the origin, and with constant boundary complexity, the distance it induces for any pair of points $\mathsf{p}, \mathsf{u} \in \mathbb{R}^d$, is the scaling distance of C centered at p to u (or, by symmetry, the scaling distance of p from C centered at u). Under this distance $\mathbb{R}^d$ is a metric space, and of course, the triangle inequality holds. By an appropriate scaling of space, which does not affect the norm (except for scaling it) we can make C fat, and now Theorem 6.2.16 applies. Of course, Theorem 6.2.16 is considerably more general, allowing each of the points to induce a different scaling distance function, and the distance induced does not have to obey the triangle inequality.

6.4.3 Nearest furthest-neighbor

For a set of points $\mathsf{S} \subseteq \mathbb{R}^d$, and a point q, the ***furthest-neighbor distance*** of q from S, is $\mathrm{F}_{\mathsf{S}}(\mathsf{q}) = \max_{\mathsf{s} \in \mathsf{S}} \|\mathsf{s} - \mathsf{q}\|$; that is, it is the furthest one might have to travel from q to arrive to a point of S. For example, S might be the set of locations of facilities, where it is known that one of them is always open, and one is interested in the worst case distance a client has to travel to reach an open facility. The function $\mathrm{F}_{\mathsf{S}}(\cdot)$ is known as the ***furthest-neighbor Voronoi*** diagram, and while its worst case combinatorial complexity is similar to the regular Voronoi diagram, it can be approximated using a constant size representation (for constant dimensions), see [Har99] – one can compute, in linear time, a subset $\mathsf{U} \subseteq \mathsf{S}$, of size $O\left(\varepsilon^{-d} \log \varepsilon^{-1}\right)$, such that $(1-\varepsilon)\mathrm{F}_{\mathsf{S}}(\mathsf{q}) \leq \mathrm{F}_{\mathsf{U}}(\mathsf{q}) \leq \mathrm{F}_{\mathsf{S}}(\mathsf{q})$.

Given n sets of points $\mathsf{P}_1, \ldots, \mathsf{P}_n$ in $\mathbb{R}^d$, we are interested in the distance function $\mathrm{F}(\mathsf{q}) = \min_i \mathrm{F}_i(\mathsf{q})$, where $\mathrm{F}_i(\mathsf{q}) = \mathrm{F}_{\mathsf{P}_i}(\mathsf{q})$. This quantity arises naturally when one tries to model uncertainty; indeed, let P_i be the set of possible locations of the ith ***uncertain*** point (i.e., the location of the ith point is chosen randomly, somehow, from the set P_i). Thus, $\mathrm{F}_i(\mathsf{q})$ is the worst case distance to the ith point, and $\mathrm{F}(\mathsf{q})$ is the worst-case nearest-neighbor distance to the random point-set generated by picking the ith point from P_i, for $i = 1, \ldots, n$. We refer to $\mathrm{F}(\cdot)$ as the ***nearest furthest-neighbor*** Voronoi diagram, and we are interested in its approximation.

6.4.4 Satisfaction of conditions

Observation 6.4.12 *We have that* $(\mathrm{F}_i)_{\preceq y} = \bigcap_{\mathsf{u} \in \mathsf{P}_i} \mathsf{ball}(\mathsf{u}, y)$, *and* $\mathsf{diam}\Big((\mathrm{F}_i)_{\preceq y}\Big) \leq 2y$.

Given the above observation, it is easy to see that Condition (P1) is true, as $(\mathrm{F}_i)_{\preceq y}$ is a finite intersection of compact sets. The following Lemma shows that Condition (P2) is also true, by letting the growth function $\lambda_{(\mathrm{F}_i)}(y) = y$. Since $y \geq \mathsf{diam}\Big((\mathrm{F}_i)_{\preceq y}\Big)/2$ by Observation 6.4.12, it follows that we can choose the growth constant ζ to be 2.

Lemma 6.4.13 *For any i with $1 \leq i \leq n$, if $(\mathrm{F}_i)_{\preceq y} \neq \emptyset$, we have $(\mathrm{F}_i)_{\preceq y} \oplus \mathsf{ball}(0, \varepsilon y) \subseteq (\mathrm{F}_i)_{\preceq (1+\varepsilon)y}$.*

Proof: Consider any point q in $(\mathrm{F}_i)_{\preceq y} \oplus \mathsf{ball}(0, \varepsilon y)$. It is easy to see that $\mathsf{ball}(\mathsf{q}, (1+\varepsilon)y) \supseteq \mathsf{P}_i$ by the triangle inequality, and so $\mathsf{q} \in (\mathrm{F}_i)_{\preceq (1+\varepsilon)y}$. ■

