

National Kapodistrian University of Athens
Graduate Program of Logic, Algorithms and Computations
Department of Mathematics

Geometric Proximity Problems in High Dimensions

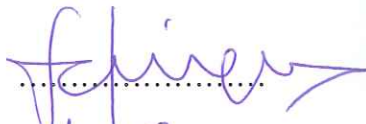


Georgia Avarikioti

Thesis advisor: Ioannis Z. Emiris

Submitted in part fulfilment of the requirements for
the Master Degree in Theoretical Computer Science
at the University of Athens, June 2017

Η παρούσα Διπλωματική Εργασία
εκπονήθηκε στα πλαίσια των σπουδών
για την απόκτηση του
Μεταπτυχιακού Διπλώματος Ειδίκευσης
στη
Λογική και Θεωρία Αλγορίθμων και Υπολογισμού
που απονέμει το
Τμήμα Μαθηματικών
του
Εθνικού και Καποδιστριακού Πανεπιστημίου Αθηνών

Εγκρίθηκε την 21/6/17 από Εξεταστική Επιτροπή
αποτελούμενη από τους:

<u>Όνοματεπώνυμο</u>	<u>Βαθμίδα</u>	<u>Υπογραφή</u>
1. <u>Ι.Ε.ΜΙΡΗΣ</u>	<u>ΚΑΘΗΓΗΤΗΣ</u>	
2. <u>Β. ΖΗΣΙΜΟΠΟΥΛΟΣ</u>	<u>ΚΑΘΗΓΗΤΗΣ</u>	
3. <u>Α. ΠΑΓΟΥΡΤΖΗΣ</u>	<u>ΑΝ. ΚΑΘΗΓΗΤΗΣ</u>	

Abstract

Geometric proximity problems is a class of problems in computational geometry that involve estimation of distances between geometric objects. In this work, we focus on two specific problems of this class, the computation of r -nets and the near neighbor decision problem on high dimensional spaces under the Euclidean distance, both of which are powerful tools in computational and metric geometry. Specifically, we present a new randomized algorithm which efficiently computes high dimensional approximate r -nets with respect to Euclidean distance. For any fixed $\epsilon > 0$, the approximation factor is $1 + \epsilon$ and the complexity is polynomial in the dimension and subquadratic in the number of points; the algorithm succeeds with high probability. We improve upon the best previously known (LSH-based) construction of Eppstein et al. in terms of complexity, by reducing the dependence on ϵ , provided that ϵ is sufficiently small. Moreover, our method does not require LSH but follows Valiant's approach in designing a sequence of reductions of our problem to other problems in different spaces, under Euclidean distance or inner product, for which r -nets are computed efficiently and the error can be controlled. Our result immediately implies efficient solutions to a number of geometric problems in high dimension, such as finding the $(1 + \epsilon)$ -approximate k -th nearest neighbor distance in time subquadratic in the size of the input. Additionally, we propose a new and simple data structure for the c -approximate near neighbor decision problem in high-dimensional spaces using linear space and sublinear query time for any $c > 1$: given an LSH family of functions for some metric space, we randomly project points to vertices of the Hamming cube in dimension $\leq \log n$, where n is the number of input points. The projected space contains strings which serve as keys for buckets containing the input points. The query algorithm simply projects the query point, then examines points which are assigned to the same or nearby vertices on the Hamming cube. We analyze in detail the query time for some standard LSH families.

