Approximating Minimization Diagrams and Generalized Proximity Search*

Sariel Har-Peled† Nirman Kumar‡

August 19, 2016

Abstract

We investigate the classes of functions whose minimization diagrams can be approximated efficiently in $\mathbb{R}^d$. We present a general framework and a data-structure that can be used to approximate the minimization diagram of such functions. The resulting data-structure has near linear size and can answer queries in logarithmic time. Applications include approximating the Voronoi diagram of multiplicatively weighted points, but the new technique also works for more general distance functions. For example, we get such data-structures for metrics induced by convex bodies, and the nearest furthest-neighbor distance to a set of point sets. Interestingly, our framework also works for distance functions that do not obey the triangle inequality. For many of these functions no near linear size approximation was known before.

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 from 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 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,

*Work on this paper was partially supported by NSF AF awards CCF-0915984, CCF-1421231, and CCF-1217462. A preliminary version of this paper appeared in FOCS 2013 [HK13].

†Department of Computer Science; University of Illinois; 201 N. Goodwin Avenue; Urbana, IL, 61801, USA; sariel@illinois.edu; http://sarielhp.org/.

‡Department of Computer Science; University of Illinois; 201 N. Goodwin Avenue; Urbana, IL, 61801, USA; nkumar5@illinois.edu; http://www.cs.uiuc.edu/~nkumar5/.

such that given a query point $\mathsf{q} \in \mathcal{X}$, one can find (quickly) the point $\mathsf{n}_\mathsf{q} \in \mathsf{P}$ 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.

Exact nearest-neighbor search. The exact nearest-neighbor search problem has a naive linear time algorithm without any preprocessing. However, by doing some nontrivial preprocessing, one can achieve a sub-linear query time. In $\mathbb{R}^d$, this is facilitated by answering point location queries using a Voronoi diagram [BCKO08]. However, this approach is only suitable for low dimensions, as the complexity of the Voronoi diagram is $\Theta\left(n^{\lceil d/2 \rceil}\right)$ in the worst case. Specifically, Clarkson [Cla88] showed a data-structure with $O(\log n)$ query time, and $O\left(n^{\lceil d/2 \rceil + \delta}\right)$ space, where $\delta > 0$ is a prespecified constant (the $O(\cdot)$ notation here, hides constants that are exponential in the dimension). One can trade-off the space used with the query time [AM93]. Meiser [Mei93] provided a data-structure with query time $O(d^5 \log n)$ (which has polynomial dependency on the dimension), where the space used is $O\left(n^{d+\delta}\right)$. These solutions are impractical even for data-sets of moderate size if the dimension is larger than two.

Approximate nearest-neighbor. In typical applications however, it is usually sufficient to return an ***approximate nearest-neighbor*** (**ANN**). Given an $\varepsilon > 0$, a $(1+\varepsilon)$-ANN, to a query point q, is a point $y \in \mathsf{P}$, such that

$$\|\mathsf{q} - y\| \leq (1+\varepsilon)\,\|\mathsf{q} - \mathsf{n}_\mathsf{q}\|\,,$$

where $\mathsf{n}_\mathsf{q} \in \mathsf{P}$ is the nearest-neighbor to q in P. Considerable amount of work was done on this problem, see [Cla06] and references therein.

Indyk and Motwani showed that ANN queries can be reduced to a small number of near-neighbor queries [IM98, HIM12]. For high dimensional Euclidean spaces, they used locality sensitive hashing to solve the near-neighbor problem and provided a data-structure that answers ANN queries in time (roughly) $\widetilde{O}\left(n^{1/(1+\varepsilon)}\right)$ with preprocessing time and space $\widetilde{O}\left(n^{1+1/(1+\varepsilon)}\right)$; here the $\widetilde{O}(\cdot)$ hides terms polynomial in $\log n$ and $1/\varepsilon$. This was improved to $\widetilde{O}\left(n^{1/(1+\varepsilon)^2}\right)$ query time, and preprocessing time and space $\widetilde{O}\left(n^{1+1/(1+\varepsilon)^2}\right)$ [AI08]. These bounds are near optimal [MNP06].

Low dimensional ANN search. In low dimensions (i.e., $\mathbb{R}^d$ for constant, small d, say $d \leq 20$), one can use linear space (independent of ε) and get ANN query time $O(\log n + 1/\varepsilon^{d-1})$ [AMN+98, Har11]. The trade-off for this logarithmic query time is of course an exponential dependence on d. Interestingly, for this data-structure, the approximation parameter ε need not be prespecified during the construction; one can provide it during the query. An alternative approach is to use Approximate Voronoi Diagrams (AVD), introduced by Har-Peled [Har01], which is a partition of space into regions, of near linear total complexity, typically with a representative point for each region that is an ANN for any point in the region. In particular, Har-Peled showed that there is such a decomposition of size $O\left((n/\varepsilon^d)\log^2 n\right)$, such

that ANN queries can be answered in $O(\log n)$ time. Arya and Malamatos [AM02] showed how to build AVD's of linear complexity (i.e., $O(n/\varepsilon^d)$). Their construction uses Well-Separated Pairs Decomposition [CK95]. Further trade-offs between query time and space for AVD's were studied by Arya *et al.* [AMM09].

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 $f_{\mathsf{p}}(\mathsf{q}) = \omega_{\mathsf{p}} \|\mathsf{q} - \mathsf{p}\|$. 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 fast proximity and small space is 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.

1.1. Our results

We present conditions that are sufficient to allow efficient approximation of the minimization diagram of functions. Using this framework, one can quickly evaluate the lower envelope approximately for large classes of functions that arise naturally from proximity problems. Our data-structure can be constructed in near linear time, uses near linear space, and answers proximity queries in logarithmic time (for constant dimension d). Our framework is quite general and should be applicable to many distance functions, and in particular we present the following specific cases where the new data-structure can be used:

(A) **Multiplicative Voronoi diagrams.** Given a set of points P, where the ith point p_i has associated weight $w_i > 0$, for $i = 1, \ldots, n$, consider the functions $f_i(\mathsf{q}) = w_i \|\mathsf{q} - \mathsf{p}_i\|$. The minimization diagram for this set of functions, corresponds to the multiplicative weighted Voronoi diagram of the points. The approach of Arya and Malamatos [AM02] to construct AVD's using WSPD's, fails

for this problem, as that construction relies on the triangle inequality that the regular Euclidean distance possesses, but which does not hold in this case.
We provide a near linear space AVD construction for this case. We are unaware of any previous results on AVD for multiplicative weighted Voronoi diagrams.

(B) **Minkowski norms of fat convex bodies.** Given a bounded symmetric convex body C centered at the origin, it defines a natural metric; that is, for points u and v their distance, as induced by C, denoted by $\|\mathsf{u}-\mathsf{v}\|_C$, is the minimum x such that $xC+\mathsf{u}$ contains v, where $xC = \left\{ x\mathsf{p} \;\middle|\; \mathsf{p} \in C \right\}$ is the scaling of C by x. So, given a set of n data points $\mathsf{P} = \{\mathsf{p}_1, \ldots, \mathsf{p}_n\}$, and n centrally symmetric and fat convex bodies $C_1, \ldots, C_n$, we define $f_i(\mathsf{q}) = \|\mathsf{p}_i - \mathsf{q}\|_{C_i}$, for $i = 1, \ldots, n$. Since each point induces a distance by a different convex body, this collection no longer defines a metric, and this makes the problem significantly more challenging. In particular, existing techniques for AVD and ANN cannot be readily applied. If the bodies are all fat and of the same size, then intuitively, since the distance functions induced by them, are all "close" to a scaled version of the Euclidean metric, it should be possible to approximate their minimization diagram, as shown for multiplicative weighted Voronoi diagrams. Note however, that we do not assume that the bodies are of similar size, see Section 4.2 for more details.

(C) **Nearest furthest-neighbor.** Consider a situation where the given input is uncertain; specifically, for the ith ***uncertain*** point we are given a set of points $\mathsf{P}_i \subseteq \mathbb{R}^d$ where it might lie (the reader might consider the case where the ith point randomly chooses its location out of the points of P_i). There is a growing interest in how to handle such inputs, as real world measurements are fraught with uncertainty, see [DRS09, Agg09, AESZ12, AAH+13] and references therein. In particular, in the worst case, the distance of the query point q to the ith point, is the distance from q to the furthest-neighbor of q in P_i; that is, $\mathrm{F}_i(\mathsf{q}) = \max_{\mathsf{p}\in\mathsf{P}_i} \|\mathsf{q}-\mathsf{p}\|$. Thus, in the worst case, the nearest uncertain point to the query is at a distance $\mathrm{F}(\mathsf{q}) = \min_i \mathrm{F}_i(\mathsf{q})$. Using our framework we can approximate this function efficiently, using near linear space, and providing logarithmic query time. Note that surprisingly, the space requirement is independent of the original input size, and only depends on the number of uncertain points. See Section 4.3 for details.

Paper organization. In Section 2 we define our framework and prove some basic properties. Since we are trying to make our framework as inclusive as possible, its description is somewhat abstract. In Section 2.5 we summarize the results that follow from the framework. In Section 3, we describe the construction of the AVD and its associated data-structure. We describe in Section 4 some specific cases where the new AVD construction can be used. We conclude in Section 5.