Lemma 6.4.14 (Existence of a sketch) *Let $\mathcal{G} \subseteq \{\mathrm{F}_1, \ldots, \mathrm{F}_n\}$, denote a set of functions as above. Then, given any $\delta > 0$, there is a subset $\mathcal{H} \subseteq \mathcal{G}$ with $|\mathcal{H}| = 1$, and a y_0 with $y_0 = O(\ell_{\mathrm{con}}(\mathcal{G})\,|\mathcal{G}|\,/\delta)$, such that $\mathcal{H}$ is a (δ, y_0)-sketch for $\mathcal{G}$.*

Proof: Let $\mathcal{G} = \{\mathrm{F}_1, \ldots, \mathrm{F}_m\}$, where $m = |\mathcal{G}|$, and $z = \ell_{\mathrm{con}}(\mathcal{G})$. Now, $(\mathrm{F}_i)_{\preceq z}$ for $i = 1, \ldots, m$, are all connected, and by Observation 6.4.12 $\mathsf{diam}\Big((\mathrm{F}_i)_{\preceq z}\Big) \leq 2z$. Thus, $\forall \mathsf{u} \in \mathsf{P}_j, \forall \mathsf{v} \in \mathsf{P}_k$, there are points $\mathsf{u}' \in (\mathrm{F}_j)_{\preceq z}$, and $\mathsf{v}' \in (\mathrm{F}_k)_{\preceq z}$, such that $\|\mathsf{u}' - \mathsf{u}\| \leq z$, $\|\mathsf{v}' - \mathsf{v}\| \leq z$ (by definition of the function F_j and F_k respectively), and $\|\mathsf{v}' - \mathsf{u}'\| \leq 2mz$, by the bound on the diameter of the sublevel sets and the condition of being connected, which is the same as the intersection graph of the sets being connected. It follows, by the triangle inequality, that $\|\mathsf{v} - \mathsf{u}\| \leq 2(m+1)z \leq 4mz$; namely, $\mathsf{diam}(\mathsf{P}_1 \cup \cdots \cup \mathsf{P}_m) \leq 4mz$. Let $\mathcal{H}$ be a set containing an arbitrary function from $\mathcal{G}$ say, $\mathcal{H} = \{\mathrm{F}_1\}$.

It is easy to see, that for $y \geq y_0 = 4mz/\delta$, and for all $\mathsf{u}, \mathsf{v} \in \mathsf{P}_1 \cup \cdots \cup \mathsf{P}_m$, we have that $\mathsf{ball}(\mathsf{v}, y) \subseteq \mathsf{ball}(\mathsf{u}, (1+\delta)y)$. Thus, for any i, $1 \leq i \leq m$, we have that $(\mathrm{F}_i)_{\preceq y} = \bigcap_{\mathsf{v} \in \mathsf{P}_i} \mathsf{ball}(\mathsf{v}, y) \subseteq \mathsf{ball}(\mathsf{u}, (1+\delta)y)$, for all $\mathsf{u} \in \mathsf{P}_1$ and $y \geq y_0$. As such, $(\mathrm{F}_i)_{\preceq y} \subseteq \bigcap_{\mathsf{u} \in \mathsf{P}_1} \mathsf{ball}(\mathsf{u}, (1+\delta)y) = (\mathrm{F}_1)_{\preceq (1+\delta)y}$, and the result follows. ■

Remark 6.4.15 *The satisfaction of the computability conditions (C1)–(C3) is not too hard to see, though it is tedious. We need to use a data-structure for approximate furthest-neighbor queries, as well as coresets for approximately computing the* MEB *of the point sets. The fact that the center of the* MEB *is not perturbed too much when using coresets, is guaranteed by Lemma B.0.7, presented in Appendix B, as it may also be of independent interest.*

The result

From the above, and the requirement that the sketch constant c_{sk} be at least $d+1$ for our bounds to hold, it follows that for the functions under consideration, we can choose the sketch constant $\mathsf{c}_{\mathrm{sk}} = \max(d, d+1) = d+1$.

Proof of 3 Theorem 6.2.17: *We only need to show how to get the improved space and query time – i.e., how do we get rid of the total number of points m in the space and query time. Observe that every one of the sets P_i can be replaced by a subset $\mathsf{S}_i \subseteq \mathsf{P}_i$, of size $O(1/\varepsilon^d \log(1/\varepsilon))$, such that for any point $\mathsf{q} \in \mathbb{R}^d$, we have that $\mathrm{F}_{\mathsf{S}_i}(\mathsf{q}) \leq \mathrm{F}_{\mathsf{P}_i}(\mathsf{q}) \leq (1+\varepsilon/4)\mathrm{F}_{\mathsf{S}_i}(\mathsf{q})$. Such a subset can be computed in $O(|\mathsf{P}_i|)$ time, see [Har99]. We thus perform this transformation for each one of the uncertain point sets $\mathsf{P}_1, \ldots, \mathsf{P}_n$, which reduces the input size to $O(n/\varepsilon^d \log(1/\varepsilon))$. It is easy to verify that all the other operations required to construct the data-structure can be done efficiently – for example, computing the sketch requires approximating the diameter up to a constant factor, which can easily be done in linear time. We now apply our main result to the distance functions induced by the reduced sets $\mathsf{S}_1, \ldots, \mathsf{S}_n$.*