Περίληψη

Τα γεωμετρικά προβλήματα εγγύτητας είναι μια κλάση προβλημάτων στην υπολογιστική γεωμετρία που περιλαμβάνει την εκτίμηση αποστάσεων μεταξύ γεωμετρικών αντικειμένων. Σε αυτή την εργασία, εστιάζουμε σε δύο συγκεκριμένα προβλήματα της κλάσης αυτής, τον υπολογισμό των ρ -δικτύων και το πρόβλημα απόφασης του κοντινότερου γείτονα σε χώρους υψηλών διαστάσεων υπό την Ευκλείδεια απόσταση, τα οποία αποτελούν ισχυρά εργαλεία στην υπολογιστική και τη μετρική γεωμετρία. Συγκεκριμένα, παρουσιάζουμε έναν νέο πιθανοτικό αλγόριθμο που υπολογίζει αποδοτικά προσεγγιστικά ρ -δίκτυα ως προς την Ευκλείδεια απόσταση. Για οποιοδήποτε σταθερό $\epsilon > 0$, ο προσεγγιστικός παράγοντας είναι $1+\epsilon$ και η πολυπλοκότητα πολυωνυμική στη διάσταση και υποτετραγωνική στο πλήθος των σημείων. Ο αλγόριθμος επιτυγχάνει με μεγάλη πιθανότητα. Βελτιώνουμε ως προς την πολυπλοκότητα την προηγούμενη καλύτερη γνωστή κατασκευή του Eppstein που βασιζόταν στο LSH, μειώνοντας την εξάρτηση από το ϵ δεδομένου ότι το ϵ είναι επαρκώς μικρό. Η μέθοδός μας δεν χρησιμοποιεί το LSH, αλλά αντί αυτού ακολουθεί την προσέγγιση του Valiant, σχεδιάζοντας μια σειρά από αναγωγές του προβλήματός μας σε άλλα προβλήματα σε διαφορετικούς χώρους, υπό την Ευκλείδεια απόσταση ή το εσωτερικό γινόμενο, για τα οποία τα ρ -δίκτυα υπολογίζονται αποδοτικά και το σφάλμα μπορεί να ελεγχθεί. Το αποτέλεσμά μας άμεσα συνεπάγεται αποδοτικές λύσεις σε ένα πλήθος γεωμετρικών προβλημάτων σε υψηλές διαστάσεις, όπως η εύρεση της απόστασης του $(1+\epsilon)$ -προσεγγιστικού k -κοντινότερου γείτονα σε χρόνο υποτετραγωνικό στο μέγεθος της εισόδου. Επιπλέον, προτείνουμε μια νέα και απλή στην κατασκευή βάση δεδομένων για το πρόβλημα απόφασης του δ -προσεγγιστικού κοντινότερου γείτονα σε χώρους υψηλών διαστάσεων, χρησιμοποιώντας γραμμικό χώρο και υπογραμμικό χρόνο ερώτησης για οποιοδήποτε $\delta > 1$: δεδομένης μιας οικογένειας LSH συναρτήσεων για έναν μετρικό χώρο, προβάλλουμε τυχαία τα σημεία σε κόμβους του κύβου Hamming διάστασης $\leq \log n$, όπου n είναι ο αριθμός των σημείων εισόδου. Ο προβαλλόμενος χώρος περιέχει συμβολοσειρές που λειτουργούν ως κλειδιά για τους «κουβάδες» που περιέχουν τα σημεία της εισόδου. Ο αλγόριθμος ερώτησης απλά προβάλλει το σημείο-ερώτηση κι έπειτα εξετάζει τα σημεία που έχουν αντιστοιχηθεί στον ίδιο ή διπλανό κόμβο του κύβου Hamming. Αναλύουμε λεπτομερώς τη χρονική πολυπλοκότητα της ερώτησης για κάποιες βασικές οικογένειες LSH.

Acknowledgements

I would like to express my gratitude to my supervisor Professor I. Z. Emiris, for his constant support and persistent help. Moreover, I would like to express the deepest appreciation to my colleague, Ioannis Psarros, for his unconditional guidance, continuous mentoring, delightful collaboration and friendship. Without his engagement this thesis would not have been possible.

An unexamined life is not worth living.

Socrates

Contents

Abstract	3
Acknowledgements	5
1 Background & State-of-the-Art	9
1.1 Introduction	9
1.2 Nearest Neighbor Search	11
1.2.1 Problem definition	11
1.2.2 Previous work	12
1.2.3 Contribution	13
1.3 R-nets	14
1.3.1 Problem definition	14
1.3.2 Previous Work.	15
1.3.3 Contribution	16
2 Practical linear-space approximate near neighbors in high dimensions	18
2.1 Introduction	18
2.2 Data structures	19
2.3 The ANN under Euclidean and Manhattan metrics	22
2.3.1 The ℓ_2 case	23

2.3.2	The ℓ_1 case	28
2.4	Implementation and experimental results	29
3	High-dimensional approximate r-nets	30
3.1	Introduction	30
3.2	Points on a sphere under inner product	31
3.2.1	Crude approximate nets.	32
3.2.2	Approximate inner product nets.	38
3.3	Approximate nets in high dimensions	46
3.3.1	From arbitrary to unit vectors.	46
3.3.2	Approximate nets under Euclidean distance.	48
3.4	Applications	56
3.4.1	A general framework for high dimensional distance problems	57
3.4.2	k -th nearest neighbor distance	61
4	Conclusion	63
4.1	Summary of Thesis Achievements	63
4.2	Future Work	64