2. The framework and summary of results

For the sake of simplicity of exposition, throughout the paper 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.

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 $\mathrm{d}(\mathsf{q}) \leq x \leq (1+\varepsilon)\mathrm{d}(\mathsf{q})$, where

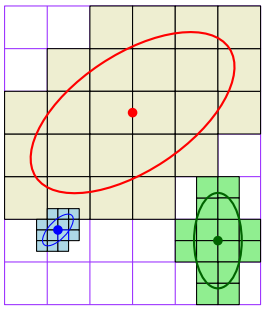
$$\mathrm{d}(\mathsf{q}) = \mathrm{d}(\mathsf{q}, \mathcal{F}) = \min_{i=1,\ldots,n} f_i(\mathsf{q}) \tag{2.1}$$

is the lower envelope of $\mathcal{F}$ at q; that is, $\mathrm{d}(\mathsf{q})$ 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 point q, if c is the cell of the AVD that contains q, then $\mathrm{d}(\mathsf{q}, \mathcal{G}_{\mathsf{c}})$ approximates $\mathrm{d}(\mathsf{q}, \mathcal{F})$ 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 $\mathrm{d}(\mathsf{q}, \mathcal{G}_{\mathsf{c}})$ explicitly. Usually, such an AVD is constructed using compressed quadtrees and yield $O(\log n)$ ANN query time, see [Har11] for details.

2.2. Informal description of the technique

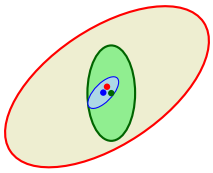
Consider n points in the plane $\mathsf{p}_1, \ldots, \mathsf{p}_n$, where the "distance" from the ith point to a query q, is the minimum scaling of an ellipse $\mathcal{E}_i$ (centered at p_i), till it covers q, and let f_i denote this distance function. Assume that these ellipses are fat, i.e., for 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 $\mathrm{d}(\mathsf{q}) = \min_{i=1,\ldots,n} f_i(\mathsf{q})$ be the minimization diagram of these functions.



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 ellipses (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 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 $\bigcup_{k=1}^{n} (\mathsf{p}_k + r_j \mathcal{E}_k)$ contain the query point q in their union, at distance r_{j+1}. Assume the set of these connected components 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.

2.2.1. 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 have 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 to 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 careful to avoid accumulation in the error as we replace clusters by sketches.

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 $\mathsf{d}(\mathsf{q}, \mathsf{P}) = \min_{x \in \mathsf{P}} \|\mathsf{q} - x\|$. For a number $\ell > 0$, the ***grid*** of side-length ℓ, denoted by G_ℓ, is the natural tiling of $\mathbb{R}^d$, with cubes

of side-length ℓ (i.e., with a vertex at the origin, and axis parallel sides). A cube $\square$ is ***canonical*** if it belongs to G_ℓ, ℓ is a power of 2, and $\square \subseteq [0,1]^d$. Informally, a canonical cube (or cell) is a region that might correspond to a cell in a quadtree having the unit cube as the root region.

Definition 2.1. 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 $X \subseteq \cup\mathsf{G}_{\approx r}(X) \subseteq X \oplus \mathsf{B}(0,r)$, where $\oplus$ denotes the Minkowski sum, and $\mathsf{B}(0,r)$ is the ball of radius r centered at the origin.

Definition 2.2. 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 $\mathcal{F}_{\preceq \ell, \approx r} = \cup_{f \in \mathcal{F}} \mathsf{G}_{\approx r}(f_{\preceq \ell})$.

Definition 2.3. 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. (2.1).

Example 2.4. 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 $f_i(\mathsf{q}) = \|\mathsf{q} - \mathsf{p}_i\|$. 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 2.5. *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 2.6. 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 2.7. A family of functions $\mathfrak{F}$ from $\mathbb{R}^d$ to $\mathbb{R}$, is *well behaved*, if for any m functions $\{f_1, \ldots, f_m\} \subseteq \mathfrak{F}$, their lower envelope has the following properties:

(A) The projection of the lower envelope into $\mathbb{R}^d$ can be decomposed into constant complexity cells. Formally, each such cell is a semialgebraic set that can be described by a constant number of algebraic inequalities, see Basu *et al.* [BPR06].
(B) One can build a data-structure that answer point-location queries in the resulting set of cells in $O(\log m)$ time.
(C) The number of cells in this decomposition, the time to compute this decomposition, and the space required to store the data-structure are all bounded by $m^{\gamma \cdot d}$.

The constant γ is the ***wellness constant*** associated with $\mathfrak{F}$.

2.3.1. 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 2.8 ($\ell_{\mathrm{con}}(\mathcal{F})$). 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 2.6.

Remark. The requirement above that every sublevel set of a function is connected (on its own) in the above definitions can be relaxed. Since we currently we have no application that uses this extra flexibility, we opted to stick with the cleaner and somewhat more restrictive definition – our results should also hold in this more general settings.

Definition 2.10. Given a set of functions $\mathcal{G}$ and $\delta \geq 0, y_0 \geq 0$, a (δ, 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 verify 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 $\mathcal{G}$. Clearly, the set $\mathcal{G}$ is a sketch of itself.

2.4. Conditions on the functions

A family of functions $\mathfrak{F}$ from $\mathbb{R}^d$ to $\mathbb{R}^+$, is ***dependable***, if there are absolute constants, $\zeta > 0$ (***growth constant***), and a positive integer c_{sk} (***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, $f_{\preceq y} \oplus \mathsf{B}(0, \varepsilon \lambda_f(y)) \subseteq f_{\preceq (1+\varepsilon)y}$, where $\mathsf{B}(\mathsf{u}, r)$ 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 $|\mathcal{H}| = O(1/\delta^{\mathsf{c}_{\mathrm{sk}}})$ and $y_0 = O\big(\ell_{\mathrm{con}}(\mathcal{G}) \cdot (|\mathcal{G}|/\delta)^{\mathsf{c}_{\mathrm{sk}}}\big)$ 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 constant time.
(C2) For any $y \geq 0, r > 0$, the set of grid cells approximating the sublevel set $f_{\preceq y}$ of f, that is $f_{\preceq y, \approx r} = \mathsf{G}_{\approx r}(f_{\preceq y})$ (see Definition 2.1), 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 constant time.

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

Remark. (A) In the following, the set of functions $\mathcal{F} = \{f_1, \ldots, f_n\}$ is taken from an implicit family $\mathfrak{F}$. Thus, the growth and sketch constants used are associated with $\mathfrak{F}$ and do not depend on n or $\mathcal{F}$. Henceforth, the terms family, set, or class of functions refers to $\mathcal{F}$, while the underlying family $\mathfrak{F}$ would be only be implicitly mentioned.

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

Remark 2.12. Condition (C2) is only used for $r = \Omega(\varepsilon\lambda_f(y))$. That is, the grid used 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 $O(1/\varepsilon^d)$. The value of r being used can be somewhat imprecise as long as it is sufficiently small.

2.4.1. 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, and are delegated to Appendix B_{p30}.

(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 B.1_{p30}.

(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 2.8 and Observation B.2_{p30}.

(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 $\mathsf{uv} \subseteq f_{\preceq (1+\zeta/2)y}$, where uv denotes the segment joining u to v, see Lemma B.3_{p31}.

(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 B.5_{p31}.

(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 $\mathrm{d}(\mathsf{q}, \mathcal{F}) \geq y_0$. Then we have that $\mathrm{d}(\mathsf{q}, \mathcal{G}) \leq (1+\delta)\mathrm{d}(\mathsf{q}, \mathcal{F})$, see Lemma B.6_{p31}.

Remark 2.13 (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. As another example, the minimum of a function $f \in \mathcal{F}$ can be computed efficiently (see (L1)).

2.5. Summary of results

The following is the main result of the paper, see Section 3 for details.

Theorem 2.14. *Let $\mathcal{F}$ be a set of n functions in $\mathbb{R}^d$ that complies with the assumptions of Section 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 3.4.1 for details.

Corollary 2.15. *Consider the settings of Theorem 2.14 with the additional condition that $\mathcal{F}$ is well-behaved, see Definition 2.7, with wellness constant γ. Then, one can build a data-structure to answer ANN for this set of functions, with the following guarantees:*
(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 $S = O\big(n\varepsilon^{-d-1-\mathsf{c}_{\mathrm{sk}}}\log^{\mathsf{c}_{\mathrm{sk}}+1} n + n\varepsilon^{-\gamma d\mathsf{c}_{\mathrm{sk}}}\big)$.
This also computes an AVD of $\mathcal{F}$ of complexity $O(S)$ – a space decomposition into cells, such that each cell has a single function of $\mathcal{F}$ associated with it, and for any point in this cell, this function is a $(1+\varepsilon)$-ANN among the functions of $\mathcal{F}$.

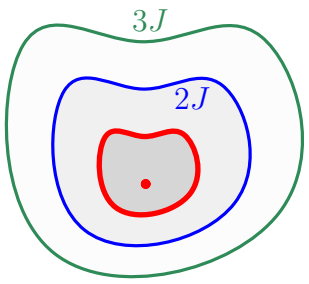
2.5.1. Distance functions for which the framework applies

2.5.1.1. Multiplicative distance functions with additive offsets. For $i = 1, \ldots, n$, let p_i be a point 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 $f_i(\mathsf{q}) = w_i \|\mathsf{q} - \mathsf{p}_i\| + \alpha_i$. Section 4.1 shows that these distance functions satisfy the conditions of Section 2.4, implying the following result.