6.5 Conclusions

In this chapter, we investigated what classes of functions have minimization diagrams that can be approximated efficiently – where our emphasis was on distance functions. We defined a general framework and the requirements on the distance functions to fall under it. For such functions, we presented a new data-structure, with near-linear space and preprocessing time, so that it can evaluate (approximately) the minimization diagram of a query point in logarithmic time. Surprisingly, one gets an AVD (approximate Voronoi diagram) of this complexity; that is, a decomposition of space with near-linear complexity, such that for every region of this decomposition a single function serves as an ANN for all points in this region.

We also showed some interesting classes of functions for which we get this AVD. For example, multiplicative weighted distance functions with additive offsets. No previous results of this kind were known, and even in the plane, multiplicative Voronoi diagrams have quadratic complexity in the worst case (for which

the new AVD has near-linear complexity). The framework also works for Minkowski metrics of fat convex bodies, and nearest furthest-neighbor. However, it seems that our main result applies to even more general distance functions.

Several questions remain open for further research:

(A) Are the additional polylog factors in the space necessary? In particular, it seems unlikely that using WSPD's directly, as done by Arya and Malamatos [AM02], should work in the most general settings, so reducing the logarithmic dependency seems quite interesting. Specifically, can the Arya and Malamatos construction [AM02] be somehow adapted to this framework, possibly with some additional constraints on the functions, to get a linear space construction?

(B) On the applications side, are constant degree polynomials a dependable family amenable to our framework? Specifically, consider a polynomial $\tau(x)$ that is positive for all $x \geq 0$. Given a point u, we associate the distance function $f(\mathsf{q}) = \tau(\|\mathsf{u} - \mathsf{q}\|)$ with u. Given a set of such distance functions, under which conditions, can one build an AVD for these functions efficiently? (It is not hard to see that in the general case this is not possible under our framework.)

Appendix A

Basic properties of the functions

Lemma A.0.1 *Let $\mathcal{F}$ be a set of functions that satisfy the compactness (P1) and bounded growth (P2) conditions. Then, for any $f \in \mathcal{F}$, either $f_{\preceq 0} = \emptyset$ or $f_{\preceq 0}$ consists of a single point.*

Proof: If $f_{\preceq 0}$ contains two distinct points $x, y \in f_{\preceq 0}$, such that $\|x - y\| = \mathsf{diam}(f_{\preceq 0}) > 0$. By the bounded growth (P2) it follows that,

$$f_{\preceq 0} \subseteq f_{\preceq 0} \oplus \mathsf{ball}\left(0, \frac{\|x-y\|}{\zeta}\right) \subseteq f_{\preceq 0} \oplus \mathsf{ball}(0, \lambda_f(0)) \subseteq f_{\preceq 0},$$

using $\varepsilon = 1$, and as $\lambda_f(0) \geq \mathsf{diam}(f_{\preceq 0})/\zeta = \|x-y\|/\zeta$. Thus, $f_{\preceq 0} \oplus \mathsf{ball}\left(0, \frac{\|x-y\|}{\zeta}\right) = f_{\preceq 0}$. Clearly in $y \oplus \mathsf{ball}\left(0, \frac{\|x-y\|}{\zeta}\right)$, there is some y' such that $\|x - y'\| > \|x - y\|$. Namely, $\mathsf{diam}(f_{\preceq 0}) > \mathsf{diam}(f_{\preceq 0})$, which is a contradiction. ■

We assume that that $\mathbb{d}(f, g) > 0$ for $f \neq g$ (this can be enforced using symbolic perturbations). Similarly, we also assume that the distances $\mathbb{d}(f, g)$ are distinct for all distinct pairs of functions.