Chapter 1

Background & State-of-the-Art

1.1 Introduction

Geometric proximity problems is a class of problems in computational geometry that involve estimation of distances between geometric objects. The computation of geometric and graph structures based on proximity, such as the Voronoi diagram and the neighborhood graph, as well as retrieval problems such as the nearest neighbor search and the concept of range search, are typical problems in this area. Other important geometric proximity problems are the the Euclidean minimum spanning tree, the computation of the diameter of a set of points in the Euclidean space and the Euclidean k -center problem. All these problems have various applications in diverse areas of computer science such as data mining, information retrieval, statistical data analysis, computer vision and machine learning. Therefore, it is very important to provide algorithms with strong theoretical guarantees as well as space and time efficiency in practice.

Despite the extensive research towards providing efficient algorithms for proximity problems, the curse of dimensionality when dealing with high dimensional data imposes a great barrier to overcome. In an attempt to overcome this exponential explosion that constitutes the low dimensional algorithms inefficient and useless in practice, we relax the accuracy of the solution and aim for more efficient approximation schemes. The most common way to handle high dimensional data sets is dimension reduction techniques. An important tool towards this direction is the JohnsonLindenstrauss lemma, which states that for any given pointset there exists a

low-distortion embedding of the points from high-dimensional into low-dimensional Euclidean space. Formally,

Definition 1.1 *Given a pointset $X \subset \mathbb{R}^d$, $|X| = n$, there exists a distribution over linear maps $f : \mathbb{R}^d \rightarrow \mathbb{R}^{d'}$ with $d' = O(\log n/\epsilon^2)$ s.t., for any $p, q \in \mathbb{R}^d$,*

$$(1 - \epsilon)\|p - q\| \leq \|f(p) - f(q)\| \leq (1 + \epsilon)\|p - q\|.$$

This lemma is the cornerstone of dimensionality reduction, since it guarantees that a set of point in a high-dimensional space can be embedded into a space of much lower dimension while the distances between the points are nearly preserved.

Another important tool in high dimensional geometry is Locality-Sensitive Hashing (LSH). LSH is a hash function that aims to maximize the probability of collision for two similar items. Specifically, LSH hashes input items so that similar items map to the same buckets with high probability, where the number of the buckets is much lower than the universe of possible input items. We give the formal definition,

Definition 1.2 *Let reals $r > 0$ (threshold), $c > 0$ (approximation factor), $1 > p_1 > 0$ and $1 > p_2 > 0$ (probabilities). We call a family F of hash functions an LSH family for a metric space \mathcal{M} if, for any $x, y \in \mathcal{M}$, and h distributed uniformly in F , it holds:*

- $d_{\mathcal{M}}(x, y) \leq r \implies Pr[h(x) = h(y)] \geq p_1,$
- $d_{\mathcal{M}}(x, y) \geq cr \implies Pr[h(x) = h(y)] \leq p_2.$

We are usually interested in a family where $p_1 > p_2$. This family F is called (r, cr, p_1, p_2) -sensitive. There are many renown LSH families, such as the projection on random lines [16] and the projection with hyperplanes [13] which we will present later in Chapter 2.

In this work, we focus on two specific proximity problems, the computation of r-nets and the near neighbor decision problem, both of which are powerful tools in computational and metric geometry. We study these problems on high dimensional spaces under the Euclidean distance and we present a novel approach for each one of them. On Chapter 2, we focus on nearest neighbor search problem and we propose a simple and efficient in practice LSH-based approach,

which we analyze for specific LSH families. On Chapter 3, we examine the efficient computation of r -nets and we present a new approach to this problem introduced by Valiant [34], not LSH-based, that actually takes advantage of phenomena that occur when the dimension is high. In Chapter 4 we conclude with a summary of our contribution and discussion concerning future research direction.