Theorem 2.16. *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 $f_i(\mathsf{q}) = w_i \|\mathsf{q} - \mathsf{p}_i\| + \alpha_i$. One can compute a $(1+\varepsilon)$-AVD for these distance functions, with near linear space complexity, and logarithmic query time. See Theorem 2.14$_{p9}$ and Corollary 2.15 for the precise bounds, with the sketch and wellness constants $\mathsf{c}_{\mathrm{sk}} = d+1$ and $\gamma = 1$.*

2.5.1.2. 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). Furthermore, 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 4.5$_{\mathrm{p22}}$ 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 2.4 implying the following result, see Section 4.2 for details.

Theorem 2.17. *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 2.14$_{p9}$ for the precise bounds, with the sketch constant being $\mathsf{c}_{\mathrm{sk}} = d+1$.*

2.5.1.3. 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 $\mathsf{f}_{\mathsf{S}}(\mathsf{q}) = \max_{\mathsf{s}\in\mathsf{S}} \|\mathsf{q}-\mathsf{s}\|$; 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 $i = 1, \ldots, n$. The function $\mathsf{f}(\cdot)$ is the ***nearest furthest-neighbor*** distance, and the task at hand is to approximate it.

The distance functions $\mathsf{f}_1, \ldots, \mathsf{f}_n$ satisfy the conditions of the framework, see Section 4.3, implying the following.

Theorem 2.18. *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, with the following guarantees:*

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

(B) The preprocessing time and space used is $O\Big(n(\varepsilon^{-1}\log n)^{O(d)}\Big)$.

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

3. Constructing the AVD

The input is a set $\mathcal{F}$ of n functions, satisfying the conditions of Section 2.4, and a number $\varepsilon \in (0,1)$. We preprocess $\mathcal{F}$, such that given a query point q, one can compute a $f \in \mathcal{F}$, where $\mathrm{d}(\mathsf{q},\mathcal{F}) \le \mathrm{d}(\mathsf{q},f) \le (1+\varepsilon)\mathrm{d}(\mathsf{q},\mathcal{F})$.

3.1. Building blocks

3.1.1. Near-neighbor

Given a set of functions $\mathcal{G}$, a real number $\alpha \ge 0$, 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 $\mathrm{d}(\mathsf{q},\mathcal{G}) \le \alpha$, and no if $\mathrm{d}(\mathsf{q},\mathcal{G}) > (1+\varepsilon)\alpha$. It can return either answer if $\mathrm{d}(\mathsf{q},\mathcal{G}) \in (\alpha, (1+\varepsilon)\alpha]$. If it returns yes, then it also returns a function $f \in \mathcal{G}$ such that $\mathrm{d}(\mathsf{q},f) \le (1+\varepsilon)\alpha$. The query time of this data-structure is denoted by $T_{\le}(m)$, where $m = |\mathcal{G}|$.

Lemma 3.1. *Given a set $\mathcal{G} \subseteq \mathcal{F}$ of m functions, $\alpha > 0$ 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 $T_{\le}(m) = O(\log(m/\varepsilon))$.*

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 $(\mathrm{P2})_{\mathrm{p8}}$. 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 $(\mathrm{P2})_{\mathrm{p8}}$ 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, we can decide if a point is covered by some cube of $\mathcal{C}$, by performing a point location query in $\mathcal{T}$.

By Remark 2.12, $|\mathsf{G}_{\approx r_f}(f_{\preceq\alpha})| = O(\varepsilon^{-d})$. As such, the total number of canonical cubes in $\mathcal{C}$ is $O\bigl(m/\varepsilon^d\bigr)$, and the compressed quadtree for storing them can be computed in $O(m\varepsilon^{-d}\log(m/\varepsilon))$ 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. ∎

3.1.2. 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 $\mathrm{d}(\mathsf{q},\mathcal{G}) \in \Bigl[\alpha,\beta\Bigr]$, 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 3.2. *Given a set $\mathcal{G}$ of m functions, an interval $[\alpha,\beta]$, and an approximation parameter $0 < \tau \leq 4$, such that $\log(4\beta/\alpha) = O(m/\tau)$, one can construct an interval data-structure of size $O\bigl(m\tau^{-d-1}\log\frac{4\beta}{\alpha}\bigr)$, in time $O\bigl(m\tau^{-d-1}\log\frac{4\beta}{\alpha}\log\frac{m}{\tau}\bigr)$, such that given a query point q one can answer a $(1+\tau)$-ANN query for the distances in the interval $[\alpha,\beta]$, in time $T_{\mathrm{I}}(m,\alpha,\beta) = O\left(\log\dfrac{m\log(4\beta/\alpha)}{\tau}\right)$.*

Proof: Using Lemma 3.1, build a $(1+\tau/4)$ near-neighbor data-structure D_i for $\mathcal{G}$, for distance $r_i = (\alpha/2)(1+\tau/4)^i$, for $i = 0,\ldots,L = \bigl\lceil\log_{1+\tau/4}(4\beta/\alpha)\bigr\rceil = O(\tau^{-1}\log(4\beta/\alpha))$. Clearly, an interval query can be answered in three stages:

(A) Perform a point-location query in 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 D_L. If the answer is no then $\mathrm{d}(\mathsf{q},\mathcal{G}) > \beta$ and we are done.
(C) It must be that $\mathrm{d}(\mathsf{q},\mathcal{G}) \in \Bigl[r_i, r_{i+1}\Bigr]$ for some i. Find this i by performing a binary search on the data-structures $\mathsf{D}_0,\ldots,\mathsf{D}_L$, for the first i such that D_i returns no, but D_{i+1} returns yes. Clearly, 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 $\mathsf{D}_0,\ldots,\mathsf{D}_L$, 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

$$O(S\log L) = O\left(\frac{Lm}{\tau^d}\log L\right) = O\left(\frac{m\tau^{-1}\log(4\beta/\alpha)}{\tau^d}\log\big(m\tau^{-1}\log(4\beta/\alpha)\big)\right) = O\left(\frac{m\log(4\beta/\alpha)}{\tau^{d+1}}\log\frac{m}{\tau}\right),$$

since $\log(4\beta/\alpha) = O(m/\tau)$. Indeed, overlaying L quadtrees is equivalent to merging L sorted lists, see [Har11, Lemma 2.19]. The time to perform a point-location query in the overlayed quadtree is $O(\log S)$.■

Lemma 3.2 readily implies that if the nearest-neighbor distance lies in an interval of values of polynomial spread, then Lemma 3.2 yields the desired data-structure. To overcome this unbounded spread problem, we first argue that under our assumptions there are only linear number of intervals where interesting things happen to the distance function.

3.1.3. 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 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 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 i, 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 constant time.

Definition 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, let $\Pi[X'_i]$ denote 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 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 the key in decomposing the nearest-neighbor search on $\mathcal{F}$, into a search on smaller sets.

Observation 3.7. *If $0 \leq a \leq b$, then $\mathsf{C}(\mathcal{G}, a) \sqsubseteq \mathsf{C}(\mathcal{G}, b)$.*

For any number $\ell \geq 0$, the following lemma testifies that ℓ-clustering can be approximated quickly.

Lemma 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 the bounded growth condition $(\mathrm{P2})_{\mathrm{p8}}$ assures that all the 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 $\mathcal{C}_f = f_{\preceq \ell, \approx r_f}$, see Definition 2.2_{p7}. It is easy to verify that

$$(\mathcal{G})_{\preceq \ell} \subseteq \bigcup_{f \in \mathcal{G}} (\cup \mathcal{C}_f) \subseteq (\mathcal{G})_{\preceq (1+\varepsilon)\ell}, \tag{3.1}$$

see Definition 2.1. 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 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 $\ell_{\mathrm{con}}(C) \leq (1+\varepsilon)\ell$.

3.1.4. Computing a splitting distance

Definition 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 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 $r_C = \min_{f \in C} \min_{g \in \mathcal{G} \setminus C} \mathrm{d}(f, g)$. By Lemma 3.8, any two functions that are in distance $\leq \ell$ are in the same cluster of Ψ. This implies that $r_C > \ell$. 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 $m - m'$,

```
AprxNN( 𝒢, ϒ, q )
    // 𝒢: set of functions, ϒ = Ψ_1(𝒢, ℓ) for some value ℓ, q: query point.
    if |ϒ| ≤ log n then
        return d(q, 𝒢) = min_{f∈𝒢} d(q, f)
    x ← compute a splitting distance of ϒ, see Lemma 3.10.

    // Perform an interval ANN query on [x/8, Nx], see Lemma 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 3.1.
        Find cluster C ∈ Ψ_1(𝒢, x/4), such that f ∈ C, see Lemma 3.8.
        return AprxNN( C, ϒ[C], q )

    // Must be that d(q, 𝒢) > Nx
    return AprxNN( 𝒢, Ψ_1(𝒢, x), q )                                   (⋆)
```

Figure 3.1: Search algorithm: 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., $\ell = 0$).

while the number of components 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 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. ∎

3.2. The search procedure

3.2.1. An initial "naive" implementation

The search procedure is presented in Figure 3.1.