Observation A.0.2 *If $\ell_{\mathrm{con}}(\mathcal{F}) = 0$ for any non-empty set $\mathcal{F}$, then $|\mathcal{F}| = 1$.*

Lemma A.0.3 *Let $f \in \mathcal{F}$ and $y \geq 0$. Suppose $\mathsf{u}, \mathsf{v} \in f_{\preceq y}$. Then, $\mathsf{uv} \subseteq \mathcal{F}_{\preceq (1+\zeta/2)y}$, where uv denotes the segment joining u to v.*

Proof: If $\mathsf{u} = \mathsf{v}$, the claim is obvious. Using bounded growth (P2) with $\varepsilon = \zeta/2$, and the inequality $\lambda_f(y) \geq \mathsf{diam}(f_{\preceq y})/\zeta$, it follows that $f_{\preceq y} \oplus \mathsf{ball}(0, \mathsf{diam}(f_{\preceq y})/2) \subseteq f_{\preceq (1+\zeta/2)y}$. Thus, $\mathsf{u} \oplus \mathsf{ball}(0, \mathsf{diam}(f_{\preceq y})/2) \subseteq f_{\preceq (1+\zeta/2)y}$ as well as $\mathsf{ball}(\mathsf{v}, \mathsf{diam}(f_{\preceq y})/2) \subseteq f_{\preceq (1+\zeta/2)y}$. Since $\|\mathsf{u} - \mathsf{v}\| \leq \mathsf{diam}(f_{\preceq y})$, it follows that the entire segment uv is in $f_{\preceq (1+\zeta/2)y}$. ■

The following is easy to verify.

Lemma A.0.4 *Let $A_1, \ldots, A_m \subseteq \mathbb{R}^d$, be compact connected sets. Let uv be any segment. Suppose that $\mathsf{uv} \cap A_i \neq \emptyset$ for all $1 \leq i \leq k$, and $\mathsf{uv} \subseteq \bigcup_{i=1}^{k} A_i$. Then, the set $\{A_1, \ldots, A_k\}$ is connected, see Definition 6.2.5.*

Lemma A.0.5 *Suppose we are given $\mathcal{G} \subseteq \mathcal{F}$, $\delta \geq 0$ and $y \geq 0$, such that $\mathcal{G}$ is a (δ, y)-sketch for $\mathcal{F}$. Then, $\ell_{\text{con}}(\mathcal{G}) \leq (1+\delta)(1+\zeta/2)\max(y, \ell_{\text{con}}(\mathcal{F}))$.*

Proof: Assume that $\mathcal{F} = \{f_1, \ldots, f_m\}$, and $\mathcal{G} = \{f_1, \ldots, f_k\}$, where $k \leq m$. If $m = 1$ then $k = 1$, and we have by definition $\ell_{\text{con}}(\mathcal{F}) = \ell_{\text{con}}(\mathcal{G}) = 0$, and the result clearly holds true. If $m > 1$, we need to show that $(f_i)_{\preceq y'}$, for $i = 1, \ldots, k$, are connected, where $y' = (1+\delta)(1+\zeta/2)l$, and $\ell = \max(y, \ell_{\text{con}}(\mathcal{F}))$. Now by definition, $\mathcal{F}_{\preceq \ell}$ is a connected set. Consider any $1 \leq i \neq j \leq k$. Then there is a sequence of distinct indices $i = i_1, i_2, \ldots, i_s = j$ such that $(f_{i_r})_{\preceq \ell} \cap \left(f_{i_{r+1}}\right)_{\preceq \ell} \neq \emptyset$ for $1 \leq r \leq s-1$. Consider any such index, say i_r, such that $i_r > k$, i.e., $f_{i_r} \notin \mathcal{G}$. Since, $(f_{i_r})_{\preceq \ell} \cap \left(f_{i_{r-1}}\right)_{\preceq \ell} \neq \emptyset$ and $(f_{i_r})_{\preceq \ell} \cap \left(f_{i_{r+1}}\right)_{\preceq \ell} \neq \emptyset$ we can choose points $\mathsf{u} \in \left(f_{i_{r-1}}\right)_{\preceq \ell} \cap (f_{i_r})_{\preceq \ell}$ and $\mathsf{v} \in (f_{i_r})_{\preceq \ell} \cap \left(f_{i_{r+1}}\right)_{\preceq \ell}$. Now the entire segment $\mathsf{uv} \subseteq (f_{i_r})_{\preceq (1+\zeta/2)l}$, by Lemma A.0.3. Since $(1+\zeta/2)l \geq y$, it follows by the sketch property (P3), that $\mathsf{uv} \subseteq (f_{i_r})_{\preceq (1+\zeta/2)l} \subseteq \mathcal{G}_{\preceq (1+\zeta/2)(1+\delta)l}$. By Lemma A.0.4, the sets in the minimal cover of uv by the sublevel sets $(f_i)_{\preceq (1+\zeta/2)(1+\delta)l}, 1 \leq i \leq k$, are connected. It follows that $(f_{i_r})_{\preceq (1+\zeta/2)l}$ can be replaced by a sub-collection of the $(f_i)_{\preceq (1+\zeta/2)(1+\delta)l}, 1 \leq i \leq k$, and the property of neighbor intersections is still valid in the chain. We replace each occurrence of the set $(f_{i_r})_{\preceq (1+\zeta/2)l}$ for $i_r > k$, by the corresponding chain. It is easy to see that the resulting chain connects up $(f_{i_1})_{\preceq (1+\zeta/2)(1+\delta)l}$ and $(f_{i_s})_{\preceq (1+\zeta/2)(1+\delta)l}$. Now, duplicate elements can be easily removed without affecting the neighbor intersection property of the chain. ■

The following testifies that a sketch approximates the distance of a set of functions.