1.2 Nearest Neighbor Search

The search for a nearest neighbor among points on a specific database is a fundamental computational task that arises in a variety of application areas, such as information retrieval, data mining, pattern recognition, machine learning, computer vision, and statistical data analysis. In many of these cases the database points are represented as vectors in some high dimensional space. Although, dimension reduction techniques are commonly used in these applications, vector spaces of several hundred dimensions are typical. Due to the curse of dimensionality that appears in this problem when the dimension is high, we usually aim for an efficient approximation search, which is often as good as an exact search regarding practical applications.

1.2.1 Problem definition

In this subsection we focus on the problem of Approximate Nearest Neighbor search in Euclidean or other metric spaces, when the dimension is high; Typically one assumes for dimension $d \gg \log n$, where n denotes the number of input data points; in fact, dimension is an input parameter, so we need to address the curse of dimensionality. Due to known reductions, e.g. [20], it is apparent that one may focus on designing an efficient data structure for the Approximate Near Neighbor (ANN) problem instead of directly solving the Approximate Nearest Neighbor problem. The former is a decision problem, whose output may contain a witness point (as in Definition 1.3 below), whereas the latter is an optimization question. Here we trade exactness for efficiency in order to tackle the case of general dimension, where dimension is an input parameter. The $(1 + \epsilon, r)$ -ANN problem, where $c = 1 + \epsilon$, is defined as follows.

Definition 1.3 (Approximate Near Neighbor problem) *Let $(\mathcal{M}, d_{\mathcal{M}})$ be a metric space.*

Given $P \subseteq \mathcal{M}$, and reals $r > 0$ and $\epsilon > 0$, build a data structure s.t. for any query $q \in \mathcal{M}$, there is an algorithm performing as follows:

- if $\exists p^* \in P$ s.t. $d_{\mathcal{M}}(p^*, q) \leq r$, then return any point $p' \in P$ s.t. $d_{\mathcal{M}}(p', q) \leq (1 + \epsilon) \cdot r$,
- if $\forall p \in P$, $d_{\mathcal{M}}(p, q) > (1 + \epsilon) \cdot r$, then report “no”.

An important approach for such problems today is Locality Sensitive Hashing (LSH). It has been designed precisely for problems in general dimension. The LSH method is based on the idea of using hash functions enhanced with the property that it is more probable to map nearby points to the same bucket.

Definition 1.4 Let reals $r_1 < r_2$ and $p_1 > p_2 > 0$. We call a family F of hash functions (p_1, p_2, r_1, r_2) -sensitive for a metric space \mathcal{M} if, for any $x, y \in \mathcal{M}$, and h distributed uniformly in F , it holds:

- $d_{\mathcal{M}}(x, y) \leq r_1 \implies Pr[h(x) = h(y)] \geq p_1$,
- $d_{\mathcal{M}}(x, y) \geq r_2 \implies Pr[h(x) = h(y)] \leq p_2$.

1.2.2 Previous work

Let us survey previous work, focusing on methods whose complexity has polynomial, often even linear, dependence on the dimension. LSH was introduced in [23, 20] and yields data structures with query time $O(dn^\rho)$ and space $O(n^{1+\rho} + dn)$. Since then, the optimal value of the LSH exponent, $\rho < 1$, has been extensively studied for several interesting metrics, such as ℓ_1 and ℓ_2 . In a series of papers [23, 16, 29, 6, 30], it has been established that the optimal value for the Euclidean metric is $\rho = 1/c^2 \pm o(1)$, and that for the Hamming distance is $\rho = 1/c \pm o(1)$.

In contrast to the definition above, which concerns data-independent LSH, quite recently the focus has been shifted to data-dependent LSH. In the latter case, the algorithms exploit the fact that every dataset has some structure and consequently this approach yields better bounds for the LSH exponent. Specifically, in [10] they showed that $\rho = 1/(2c - 1)$ for the ℓ_1 distance and $\rho = 1/(2c^2 - 1)$ for the ℓ_2 distance.

The data-dependent algorithms, though better in theory, are quite challenging in practice. In [7], they present an efficient implementation of one part of [10]. Another attempt towards practicality for a data-dependent algorithm was recently made in [9], where they presented a new approach based on LSH forests. Typically though, data-independent algorithms, such as the one proposed in this work, yield better results in practice than data-dependent algorithms.