Lemma 3.11. $\mathbf{AprxNN}(\mathcal{G}, \Upsilon, \mathsf{q})$ *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* **AprxNN** *is* $\mathsf{h} = O(\log n)$, *where* $n = |\mathcal{G}|$.

Proof: The proof is by induction on the size of Υ. If $|\Upsilon| \leq \log n$, 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. 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 **AprxNN** recurses on this cluster. Now, by induction, the returned ANN is correct.
Since x is a splitting distance for Υ, see Definition 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 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. ∎

Remark 3.12. Case (B) in the proof of Lemma 3.11 does not introduce any error, because we are still conceptually working with all the functions. For a more efficient implementation, we would have to sketch the sets of functions, and that would introduce an error in the search which can accumulate. See Corollary 3.13 below.

3.2.2. But where is the beef? Modifying AprxNN to provide fast query time

The reader might wonder how we are going to get an efficient algorithm out of **AprxNN**, as in the case that Υ has a small number of clusters (i.e., $\log n$), 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), and all the functions in the final set of at most $\log n$ clusters, have also been sketched to form a sketch of size $O(1/\varepsilon^{\mathsf{c}_{\mathrm{sk}}})$. Initially, since all the clusters are singletons, this is clearly true. The merge of clusters merge happens during step $(\star)$ of **AprxNN**, see Figure 3.1. In future iterations one can 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.

3.2.3. Sketching a sketch

To improve the efficiency of the preprocessing for **AprxNN**, 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 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 $(\mathrm{P3})_{\mathrm{p8}}$. Now assuming $\delta = O(\varepsilon/\log n)$, and ε is 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 distance $\geq N\ell_{\mathrm{con}}(\mathcal{G})$.

The only place in the algorithm where we need to compute the sketches, is in step $(\star)$ of **AprxNN**, see Figure 3.1. 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 serves as this cluster's sketch from this point on. In particular, the recursive calls to **AprxNN** 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 $(1+\delta)^{\mathsf{h}} \leq 1 + \varepsilon/4$.

If the number of clusters is small enough, the recursion bottoms out; here we store a $\varepsilon/8$-sketch of the set of functions in the union of the parts, in the search structure. This would reduce the sketch size to $O(1/\varepsilon^{\mathsf{c}_{\mathsf{sk}}})$. Note however, that this still does not help us as far as recursion - we must pass the larger δ-sketches in the recursive call in step $(\star)$ of **AprxNN**, see Figure 3.1.

Corollary 3.13. *The modified search procedure that uses sketches, as described above, returns a $(1+\varepsilon)$-ANN to the query point.*

Proof: The accumulated error in the NN search because of the sketching is bounded by $1 + \varepsilon/4$, by the above. Another factor of $1 + \varepsilon/4$ is introduced by the interval data-structure used by **AprxNN**, if the interval data-structure finds the desired nearest neighbor. Finally, the shrinking of the sketch at a leaf node introduces another $1 + \varepsilon/8$ factor to the error.

Now, following the proof of Lemma 3.11 implies that the error is bounded by $(1 + \varepsilon/4)(1 + \varepsilon/4)(1 + \varepsilon/8) \leq 1 + \varepsilon$. ∎

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.

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 $\mathrm{d}(X', Y')$. 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 $\mathrm{d}(\cdot, \cdot)$.

To make the presentation consistent, let $\mathrm{d}_{\approx}(X, Y)$ be the minimum x, such that $\Psi_1(X \cup Y, x)$ is connected (see Lemma 3.8$_{\mathrm{p13}}$), and x is a power of two. Computing $\mathrm{d}_{\approx}(X, Y)$ can be done by computing $\mathrm{d}_{\approx}(f, g)$ for each pair of functions separately. This in turn, can be done by first computing $\alpha = \mathrm{d}(f, g)$, and observing that x is between $\alpha/2$ and α. In particular, since x must be a power of two, there are at most 2 candidate values to consider, and which is the right one can be decided using Lemma 3.8.

So, in the above, we use $\mathrm{d}_{\approx}(\cdot, \cdot)$ instead of $\mathrm{d}(\cdot, \cdot)$, 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 $\mathrm{d}_{\approx}(X, Y)$ 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 3.14. *Consider a recursive call* $\mathbf{AprxNN}(\mathcal{G}, \Upsilon, \mathsf{q})$*, see Figure 3.1*$_{p15}$*, made during the search algorithm execution. Then there exists a node* u*, and a value* ℓ*, such that* $\mathcal{G} = \mathrm{F}(u)$ *and* $\Upsilon = \left\{ \mathrm{F}(v) \;\middle|\; v \in \mathrm{L}_{\mathcal{H}}(\ell), \text{ and } v \text{ is in the subtree of } u \right\}$*. That is, a recursive call of* **AprxNN** *corresponds to a subtree of* $\mathcal{H}$*.*

Of course, not all possible subtrees are candidates to be such a recursive call. In particular, **AprxNN** can now be interpreted as working on a subtree T of $\mathcal{H}$.

Specifically, the set of all functions stored in a tree T is the set $\mathrm{F}(T) = \bigcup_{u \in T} \mathrm{F}(T)$, which is equal to $\mathrm{F}(\mathsf{root}(T))$. On the other hand, this set of functions is also $\mathrm{F}(T) = \bigcup_{u \text{ is a leaf of } T} \mathrm{F}(T)$ (which is the most refined partition associated with this tree). Namely, a tree (informally) encodes different partitions that are all refinements of $\mathrm{F}(T)$, that (somewhat imprecisely) corresponds to different resolutions. Our task is to find the right resolution/partition where we can readily answer the ANN query, and this is done by doing a divide-and-conquer on the tree T. In the process, the set of functions under consideration would shrink (i.e., as the root of the current tree changes to a subtree's root), and the size of the set of possible partitions would become smaller, as the number of leafs in the subtree similarly decreases. As such, either we would find the ANN during this process, or we would be left with a single node, where the ANN query can be answered directly using the sketch.

Reinterpreting **AprxNN** (see Figure 3.1) as working on subtrees, results in the following algorithm:

(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.
Note that the number of clusters associated with the finest partition induced by T is exactly the number of leafs of T. As such, the splitting distance (see Lemma 3.10), is exactly such a resolution that shrinks the numbers of leafs in a subtree, by a constant factor.

(C) Using interval data-structure determine if the distance $\mathrm{d}(\mathsf{q}, \mathrm{F}(T))$ is in the range $[x/8, Nx]$. If so, we found the desired ANN.

(D) If $\mathrm{d}(\mathsf{q}, \mathrm{F}(T)) > Nx$ then continue recursively on portion of T above $\mathrm{L}_T(x)$.

(E) If $\mathrm{d}(\mathsf{q}, \mathrm{F}(T)) < x/8$ 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, **AprxNN** breaks T into subtrees, and continues the search recursively on one of the subtrees. Significantly, every such subtree has size which 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 **AprxNN**. 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 2.13$_{p9}$. Computing the splitting distance, using Lemma 3.10, takes $U_2 = O(M(\log M + 1/\delta^{\mathsf{c}_{\mathrm{sk}}})) = O(m \log n \delta^{-\mathsf{c}_{\mathrm{sk}}} + m\delta^{-2\mathsf{c}_{\mathrm{sk}}})$

time. Computing the interval data-structure of Lemma 3.2 takes

$$U_3 = O\big(M\varepsilon^{-d-1}\log n \log M\big) = O\big(m\delta^{-\mathsf{c}_{\mathrm{sk}}}\varepsilon^{-d-1}\log^2 n\big)$$

time, and requires $S_1 = O\big(M\varepsilon^{-d-1}\log n\big) = O\big(m\delta^{-\mathsf{c}_{\mathrm{sk}}}\varepsilon^{-d-1}\log n\big)$ space. here, the required condition on β, α and m in Lemma 3.2 holds as **AprxNN** bottoms out when the set of functions is of size $O(\log n)$. 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 than the original tree). Finally, we need to compute the δ-sketches for the clusters sent to the appropriate recursive calls, and this takes $U_4 = O(M/\delta^{\mathsf{c}_{\mathrm{sk}}}) = O(m\delta^{-2\mathsf{c}_{\mathrm{sk}}})$ time, by Remark 2.13$_{\mathrm{p9}}$.

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

$$\begin{aligned} W &= O\Big((U_1 + U_2 + U_3 + U_4)/m\Big) = O\big((\varepsilon\delta)^{-\mathsf{c}_{\mathrm{sk}}} + \delta^{-\mathsf{c}_{\mathrm{sk}}}\log n + \delta^{-2\mathsf{c}_{\mathrm{sk}}} + \varepsilon^{-d-1}\delta^{-\mathsf{c}_{\mathrm{sk}}}\log^2 n + \delta^{-2\mathsf{c}_{\mathrm{sk}}}\big) \\ &= O\Big(\varepsilon^{-\max(d+1+\mathsf{c}_{\mathrm{sk}}, 2\mathsf{c}_{\mathrm{sk}})}\log^{\max(\mathsf{c}_{\mathrm{sk}}+2, 2\mathsf{c}_{\mathrm{sk}})} n\Big). \end{aligned}$$

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)$_{\mathrm{p8}}$. 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 $O\big(n\varepsilon^{-2\mathsf{c}_{\mathrm{sk}}}\log^{2\mathsf{c}_{\mathrm{sk}}+1} n\big)$.