Lemma A.0.6 *Let $\mathcal{G} \subseteq \mathcal{F}$ be sets of functions, where $\mathcal{G}$ is a (δ, y_0)-sketch for $\mathcal{F}$ for some $\delta \geq 0$ and $y_0 \geq 0$. Let q be a point such that $\mathbb{d}(\mathsf{q}, \mathcal{F}) \geq y_0$. Then we have that $\mathbb{d}(\mathsf{q}, \mathcal{G}) \leq (1+\delta)\mathbb{d}(\mathsf{q}, \mathcal{F})$.*

Proof: Let $\ell = \mathbb{d}(\mathsf{q}, \mathcal{F})$ and let $f \in \mathcal{F}$ be a witness that $\mathsf{q} \in f_{\preceq \ell}$. As $\ell \geq y_0$ we have that $f_{\preceq \ell} \subseteq \bigcup_{g \in \mathcal{G}} g_{\preceq (1+\delta)l}$ by the sketch property (Definition 6.2.9). As such there is some function $g \in \mathcal{G}$, such that $\mathsf{q} \in g_{\preceq (1+\delta)l}$. It follows that $\mathbb{d}(\mathsf{q}, g) \leq (1+\delta)\mathbb{d}(\mathsf{q}, \mathcal{F})$. ■

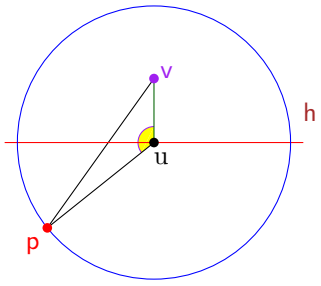
Appendix B

Bounding the size of intersection of balls of the same radius

Lemma B.0.7 *Let* $\mathsf{ball}(\mathrm{u},\mathrm{z})$ *be the MEB (minimum enclosing ball) of a set of points* $\mathsf{P} \subseteq \mathbb{R}^d$. *Then, for any* $\delta \in (0,1)$, *we have* $\mathsf{ball}(\mathrm{u},\delta\mathrm{z}) \subseteq \bigcap_{\mathsf{p}\in\mathsf{P}} \mathsf{ball}(\mathsf{p},(1+\delta)\mathrm{z}) \subseteq \mathsf{ball}\Big(\mathrm{u}, \sqrt{4\delta+2\delta^2}\,\mathrm{z}\Big)$.

Proof: There are affinely independent points from P, on the surface of the MEB, such that u lies in their convex hull (otherwise the MEB is not minimal). Thus, there is a set $\mathsf{S} = \{\mathsf{p}_1,\ldots,\mathsf{p}_k\} \subseteq \mathsf{P}$, such that (i) $\forall \mathsf{p} \in \mathsf{S} \quad \|\mathsf{p}-\mathrm{u}\| = \mathrm{z}$, and (ii) $\mathrm{u} \in \mathcal{CH}(\mathsf{S})$. Thus, any closed halfspace h^+ that its boundary passes through u must contain at least one of the points of S.

Consider any point $\mathsf{v} \in \bigcap_{\mathsf{p}\in\mathsf{S}} \mathsf{ball}(\mathsf{p},(1+\delta)\mathrm{z})$. Let h be the hyperplane passing through u, and orthogonal to $\mathrm{u}-\mathsf{v}$, and let h^+ be the closed halfspace having h on its boundary that does not contain v. Then, by the above observation, there must be a point $\mathsf{p} \in \mathsf{S}$, such that $\mathsf{p} \in \mathsf{h}^+$. Now, by the law of cosines, we have that



$$\begin{aligned}(1+\delta)^2\mathrm{z}^2 &\geq \|\mathsf{p}-\mathsf{v}\|^2 = \|\mathrm{u}-\mathsf{v}\|^2 + \|\mathsf{p}-\mathrm{u}\|^2 - 2\,\|\mathrm{u}-\mathsf{v}\|\,\|\mathsf{p}-\mathrm{u}\|\cos\angle\mathsf{vup} \\ &\geq \|\mathrm{u}-\mathsf{v}\|^2 + \|\mathsf{p}-\mathrm{u}\|^2 = \|\mathrm{u}-\mathsf{v}\|^2 + \mathrm{z}^2,\end{aligned}$$