For practical applications, an equally important parameter is memory usage. Most of the previous work in the (near) linear space regime focuses on the case that c is greater than 1 by a constant term. When c approaches 1, these methods become trivial in the sense that query time becomes linear in n . One such LSH-based approach [31] offers query time proportional to $dn^{O(1/c)}$, which is sublinear in n only for large enough $c > 1$. Improvements on practical aspects of the above result led to the novel multi-probe scheme for LSH [27]. Two noteworthy exceptions are the recently accepted papers [8] and [24], where they achieve near-linear space and sublinear query time, even for $c \rightarrow 1^+$.

Another line of work that achieves linear space and sublinear query time for the Approximate Nearest Neighbor problem is based on random projections to drastically lower-dimensional spaces, where one can simply search using tree-based methods, such as BBD-trees [3, 4]. This method relies on a projection extending the type of projections typically based on the Johnson-Lindenstrauss lemma. The new projection only ensures that an approximate nearest neighbor in the original space can be found among the preimages of k approximate nearest neighbors in the projection. From the practical perspective, similar ideas have been employed in [32]. Random projections which preserve the relative order of points have been also considered in [26].

1.2.3 Contribution

Towards practicality, we present a new data structure for the c -approximate near neighbor decision problem in high-dimensional spaces for any $c > 1$. This problem has been mainly addressed by Locality Sensitive Hashing (LSH), which offers polynomial dependence on the dimension, query time sublinear in the size of the dataset, and subquadratic space requirement. While, in practice, it is important to ensure linear space usage, most previous work in this regime focuses on the case that c exceeds 1 by a constant term. In this work, we propose a

a simple data structure using linear space and sublinear query time for any $c > 1$: given an LSH family of functions for some metric space, we randomly project points to vertices of the Hamming cube in dimension $\leq \log n$, where n is the number of input points. The projected space contains strings which serve as keys for buckets containing the input points. The query algorithm simply projects the query point, then examines points which are assigned to the same or nearby vertices on the Hamming cube. We analyze in detail the query time for some standard LSH families.

1.3 R-nets

1.3.1 Problem definition

We study r -nets, a powerful tool in computational and metric geometry, with several applications in approximation algorithms. An r -net for a metric space $(X, \|\cdot\|)$, $|X| = n$ and for numerical parameter r is a subset $R \subseteq X$ such that the closed $r/2$ -balls centered at the points of R are disjoint, and the closed r -balls around the same points cover all of X . Thus, the construction of r -nets provides a sketch of the point set for distances that are r or larger. Nets are a useful tool in presenting point sets hierarchically. In particular, computing nets of different resolutions and linking between different levels, leads to a tree like data-structure that can be used to facilitate many tasks. Nets can be defined in any metric space, but in Euclidean space a grid can sometimes provide an equivalent representation. Furthermore, the problem of computing an r -net is closely related to the very well-studied k -center problem.

We define approximate r -nets analogously. Formally,

Definition 1.5 *Given a pointset $X \subseteq \mathbb{R}^d$, a distance parameter $r \in \mathbb{R}$ and an approximation parameter $\epsilon > 0$, a $(1 + \epsilon)r$ -net of X is a subset $R \subseteq X$ s.t. the following properties hold:*

1. (packing) *For every $p, q \in R$, $p \neq q$, we have that $\|p - q\|_2 \geq r$.*
2. (covering) *For every $p \in X$, there exists a $q \in R$ s.t. $\|p - q\|_2 \leq (1 + \epsilon)r$.*

The only difference between the definition of exact r -nets and approximate r -nets is that we relax the covering property. This allows us to compute approximate r -nets in high dimensional