As for the space, we have by the same argument, that each edge requires $O(\log n \cdot (S_1/m)) = O\big(\varepsilon^{-d-1-\mathsf{c}_{\mathrm{sk}}}\log^{\mathsf{c}_{\mathrm{sk}}+1} n\big)$. Moreover, the space used to store the $\varepsilon/8$-sketches at the leaf nodes is $O(n\varepsilon^{-\mathsf{c}_{\mathrm{sk}}})$. 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.

3.4. The result

Proof of Theorem 2.14$_{p9}$: 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 **AprxNN** 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 $\mathcal{Q}_1$ has higher priority than a compressed quadtree $\mathcal{Q}_2$, if $\mathcal{Q}_1$ is queried after $\mathcal{Q}_2$, 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 find the leaf that contains the query point, we scan the sketch associated with this cell, and return the computed nearest-neighbor. ∎

3.4.1. The AVD construction

Proof of Corollary 2.15$_{p10}$: We build the data-structure of Theorem 2.14, 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 2.7. 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 $O(n\varepsilon^{-d-1-\mathsf{c}_{\mathrm{sk}}} \log^{\mathsf{c}_{\mathrm{sk}}+1} n + n\varepsilon^{-\gamma d\mathsf{c}_{\mathrm{sk}}})$, where the first part comes from Theorem 2.14, 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 defined over the decomposition into cells of the projection of the lower envelope of the sketch onto $\mathbb{R}^d$. One need to clip each such cell to the region associated with the node/leaf (of the compressed quadtree) that uses it, but since each such cell has constant descriptive complexity as a semi-algebraic set, this clipping does not change its asymptotic complexity. ∎

4. Applications

We present some concrete classes of functions that satisfy our framework, and for which we construct AVD's efficiently.

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 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 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 $(f_i)_{\preceq y} = \mathsf{B}\Big(\mathsf{p}_i, \frac{y-\alpha_i}{w_i}\Big)$ for $y \geq \alpha_i$. Checking conditions $(\mathrm{C1})_{\mathrm{p8}}$ and $(\mathrm{C2})_{\mathrm{p8}}$ is trivial. As for $(\mathrm{C3})_{\mathrm{p8}}$ we have the following easy lemma,

Lemma 4.1. *For any i, j, such that $1 \leq i, j \leq n$, 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 $\mathsf{q} \in \mathsf{B}\Big(\mathsf{p}_i, \frac{y-\alpha_i}{w_i}\Big)$; that is, we have $(f_i)_{\preceq y} = \mathsf{B}\Big(\mathsf{p}_i, \frac{y-\alpha_i}{w_i}\Big)$, and $(f_j)_{\preceq y} = \mathsf{B}\Big(\mathsf{p}_j, \frac{y-\alpha_j}{w_j}\Big)$.

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 $\max\Big(\alpha_i, \alpha_j, \|\mathsf{p}_i - \mathsf{p}_j\| \frac{w_i w_j}{w_i + w_j} + \frac{\alpha_i w_j + \alpha_j w_i}{w_i + w_j}\Big) = \max(\alpha_i, \alpha_j,)$, as desired.

If $\mathsf{p}_i \neq \mathsf{p}_j$ the sublevel sets intersect for the first time when the balls $\mathsf{B}\Big(\mathsf{p}_i, \frac{y-\alpha_i}{w_i}\Big)$ and $\mathsf{B}\Big(\mathsf{p}_j, \frac{y-\alpha_j}{w_j}\Big)$ 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 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 $(f_i)_{\preceq y} = \mathsf{B}\Big(\mathsf{p}_i, \frac{y-\alpha_i}{w_i}\Big)$, and $(f_j)_{\preceq y} = \mathsf{B}\Big(\mathsf{p}_j, \frac{y-\alpha_j}{w_j}\Big)$. 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 $\mathsf{u} \in \mathsf{B}\Big(\mathsf{p}_j, \frac{y-\alpha_j}{w_j}\Big)$ 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 $\mathsf{B}\Big(\mathsf{p}_j, \frac{y-\alpha_j}{w_j}\Big) \subseteq \mathsf{B}\Big(\mathsf{p}_i, \frac{(1+\delta)y-\alpha_i}{w_i}\Big)$. 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 $\mathsf{B}\Big(\mathsf{p}_j, \frac{y-\alpha_j}{w_j}\Big)$ 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 $\mathsf{B}\Big(\mathsf{p}_i, \frac{(1+\delta)y-\alpha_i}{w_i}\Big)$ 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 $(\mathrm{P1})_{\mathrm{p8}}$ and bounded growth $(\mathrm{P2})_{\mathrm{p8}}$ hold for the set of functions $\mathcal{F}$ (for $(\mathrm{P2})_{\mathrm{p8}}$ 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 $(\mathrm{P3})_{\mathrm{p8}}$.

Lemma 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 $\ell \geq \max_{i: f_i \in \mathcal{G}} \alpha_i$, 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 $\mathsf{B}\Big(\mathsf{p}_i, \frac{\ell-\alpha_i}{w_i}\Big)$, 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 $\mathsf{B}\Big(\mathsf{p}_{i_r}, \frac{\ell-\alpha_{i_r}}{w_{i_r}}\Big) \cap \mathsf{B}\Big(\mathsf{p}_{i_{r+1}}, \frac{\ell-\alpha_{i_{r+1}}}{w_{i_{r+1}}}\Big) \neq \emptyset$, for $r = 1, \ldots, k-1$. By Lemma 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, $\|\mathsf{p}_{i_r} - \mathsf{p}_{i_{r+1}}\| \leq \ell \cdot \left(\frac{1}{w_{i_r}} + \frac{1}{w_{i_{r+1}}}\right) - \left(\frac{\alpha_{i_r}}{w_{i_r}} + \frac{\alpha_{i_{r+1}}}{w_{i_{r+1}}}\right) \leq \frac{2\ell}{w_1}$, as $w_1 \leq w_i$, for $i = 1, \ldots, m$. It follows by the triangle inequality and the above, that $\|\mathsf{p}_{i_1} - \mathsf{p}_{i_k}\| \leq \sum_{r=1}^{k-1} \|\mathsf{p}_{i_r} - \mathsf{p}_{i_{r+1}}\| \leq 2(k-1)\ell/w_1 \leq 2m\ell/w_1$. 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 $y \geq y_0$,

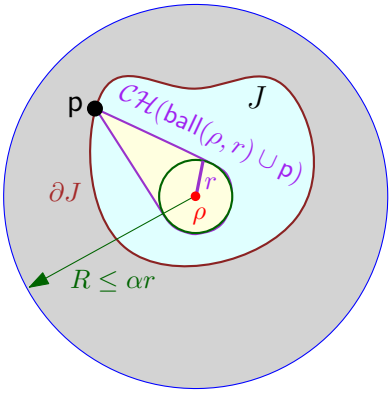


Figure 4.1: Being α-rounded fat.

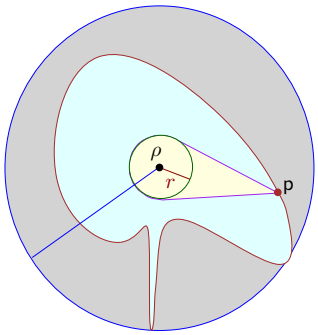


Figure 4.2: A α-fat star shaped region that is not α-rounded fat. Clearly, the gap between the rounded fatness parameter and the regular fatness parameter can be made to be arbitrarily large.

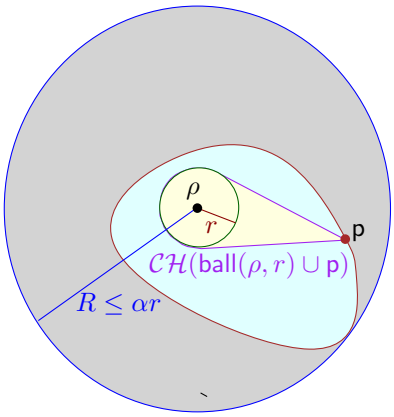


Figure 4.3: A α-fat convex body is α-rounded fat (for the same center point ρ).

$y \geq \dfrac{\|\mathsf{p}_1-\mathsf{p}_j\|+\alpha_1/w_1}{\delta/w_1} \geq \dfrac{\|\mathsf{p}_1-\mathsf{p}_j\|+\alpha_1/w_1-\alpha_j/w_j}{\delta/w_1+(1/w_1-1/w_j)} = \dfrac{\|\mathsf{p}_1-\mathsf{p}_j\|+\alpha_1/w_1-\alpha_j/w_j}{(1+\delta)/w_1-1/w_j}$, as $w_1 \leq w_j$, for $j=1,\ldots,m$. Thus, by Lemma 4.2, $\mathsf{B}\Big(\mathsf{p}_j, \frac{y-\alpha_j}{w_j}\Big) \subseteq \mathsf{B}\Big(\mathsf{p}_1, \frac{(1+\delta)y-\alpha_1}{w_1}\Big)$, for $y \geq y_0$, and therefore, by definition, $\mathcal{H}$ is a (δ, y_0)-sketch for $\mathcal{G}$. ∎

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 2.7$_{\mathrm{p7}}$, and that we can choose the wellness constant $\gamma = 1$. Thus, we get Theorem 2.16$_{\mathrm{p10}}$.