since $\angle\mathsf{vup} \geq \pi/2$. Rearranging, we have $(2\delta+\delta^2)\mathrm{z}^2 \geq \|\mathrm{u}-\mathsf{v}\|^2$, thus implying $\sqrt{2\delta+\delta^2}\mathrm{z} \geq \|\mathrm{u}-\mathsf{v}\|$, We conclude that $\bigcap_{\mathsf{p}\in\mathsf{P}} \mathsf{ball}(\mathsf{p},(1+\delta)\mathrm{z}) \subseteq \bigcap_{\mathsf{p}\in\mathsf{S}} \mathsf{ball}(\mathsf{p},(1+\delta)\mathrm{z}) \subseteq \mathsf{ball}\Big(\mathrm{u}, \sqrt{2\delta+\delta^2}\,\mathrm{z}\Big)$. ■

References

[AAH+13] P. K. Agarwal, B. Aronov, S. Har-Peled, J. M. Phillips, K. Yi, and W. Zhang. Nearest neighbor searching under uncertainty ii. In *Proc. 32nd ACM Sympos. Principles Database Syst.* (PODS), pages 115–126, 2013.

[AC09] N. Ailon and B. Chazelle. The fast johnson-lindenstrauss transform and approximate nearest neighbors. *SIAM J. Comput.*, 39(1):302–322, 2009.

[AE99] P. K. Agarwal and J. Erickson. Geometric range searching and its relatives. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, pages 1–56. Amer. Math. Soc., 1999.

[AM93a] P. K. Agarwal and J. Matoušek. Ray shooting and parametric search. *SIAM J. Comput.*, 22:540–570, 1993.

[AM93b] S. Arya and D. M. Mount. Approximate nearest neighbor queries in fixed dimensions. In *Proc. 4th ACM-SIAM Sympos. Discrete Algs.* (SODA), pages 271–280, Philadelphia, PA, USA, 1993. Society for Industrial and Applied Mathematics.

[AM98] S. Arya and D. Mount. ANN: library for approximate nearest neighbor searching. `http://www.cs.umd.edu/~mount/ANN/`, 1998.

[AM00] S. Arya and D. M. Mount. Approximate range searching. *Comput. Geom. Theory Appl.*, 17:135–152, 2000.

[AM02] S. Arya and T. Malamatos. Linear-size approximate Voronoi diagrams. In *Proc. 13th ACM-SIAM Sympos. Discrete Algs.* (SODA), pages 147–155, 2002.

[AMM02] S. Arya, T. Malamatos, and D. M. Mount. Space-efficient approximate Voronoi diagrams. In *Proc. 34th Annu. ACM Sympos. Theory Comput.* (STOC), pages 721–730, 2002.

[AMM05] S. Arya, T. Malamatos, and D. M. Mount. Space-time tradeoffs for approximate spherical range counting. In *Proc. 16th ACM-SIAM Sympos. Discrete Algs.* (SODA), pages 535–544, 2005.

[AMM09] S. Arya, T. Malamatos, and D. M. Mount. Space-time tradeoffs for approximate nearest neighbor searching. *J. Assoc. Comput. Mach.*, 57(1):1–54, 2009.

[AMN+98] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *J. Assoc. Comput. Mach.*, 45(6):891–923, 1998.

[Ass83] P. Assouad. Plongements lipschitziens dans $\mathbf{R}^n$. *Bull. Soc. Math. France*, 111(4):429–448, 1983.

[Aur91] F. Aurenhammer. Voronoi diagrams: A survey of a fundamental geometric data structure. *ACM Comput. Surv.*, 23:345–405, 1991.

[Ber93] M. Bern. Approximate closest-point queries in high dimensions. *Inf. Process. Lett.*, 45(2):95–99, February 1993.

[Bes96] S. N. Bespamyatnikh. Dynamic algorithms for approximate neighbor searching. In *Proc. 8th Canad. Conf. Comput. Geom.* (CCCG), pages 252–257, 1996.

[BHTT10] M. de Berg, H. Haverkort, S. Thite, and L. Toma. Star-quadtrees and guard-quadtrees: I/O-efficient indexes for fat triangulations and low-density planar subdivisions. *Comput. Geom. Theory Appl.*, 43:493–513, July 2010.

[BK73] W. A. Burkhard and R. M. Keller. Some approaches to best-match file searching. *Commun. ACM*, 16(4):230–236, April 1973.

[BKL06] A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *Proc. 24rd Int. Conf. Mach. Learning*, pages 97–104, 2006.