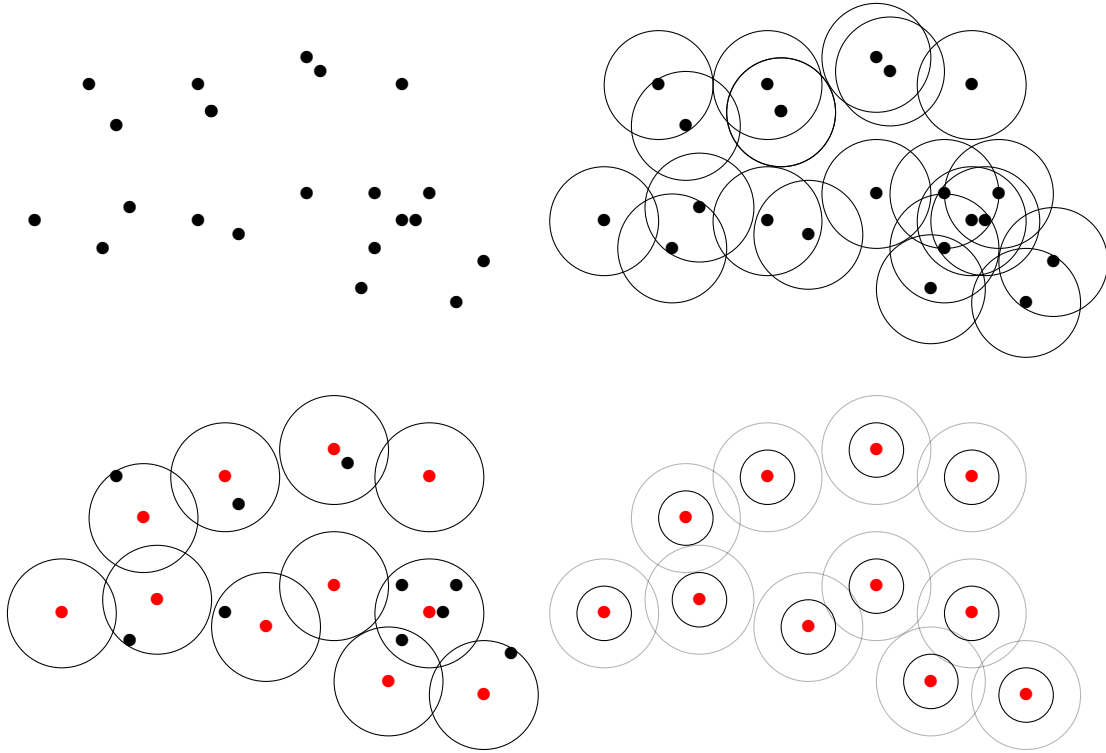


Figure 1.1: The construction of r -nets depicted in 4 steps. Left-top: The input point set. Right-top: All initial points are considered unmarked with balls of radius r centered at them. Left-bottom: Net points are marked red and every net point covers points in distance at most r . Any point may be covered by more than one net point. Right-bottom: The set of net points representing the initial pointset in resolution r .

Euclidean space with time complexity subquadratic in the number of points and polynomial in the dimension. Our algorithm outperforms the best known algorithm in this setting, by reducing the dependence on ϵ and thus, allowing better tradeoff between solution quality and running time.

1.3.2 Previous Work.

Finding r -nets can be addressed naively by considering initially all points of X unmarked: while there remains an unmarked point p , the algorithm adds it to R and marks all other points within distance r from p . (Figure 1) The performance of this algorithm can be improved by using grids and hashing [19]. However, the complexity remains too large when dealing with big data in high dimension. The naive algorithm is quadratic in n and the grid approach is in $O(d^{d/2}n)$, hence it is relevant only for constant dimension d [22]. In [21], they show that an approximate

net hierarchy for an arbitrary finite metric can be computed in $O(2^{ddim}n \log n)$, where $ddim$ is the doubling dimension. This is satisfactory when doubling dimension is constant, but requires a vast amount of resources when it is high.

When the dimension is high, there is need for algorithms with time complexity polynomial in d and subquadratic in n . One approach, which computes $(1 + \epsilon)r$ -nets in high dimension, is that of [17], which uses Locality Sensitive Hashing (LSH), see, e.g. [6]. For sufficiently small $\epsilon > 0$, the resulting time complexity is $\tilde{O}(dn^{2-\Theta(\epsilon)})$, where \tilde{O} hides polylogarithmic factors.

In general, high dimensional analogues of classical geometric problems have been mainly addressed by LSH. For instance, the approximate closest pair problem can be trivially solved by performing n approximate nearest neighbor (ANN) queries. For sufficiently small ϵ , this costs $\tilde{O}(dn^{2-\Theta(\epsilon)})$ time, due to the complexity factor of an LSH query. Several other problems have been reduced to ANN queries [18]. Recently, Valiant [33], [34] presented an algorithm for the approximate closest pair problem in time $\tilde{O}(dn^{2-\Theta(\sqrt{\epsilon})})$. This is a different approach in the sense that while LSH exploits dimension reduction through random projections, the algorithm of [34] is inspired by high dimensional phenomena. One main step of the algorithm is that of lifting the pointset up to a higher dimension. Moreover, [2] improved on Valiant's algorithm for the approximate closest pair problem by presenting an algorithm with randomized time near $dn + n^{2-\Theta(\epsilon^{1/3}/\log 1/\epsilon)}$. This approach employs probabilistic polynomial threshold function representations and improves the exponential dependence on the error ϵ from square root of ϵ to cubic root of ϵ .