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 $J = \mathsf{B}(\mathsf{p}, 1)$ with center p, then $f_J(\mathsf{q}) = \|\mathsf{p}-\mathsf{q}\|$. 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 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 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 4.1.

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 4.2. The following useful result is easy to see, also see Figure 4.3 for an illustration.

Lemma 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 2.4. Using standard techniques, we can compute the quantities required in conditions (C1)–(C3)$_{\text{p8}}$, as well as the diameter of the sublevel sets, as $\mathsf{diam}(yJ_i) = y\mathsf{diam}(J_i)$. Also, trivially we have that condition (P1)$_{\text{p8}}$ 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)$_{\text{p8}}$, and the sketch property (P3)$_{\text{p8}}$, are also true, if the objects are also α-rounded fat, for some constant α.

Lemma 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 $yJ \oplus \mathsf{B}(0, (\varepsilon/c)\mathsf{diam}(yJ)) \subseteq (1+\varepsilon)yJ$; that is, $(f_J)_{\preceq y} \oplus \mathsf{B}\Big(0, (\varepsilon/c)\mathsf{diam}\Big((f_J)_{\preceq y}\Big)\Big) \subseteq (f_J)_{\preceq (1+\varepsilon)y}$.*

Proof: Since $(f_J)_{\preceq y} = yJ$ it is enough to show that $yJ \oplus \mathsf{B}\big(0, (\varepsilon/c)\mathsf{diam}(yJ)\big) \subseteq (1+\varepsilon)yJ$. Let r be the radius guaranteed by Definition 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 $\mathsf{p} + \mathsf{B}(0, (\varepsilon/c)\mathsf{diam}(yJ)) \subseteq (1+\varepsilon)yJ$. It is sufficient to show that $\mathsf{B}(\mathsf{p}, (2\varepsilon y\alpha r/c)) \subseteq (1+\varepsilon)yJ$, for $\mathsf{p} \in \partial yJ$. Clearly $\mathsf{p}' = (1+\varepsilon)\mathsf{p} \in \partial(1+\varepsilon)yJ$. Since the cone, $\mathcal{CH}(\mathsf{B}(\rho, (1+\varepsilon)yr) \cup \mathsf{p}')$ 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 4.4. 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)$_{\text{p8}}$, 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 4.5. It is not hard to see that Lemma 4.7 implies that bounded growth (P2)$_{\text{p8}}$ 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.

4.2.1. There is a small sketch and it can be computed quickly

To show that a small sketch exists (i.e., (P3)$_{\text{p8}}$ holds) is slightly harder and is tackled next.

Lemma 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 4.5$_{\text{p22}}$. We assume without loss of generality that r_1 is the maximum

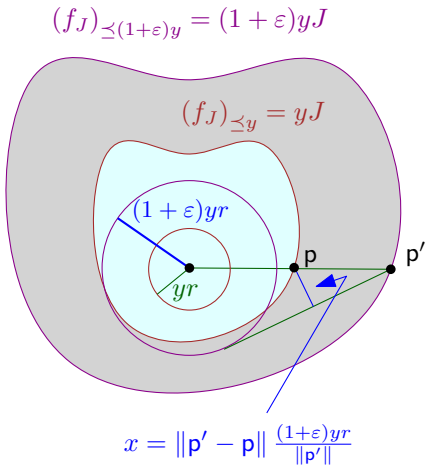


Figure 4.4: The $(1+\varepsilon)$ expansion of yJ contains $\mathsf{B}(\mathsf{p}, x)$.

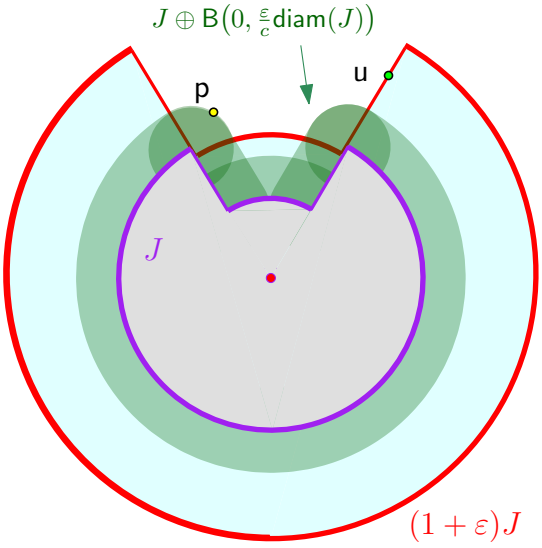


Figure 4.5: The object J is α-fat but not α'-rounded fat for any $\alpha' > 0$. In particular, the point p is in $J \oplus \mathsf{B}(0, (\varepsilon/c)\mathsf{diam}(J))$ but not in $(1+\varepsilon)J$, and the scaling distance function is not well defined at u.

of all the r_i. 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, $\bigcup_{i=1}^{n} J_i \subseteq \bigcup_{i=1}^{n} \mathsf{B}(0, \alpha r_i) \subseteq \mathsf{B}(0, \alpha r_1)$. We tile the ball $\mathsf{B}(0, \alpha r_1)$ 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, $\bigcup_{i=1}^{n} J_i \subseteq \bigcup_{\mathsf{c} \in \mathcal{A}} \mathsf{c}$, as clearly we have that,

$$\cup_{i=1}^{n} J_i = \mathsf{B}(0, \alpha r_1) \bigcap (\cup_{i=1}^{n} J_i) \;\subseteq\; (\cup_{\mathsf{c} \in \mathcal{C}} \mathsf{c}) \bigcap (\cup_{i=1}^{n} J_i) \subseteq \cup_{\mathsf{c} \in \mathcal{A}} \mathsf{c}.$$

Observe that $|\mathcal{I}| = |\mathcal{A}| \leq |\mathcal{C}| = O\big(\delta^{-d}\big)$. Next, we show that, $\bigcup_{\mathsf{c} \in \mathcal{A}} \mathsf{c} \subseteq \bigcup_{i \in \mathcal{I}} (1+\delta) J_i$. Suppose $\mathsf{c} \in \mathcal{A}$ and let $i \in \mathcal{I}$ be an index such that $\mathsf{c} \cap J_i \neq \emptyset$. Since $\mathsf{diam}(\mathsf{c}) \leq \delta \alpha r_1 / c'$, $\mathsf{c} \subseteq J_i \oplus \mathsf{B}(0, \delta \alpha r_1 / c')$. Now, $\frac{\delta \alpha r_1}{c'} \leq \frac{\delta \alpha^2 r_i}{c'} \leq \frac{\delta \alpha^2 \mathsf{diam}(J_i)}{2c'} \leq \frac{\delta \mathsf{diam}(J_i)}{c}$, where $c = 2\alpha$ is the constant from Lemma 4.7. Then, by Lemma 4.7, we have

$$\mathsf{c} \;\subseteq\; J_i \oplus \mathsf{B}\big(0, \delta \alpha r_1 / c'\big) \;\subseteq\; J_i \oplus \mathsf{B}\big(0, \delta \mathsf{diam}(J_i)/c\big) \;\subseteq\; (1+\delta) J_i,$$

which implies the claim. ∎

Lemma 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, $J + \mathsf{u} \subseteq J \oplus \mathsf{B}(0, \|\mathsf{u}\|) \;\subseteq J \oplus \mathsf{B}(0, \delta \mathsf{diam}(J)/c)$ as $\|\mathsf{u}\| \leq \delta \mathsf{diam}(J)/c$. Now, the claim follows by Lemma 4.7. ∎

Lemma 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 4.5$_{p22}$.*

Proof: Let $\mathcal{F} = \left\{ f_i = f_{J_i} \middle| i = 1, \ldots, n \right\}$ 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 $\ell\mathcal{J} = \left\{ \ell J_i \;\middle|\; J_i \in \mathcal{F} \right\}$ is connected, see Definition 2.6. Since $\ell J_i \subseteq \mathsf{B}(\mathsf{p}_i, \ell\alpha r_i) \subseteq \mathsf{B}(\mathsf{p}_i, l\alpha r)$, it follows that $\mathcal{B} = \left\{ \mathsf{B}(\mathsf{p}_i, \ell\alpha r) \;\middle|\; i = 1, \ldots, n \right\}$ 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 $\mathsf{B}(\mathsf{p}_{i_t}, \ell\alpha r) \cap \mathsf{B}\big(\mathsf{p}_{i_{t+1}}, \ell\alpha r\big) \neq \emptyset$, and thus $\left\|\mathsf{p}_{i_t} - \mathsf{p}_{i_{t+1}}\right\| \leq 2\ell\alpha r$, for $t = 1, \ldots, k-1$. By the triangle inequality,

$$\mathsf{diam}(\mathsf{P}) = \|\mathsf{u} - \mathsf{v}\| = \|\mathsf{p}_{i_1} - \mathsf{p}_{i_k}\| \leq \sum_{t=1}^{k-1} \left\|\mathsf{p}_{i_t} - \mathsf{p}_{i_{t+1}}\right\| \leq \sum_{t=1}^{k-1} 2\ell\alpha r = 2(k-1)\ell\alpha r \leq 2n\ell\alpha r, \qquad \blacksquare$$

We can now show that condition (P3)$_{p8}$ holds for the f_{J_i}.