[Bri95] S. Brin. Near neighbor search in large metric spaces. In *Proc. 21st Intl. Conf. Very Large Data Bases*, pages 574–584, 1995.

[BSLT00] M. Bernstein, V. De Silva, J. C. Langford, and J. B. Tenenbaum. Graph approximations to geodesics on embedded manifolds, 2000.

[CDH+05] P. Carmi, S. Dolev, S. Har-Peled, M. J. Katz, and M. Segal. Geographic quorum systems approximations. *Algorithmica*, 41(4):233–244, 2005.

[CG06] R. Cole and L. Gottlieb. Searching dynamic point sets in spaces with bounded doubling dimension. In *Proc. 38th Annu. ACM Sympos. Theory Comput.* (STOC), pages 574–583, 2006.

[Cha98] T. M. Chan. Approximate nearest neighbor queries revisited. *Discrete Comput. Geom.*, 20:359–373, 1998.

[Cha02] T. M. Chan. Closest-point problems simplified on the ram. In *Proc. 13th ACM-SIAM Sympos. Discrete Algs.* (SODA), pages 472–473. Society for Industrial and Applied Mathematics, 2002.

[Cha06] T. M. Chan. A minimalist's implementation of an approximate nearest neighbor algorithm in fixed dimensions. Manuscript, 2006.

[Cha10] T. M. Chan. Optimal partition trees. In *Proc. 26th Annu. ACM Sympos. Comput. Geom.* (SoCG), pages 1–10, 2010.

[CK95] P. B. Callahan and S. R. Kosaraju. A decomposition of multidimensional point sets with applications to k-nearest-neighbors and n-body potential fields. *J. Assoc. Comput. Mach.*, 42:67–90, 1995.

[Cla88] K. L. Clarkson. A randomized algorithm for closest-point queries. *SIAM J. Comput.*, 17:830–847, 1988.

[Cla94] K. L. Clarkson. An algorithm for approximate closest-point queries. In *Proc. 10th Annu. ACM Sympos. Comput. Geom.* (SoCG), pages 160–164, 1994.

[Cla99] K. L. Clarkson. Nearest neighbor queries in metric spaces. *Discrete Comput. Geom.*, 22(1):63–93, 1999.

[Cla06] K. L. Clarkson. Nearest-neighbor searching and metric space dimensions. In G. Shakhnarovich, T. Darrell, and P. Indyk, editors, *Nearest-Neighbor Methods for Learning and Vision: Theory and Practice*, pages 15–59. MIT Press, 2006.

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

[DGL96] L. Devroye, L. Györfi, and G. Lugosi. *A probabilistic theory of pattern recognition.* Spring-Verlag, New York, 1996.

[DHS01] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, New York, 2nd edition, 2001.

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

[Dun99] C. A. Duncan. *Balanced Aspect Ratio Trees*. Ph.D. thesis, Department of Computer Science, Johns Hopkins University, Baltimore, Maryland, 1999.

[Eri96] J. Erickson. New lower bounds for Hopcroft's problem. *Discrete Comput. Geom.*, 16:389–418, 1996.

[FS82] C. D. Feustel and L. G. Shapiro. The nearest neighbor problem in an abstract metric space. *Pattern Recognition Letters*, 1(2):125–128, 1982.

[GK11] L.A. Gottlieb and R. Krauthgamer. A nonlinear approach to dimension reduction. In *Proc. 22nd ACM-SIAM Sympos. Discrete Algs.* (SODA), pages 888–899, 2011.

[GKL03] A. Gupta, R. Krauthgamer, and J. R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *Proc. 44th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), pages 534–543, 2003.

[GMM11] L. J. Guibas, Q. Mérigot, and D. Morozov. Witnessed k-distance. In *Proc. 27th Annu. ACM Sympos. Comput. Geom.* (SoCG), pages 57–64, 2011.

[GN98] R. M. Gray and D. L. Neuhoff. Quantization. *IEEE Trans. on Inf. The.*, 44(6):2325 –2383, oct 1998.

[Har99] S. Har-Peled. Constructing approximate shortest path maps in three dimensions. *SIAM J. Comput.*, 28(4):1182–1197, 1999.

[Har01] S. Har-Peled. A replacement for Voronoi diagrams of near linear size. In *Proc. 42nd Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), pages 94–103, 2001.

[Har06] S. Har-Peled. Coresets for discrete integration and clustering. In *Proc. 26th Conf. Found. Soft. Tech. Theoret. Comput. Sci.* (FSTTCS), pages 33–44, 2006.

[Har11] S. Har-Peled. *Geometric Approximation Algorithms*, volume 173 of *Mathematical Surveys and Monographs*. Amer. Math. Soc., 2011.