Lemma 4.11. *Consider the setting of Lemma 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 $\left\{ f_j \;\middle|\; j \in \mathcal{I} \right\}$, 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 4.8 there is an index set $\mathcal{I} \subseteq \{1, \ldots, n\}$, with $|\mathcal{I}| = O\big(\delta^{-d}\big)$, such that for all y and any fixed j with $1 \leq j \leq n$, we have that $\bigcup_{i=1}^{n} yJ_{ij} \subseteq \bigcup_{i \in \mathcal{I}} (1 + \delta/4) yJ_{ij}$. Let r_i denote the radius of the ball for J_i from Definition 4.5, and let $r = \max_i r_i$. By Lemma 4.10, we have that, $\ell \geq \mathsf{diam}(\mathsf{P})/(2n\alpha r)$. Lemma 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 $\mathsf{u} = \mathsf{p}_j - \mathsf{p}_i$. Now for any $y \geq 0$,

$$\bigcup_{i=1}^{n} yJ_i \subseteq \bigcup_{j=1}^{n} \bigcup_{i=1}^{n} yJ_{ij} \subseteq \bigcup_{j=1}^{n} \bigcup_{i \in \mathcal{I}} (1 + \delta/4) yJ_{ij} = \bigcup_{j=1}^{n} \bigcup_{i \in \mathcal{I}} ((1 + \delta/4) yJ_i + (\mathsf{p}_j - \mathsf{p}_i)).$$

As $\ell \geq \mathsf{diam}(\mathsf{P})/(2n\alpha r)$, there is a $y_0 = O(\ell n/\delta)$ such that,

$$\|\mathsf{p}_j - \mathsf{p}_i\| \leq \delta \mathsf{diam}\big(y_0 J_i\big)/4c \leq \delta \mathsf{diam}\big((1 + \delta/4) y_0 J_i\big)/4c,$$

for all $1 \leq i, j \leq n$, where $c = 2\alpha$. Thus using Lemma 4.9 for $y = y_0$ on a $(1 + \delta/4)$ expansion of the J_i we have that,

$$\bigcup_{=1}^{n} \bigcup_{i \in \mathcal{I}} (1 + \delta/4) yJ_i + (\mathsf{p}_j - \mathsf{p}_i) \subseteq \bigcup_{j=1}^{n} \bigcup_{i \in \mathcal{I}} (1 + \delta/4)^2 yJ_i \subseteq \bigcup_{j=1}^{n} \bigcup_{i \in \mathcal{I}} (1 + \delta) yJ_i = \bigcup_{i \in \mathcal{I}} (1 + \delta) yJ_i.$$

It is easy to verify that this also holds for any $y \geq y_0$. As such by definition, $\left\{ f_i \;\middle|\; i \in \mathcal{I} \right\}$, is a (δ, y_0)-sketch. $\blacksquare$

4.2.2. 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 2.17$_{\mathrm{p10}}$.

Note that the result in Theorem 2.17 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 2.17 applies. Of course, Theorem 2.17 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.

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 $\mathrm{F}_{\mathsf{S}}(\mathsf{q}) = \max_{\mathsf{s}\in\mathsf{S}} \|\mathsf{q}-\mathsf{s}\|$; 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.

4.4. Satisfaction of conditions

Observation 4.12. *We have that* $(\mathrm{F}_i)_{\preceq y} = \bigcap_{\mathsf{u}\in\mathsf{P}_i} \mathsf{B}(\mathsf{u}, y)$, *and* $\mathsf{diam}\Big((\mathrm{F}_i)_{\preceq y}\Big) \leq 2y$.

Given the above observation, it is easy to see that Condition (P1)$_{\mathrm{p8}}$ is true, as $(\mathrm{F}_i)_{\preceq y}$ is a finite intersection of compact sets. The following Lemma shows that Condition (P2)$_{\mathrm{p8}}$ 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 4.12, it follows that we can choose the growth constant ζ to be 2.

Lemma 4.13. *For any i with $1 \leq i \leq n$, if $(\mathrm{F}_i)_{\preceq y} \neq \emptyset$, we have $(\mathrm{F}_i)_{\preceq y} \oplus \mathsf{B}(0, \varepsilon y) \subseteq (\mathrm{F}_i)_{\preceq (1+\varepsilon)y}$.*

Proof: Consider any point q in $(\mathrm{F}_i)_{\preceq y} \oplus \mathsf{B}(0, \varepsilon y)$. Now, one can easily verify that $\mathsf{B}(\mathsf{q}, (1+\varepsilon)y) \supseteq \mathsf{P}_i$ by the triangle inequality, and so $\mathsf{q} \in (\mathrm{F}_i)_{\preceq (1+\varepsilon)y}$. ∎

Lemma 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 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 $\|\mathsf{u}' - \mathsf{v}'\| \leq 2mz$, 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 $\|\mathsf{u} - \mathsf{v}\| \leq 2(m+1)z \leq 4mz$; 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 $\mathsf{B}(\mathsf{v}, y) \subseteq \mathsf{B}(\mathsf{u}, (1+\delta)y)$. Thus, for any i, $1 \leq i \leq m$, we have that $(\mathrm{F}_i)_{\preceq y} = \bigcap_{\mathsf{v} \in \mathsf{P}_i} \mathsf{B}(\mathsf{v}, y) \subseteq \mathsf{B}(\mathsf{u}, (1+\delta)y)$, for all $\mathsf{u} \in \mathsf{P}_1$ and $y \geq y_0$. As such, $(\mathrm{F}_i)_{\preceq y} \subseteq \bigcap_{\mathsf{u} \in \mathsf{P}_1} \mathsf{B}(\mathsf{u}, (1+\delta)y) = (\mathrm{F}_1)_{\preceq (1+\delta)y}$, and the result follows. ∎

Remark 4.15. It is not hard, if somewhat tedious, to verify that conditions (C1)–(C3)$_{\mathrm{p8}}$ hold. 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. Significantly, the center of the MEB is not perturbed too much when using coresets – see Lemma A.1 in Appendix A.

4.4.1. 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 Theorem 2.18$_{\mathrm{p11}}$: 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 $\mathsf{S}_1, \ldots, \mathsf{S}_n$. ∎

5. Conclusions

In this paper, 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 $f(\mathsf{q}) = \tau(\|\mathsf{q} - \mathsf{u}\|)$ 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.)

Acknowledgments. The authors thank the anonymous referees for their insightful and useful comments on the paper.

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.

[AESZ12] P. K. Agarwal, A. Efrat, S. Sankararaman, and W. Zhang. Nearest-neighbor searching under uncertainty. In *Proc. 31st ACM Sympos. Principles Database Syst.* (PODS), pages 225–236, 2012.

[Agg09] C. Aggarwal. *Managing and Mining Uncertain Data*. Springer, 2009.

[AI08] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, 2008.

[AM93] P. K. Agarwal and J. Matoušek. Ray shooting and parametric search. *SIAM J. Comput.*, 22:540–570, 1993.

[AM02] S. Arya and T. Malamatos. Linear-size approximate Voronoi diagrams. In *Proc. 13th ACM-SIAM Sympos. Discrete Algs.* (SODA), pages 147–155, 2002.

[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.

[BCKO08] M. de Berg, O. Cheong, M. van Kreveld, and M. H. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag, 3rd edition, 2008.

[BPR06] S. Basu, R. Pollack, and M. F. Roy. *Algorithms in Real Algebraic Geometry*. Algorithms and Computation in Mathematics. Springer, 2006.

[Cha10] T. M. Chan. Optimal partition trees. In *Proc. 26th Annu. 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.

[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.

[DRS09] N. N. Dalvi, C. Ré, and D. Suciu. Probabilistic databases: Diamonds in the dirt. *Commun. ACM*, 52(7):86–94, 2009.

[Eri96] J. Erickson. New lower bounds for Hopcroft's problem. *Discrete Comput. Geom.*, 16:389–418, 1996.

[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.

[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.

[HK13] 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.

[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.

[Mat92] J. Matoušek. Efficient partition trees. *Discrete Comput. Geom.*, 8:315–334, 1992.

[Mei93] S. Meiser. Point location in arrangements of hyperplanes. *Inform. Comput.*, 106:286–303, 1993.

[MNP06] R. Motwani, A. Naor, and R. Panigrahi. Lower bounds on locality sensitive hashing. In *Proc. 22nd Annu. Sympos. Comput. Geom.* (SoCG), pages 154–157, 2006.

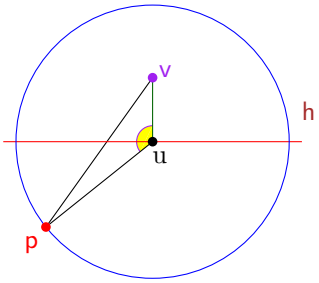
[SA95] M. Sharir and P. K. Agarwal. *Davenport-Schinzel Sequences and Their Geometric Applications*. Cambridge University Press, New York, 1995.

A. Bounding the size of intersection of balls of the same radius

Lemma A.1. *Let* $\mathsf{ball}(\mathsf{u},\mathsf{z})$ *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* $\mathsf{ball}(\mathsf{u},\delta\mathsf{z}) \subseteq \bigcap_{\mathsf{p}\in\mathsf{P}} \mathsf{B}(\mathsf{p},(1+\delta)\mathsf{z}) \subseteq \mathsf{ball}\Big(\mathsf{u}, \sqrt{4\delta+2\delta^2}\,\mathsf{z}\Big)$.

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) $\forall \mathsf{p}\in\mathsf{S} \quad \|\mathsf{p}-\mathsf{u}\| = \mathsf{z}$, and (ii) $\mathsf{u} \in \mathcal{CH}(\mathsf{S})$. Thus, any closed halfspace h^+ such that its boundary passes through u must contain at least one of the points of S.

Consider any point $\mathsf{v} \in \bigcap_{\mathsf{p}\in\mathsf{S}} \mathsf{B}(\mathsf{p},(1+\delta)\mathsf{z})$. Let h be the hyperplane passing through u, and orthogonal to $\mathsf{u}-\mathsf{v}$, 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



$$\begin{aligned}(1+\delta)^2\mathsf{z}^2 &\geq \|\mathsf{v}-\mathsf{p}\|^2 = \|\mathsf{v}-\mathsf{u}\|^2 + \|\mathsf{u}-\mathsf{p}\|^2 - 2\,\|\mathsf{v}-\mathsf{u}\|\,\|\mathsf{u}-\mathsf{p}\|\cos\angle\mathsf{vup}\\ &\geq \|\mathsf{v}-\mathsf{u}\|^2 + \|\mathsf{u}-\mathsf{p}\|^2 = \|\mathsf{v}-\mathsf{u}\|^2 + \mathsf{z}^2,\end{aligned}$$

since $\angle\mathsf{vup} \geq \pi/2$. Rearranging, we have $(2\delta+\delta^2)\mathsf{z}^2 \geq \|\mathsf{v}-\mathsf{u}\|^2$, thus implying $\sqrt{2\delta+\delta^2}\,\mathsf{z} \geq \|\mathsf{v}-\mathsf{u}\|$, We conclude that $\bigcap_{\mathsf{p}\in\mathsf{P}} \mathsf{B}(\mathsf{p},(1+\delta)\mathsf{z}) \subseteq \bigcap_{\mathsf{p}\in\mathsf{S}} \mathsf{B}(\mathsf{p},(1+\delta)\mathsf{z}) \subseteq \mathsf{ball}\Big(\mathsf{u}, \sqrt{2\delta+\delta^2}\,\mathsf{z}\Big)$. ∎

B. Basic properties of the functions

Lemma B.1. *Let* $\mathcal{F}$ *be a set of functions that satisfy the compactness* $(P1)_{p8}$ *and bounded growth* $(P2)_{p8}$ *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 at least two points, then by compactness it contains two distinct points x and y, such that $\|x-y\| = \mathsf{diam}(f_{\preceq 0}) > 0$. By bounded growth $(\mathrm{P2})_{\mathrm{p8}}$, with $\varepsilon = 1$, it follows

$$f_{\preceq 0} \subseteq f_{\preceq 0} \oplus \mathsf{B}\left(0, \frac{\|x-y\|}{\zeta}\right) \subseteq f_{\preceq 0} \oplus \mathsf{B}(0,\lambda_f(0)) \subseteq f_{\preceq 0},$$

as $\lambda_f(0) \geq \mathsf{diam}(f_{\preceq 0})/\zeta = \|x-y\|/\zeta$. Thus, $f_{\preceq 0} \oplus \mathsf{B}\left(0, \frac{\|x-y\|}{\zeta}\right) = f_{\preceq 0}$. Clearly $y \oplus \mathsf{B}\left(0, \frac{\|x-y\|}{\zeta}\right)$ contains a point 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 $\mathrm{d}(f,g) > 0$ for $f \neq g$ (this can be enforced using symbolic perturbations). Similarly, we also assume that the distances $\mathrm{d}(f,g)$ are distinct for all pairs of functions.

Observation B.2. *If* $\ell_{\mathrm{con}}(\mathcal{F}) = 0$ *for any non-empty set* $\mathcal{F}$, *then* $|\mathcal{F}| = 1$.

Lemma B.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 $(\mathrm{P2})_{\mathrm{p8}}$ with $\varepsilon = \zeta/2$, and the inequality $\lambda_f(y) \geq \mathsf{diam}(f_{\preceq y})/\zeta$, it follows that $f_{\preceq y} \oplus \mathsf{B}(0, \mathsf{diam}(f_{\preceq y})/2) \subseteq f_{\preceq (1+\zeta/2)y}$. Thus, $\mathsf{u} \oplus \mathsf{B}(0, \mathsf{diam}(f_{\preceq y})/2) \subseteq f_{\preceq (1+\zeta/2)y}$ and $\mathsf{B}(\mathsf{v}, \mathsf{diam}(f_{\preceq y})/2) \subseteq f_{\preceq (1+\zeta/2)y}$. Since $\|\mathsf{u} - \mathsf{v}\| \leq \mathsf{diam}(f_{\preceq y})$, it follows that the entire segment uv is in $f_{\preceq (1+\zeta/2)y}$. ∎

Observation B.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 $i = 1, \ldots, k$, and $\mathsf{uv} \subseteq \bigcup_{j=1}^{k} A_j$. Then, the set $\{A_1, \ldots, A_k\}$ is connected, see Definition 2.6$_{\mathrm{p7}}$.*

Lemma B.5. *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, $\ell_{\mathrm{con}}(\mathcal{G}) \leq (1+\delta)(1+\zeta/2)\max(y, \ell_{\mathrm{con}}(\mathcal{F}))$.*

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 by definition $\ell_{\mathrm{con}}(\mathcal{F}) = \ell_{\mathrm{con}}(\mathcal{G}) = 0$, and the result holds. If $m > 1$, we need to show that $(f_i)_{\preceq y'}$, for $i = 1, \ldots, k$, are connected, where $y' = (1+\delta)(1+\zeta/2)\ell$, and $\ell = \max(y, \ell_{\mathrm{con}}(\mathcal{F}))$. By definition, $\mathcal{F}_{\preceq \ell}$ is a connected set. Consider any distinct i and j with values between 1 and 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 $r = 1, \ldots, s-1$. Consider such an index, say i_r, such that $i_r > k$, i.e., $f_{i_r} \notin \mathcal{G}$. Since,

$$U = (f_{i_r})_{\preceq \ell} \cap \left(f_{i_{r-1}}\right)_{\preceq \ell} \neq \emptyset \qquad \text{and} \qquad U' = (f_{i_r})_{\preceq \ell} \cap \left(f_{i_{r+1}}\right)_{\preceq \ell} \neq \emptyset$$

one can choose two arbitrary points $\mathsf{u} \in U$ and $\mathsf{v} \in U'$. Now the entire segment $\mathsf{uv} \subseteq (f_{i_r})_{\preceq (1+\zeta/2)\ell}$, by Lemma B.3. Since $(1+\zeta/2)\ell \geq y$, it follows by the definition of a (δ, y) sketch, see Definition 2.10, that $\mathsf{uv} \subseteq (f_{i_r})_{\preceq (1+\zeta/2)\ell} \subseteq \mathcal{G}_{\preceq (1+\zeta/2)(1+\delta)\ell}$. By Observation B.4, the sets in the minimal cover of uv by the sublevel sets $(\bar{f}_i)_{\preceq (1+\zeta/2)(1+\delta)\ell}$, are connected. It follows that $(f_{i_r})_{\preceq (1+\zeta/2)\ell}$ can be replaced by a subset of $\left\{ (f_i)_{\preceq (1+\zeta/2)(1+\delta)\ell} \;\middle|\; i = 1, \ldots, k \right\}$, and the property of neighbor intersections is still valid in the chain. We replace each occurrence of the set $(f_{i_r})_{\preceq (1+\zeta/2)\ell}$ for $i_r > k$, by the corresponding chain. It is easy to see that the resulting chain connects up $(f_{i_1})_{\preceq (1+\zeta/2)(1+\delta)\ell}$ and $(f_{i_s})_{\preceq (1+\zeta/2)(1+\delta)\ell}$. Now, removing duplicate elements does not affect the neighbor intersection property of the resulting chain. ∎

The following testifies that a sketch approximates the distance to a set of functions.

Lemma B.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 $\mathrm{d}(\mathsf{q}, \mathcal{F}) \geq y_0$. Then we have that $\mathrm{d}(\mathsf{q}, \mathcal{G}) \leq (1+\delta)\mathrm{d}(\mathsf{q}, \mathcal{F})$.*

Proof: Let $\ell = \mathrm{d}(\mathsf{q}, \mathcal{F})$ and let $f \in \mathcal{F}$ be a witness to this distance; that is, $\mathsf{q} \in f_{\preceq \ell}$. As $\ell \geq y_0$, we have that $f_{\preceq \ell} \subseteq \bigcup_{g \in \mathcal{G}} g_{\preceq (1+\delta)\ell}$ by the sketch property (Definition 2.10). As such there is some function $g \in \mathcal{G}$, such that $\mathsf{q} \in g_{\preceq (1+\delta)\ell}$. It follows that $\mathrm{d}(\mathsf{q}, g) \leq (1+\delta)\mathrm{d}(\mathsf{q}, \mathcal{F})$. ∎