[HIM12] S. Har-Peled, P. Indyk, and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. *Theory Comput.*, 8:321–350, 2012. Special issue in honor of Rajeev Motwani.

[HK11] S. Har-Peled and N. Kumar. Approximate nearest neighbor search for low dimensional queries. In *Proc. 22nd ACM-SIAM Sympos. Discrete Algs.* (SODA), pages 854–867, 2011.

[HK12] S. Har-Peled and N. Kumar. Down the rabbit hole: Robust proximity search in sublinear space. In *Proc. 53rd Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), pages 430–439, 2012.

[HK13a] S. Har-Peled and N. Kumar. Approximate nearest neighbor search for low dimensional queries. *SIAM J. Comput.*, 42(1):138–159, 2013.

[HK13b] S. Har-Peled and N. Kumar. Approximating minimization diagrams and generalized proximity search. In *Proc. 54th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), pages 717–726, 2013.

[HM06] S. Har-Peled and M. Mendel. Fast construction of nets in low dimensional metrics, and their applications. *SIAM J. Comput.*, 35(5):1148–1184, 2006.

[HS03] G. R. Hjaltason and H. Samet. Index-driven similarity search in metric spaces (survey article). *ACM Trans. Database Syst.*, 28(4):517–580, December 2003.

[IM98] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. 30th Annu. ACM Sympos. Theory Comput.* (STOC), pages 604–613, 1998.

[IN07] P. Indyk and A. Naor. Nearest neighbor preserving embeddings. *ACM Trans. Algo.*, 3:1–12, 2007.

[JL84] W. B. Johnson and J. Lindenstrauss. Extensions of lipschitz mapping into hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

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

[KL05] R. Krauthgamer and J. R. Lee. The black-box complexity of nearest-neighbor search. *Theo. Comp. Sci.*, 348(2-3):262–276, 2005.

[KL06] R. Krauthgamer and J. R. Lee. Algorithms on negatively curved spaces. In *Proc. 47th Annu. IEEE Sympos. Found. Comput. Sci.* (FOCS), pages 119–132, 2006.

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

[Knu98] D. E. Knuth. *The art of computer programming, volume 3: (2nd ed.) sorting and searching.* Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1998.

[KOR00] E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM J. Comput.*, 2(30):457–474, 2000.

[KR02] D. R. Karger and M. Ruhl. Finding nearest neighbors in growth-restricted metrics. In *Proc. 34th Annu. ACM Sympos. Theory Comput.* (STOC), pages 741–750, 2002.

[LBG80] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *IEEE Trans. on Commun.*, 28(1):84 – 95, jan 1980.

[Mat92] J. Matoušek. Efficient partition trees. *Discrete Comput. Geom.*, 8:315–334, 1992.

[Mat02] J. Matoušek. *Lectures on Discrete Geometry.* Springer, 2002.

[Mei93] S. Meiser. Point location in arrangements of hyperplanes. *Inform. Comput.*, 106:286–303, 1993.

[MS94] T. Martinetz and K. Schulten. Topology representing networks. *Neural Netw.*, 7(3):507–522, March 1994.

[Rup95] J. Ruppert. A Delaunay refinement algorithm for quality 2-dimensional mesh generation. *J. Algorithms*, 18(3):548–585, 1995.

[SA95] M. Sharir and P. K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications.* Cambridge University Press, New York, 1995.

[Sil86] B.W. Silverman. *Density estimation for statistics and data analysis.* Monographs on statistics and applied probability. Chapman and Hall, 1986.

[SSS02] Y. Sabharwal, N. Sharma, and S. Sen. Improved reductions of nearest neighbors search to plebs with applications to linear-sized approximate Voronoi decopositions. In *Proc. 22nd Conf. Found. Soft. Tech. Theoret. Comput. Sci.* (FSTTCS), pages 311–323, 2002.

[SSS06] Y. Sabharwal, N. Sharma, and S. Sen. Nearest neighbors search using point location in balls with applications to approximate voronoi decompositions. *J. Comput. Sys. Sci.*, 72(6):955–977, 2006.

[Tal04] K. Talwar. Bypassing the embedding: algorithms for low dimensional metrics. In *Proc. 36th Annu. ACM Sympos. Theory Comput.* (STOC), pages 281–290, 2004.

[Ten98] J. B. Tenenbaum. Mapping a manifold of perceptual observations. *Adv. Neur. Inf. Proc. Sys. 10*, pages 682–688, 1998.

[WS06] K. Q. Weinberger and L. K. Saul. Unsupervised learning of image manifolds by semidefinite programming. *Int. J. Comput. Vision*, 70(1):77–90, October 2006.

[Yia93] P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proc. 4th ACM-SIAM Sympos. Discrete Algs.* (SODA), pages 311–321, 1993.