1.3.3 Contribution

In this work, we present a new randomized algorithm which efficiently computes high dimensional approximate r -nets with respect to Euclidean distance. For any fixed $\epsilon > 0$, the approximation factor is $1 + \epsilon$ and the complexity is polynomial in the dimension and subquadratic in the number of points; the algorithm succeeds with high probability. Specifically, we improve upon the best previously known (LSH-based) construction of Eppstein et al. [17] in terms of complexity, by reducing the dependence on ϵ , provided that ϵ is sufficiently small. Moreover, our method does not require LSH but follows Valiant's [34] approach in designing a sequence of reductions of our problem to other problems in different spaces, under Euclidean distance

or inner product, for which r -nets are computed efficiently and the error can be controlled. Our result immediately implies efficient solutions to a number of geometric problems in high dimension, such as finding the $(1 + \epsilon)$ -approximate k -th nearest neighbor distance in time subquadratic in the size of the input.

Chapter 2

Practical linear-space approximate near neighbors in high dimensions

2.1 Introduction

In this section, we address the problem of designing a data structure that allows efficient search for approximate near neighbors. More specifically, given a database consisting of a set of vectors in some high dimensional Euclidean space and a specific search radius, we construct a space-efficient (linear to the input) data structure that, given a query vector, allows us to search for a close or nearly close vector in the database with time complexity sublinear to the input.

Towards this end, we specify a random projection from any space endowed with an LSH-able metric, to the vertices of the Hamming hypercube of dimension $\log n$, $\{0, 1\}^{\log n}$, where n is the number of input points. Random projections which map points from a high dimensional Hamming space to lower dimensional Hamming space have been already used in the ANN context [25]. The projected space contains strings which serve as keys for buckets containing the input points. The query algorithm simply projects the query point, then examines points which are assigned to the same or nearby vertices on the Hamming cube.

Our strategy resembles the multi-probe approach in the sense that after locating the query, we start searching in "nearby" buckets. However, in our case, nearby buckets are simply the ones which are close to each other with respect to the Hamming distance of their keys, whereas in

the multi-probe case, the nearby buckets are the ones with high probability of success for a given query. Computing which are the right buckets to explore in the multi-probe approach is potentially harder than in our case.

The random projection in our algorithm relies on the existence of an LSH family for the input metric. We study standard LSH families for ℓ_2 and ℓ_1 , for which we achieve query time $O(dn^{1-\delta})$ where $\delta = \Theta(\epsilon^2)$, $\epsilon \in (0, 1]$. The constants appearing in δ vary with the LSH family, but it holds that $\delta > 0$ for any $\epsilon > 0$. The space and preprocessing time are both linear for constant probability of success, which is important in practical applications.

This approach is illustrated with an open-source implementation [12], that reports on a series of experiments with n up to 10^6 and d up to 1000, where results are very encouraging. It is evident that the proposed algorithm is 8.5–80 times faster than brute force search, depending on the difficulty of the dataset. Furthermore, the experimental results suggest our algorithm either is comparable or outperforms the LSH-based library [7] FALCONN in terms of memory usage and query time.

The rest of the chapter is structured as follows. The next section states our main algorithmic and complexity results for the (c, r) -ANN problem, assuming an LSH family. We conclude this section with a discussion on the parameters of the algorithms. In section 2.3 we state our results for the the Euclidean and Manhattan metrics. We conclude with section 2.4, where we mention our implementation [12] and discuss the experimental results.

2.2 Data structures

This section introduces our main data structure, and the corresponding algorithmic tools. We start our presentation with an idea applicable to any metric admitting an LSH-based construction, aka LSH-able metric.

The algorithmic idea is to apply a random projection from any LSH-able metric to the Hamming hypercube, as shown in the figure 2.1. Given an LSH family of functions for some metric space, we uniformly select d' hash functions, where d' is specified later. The nonempty buckets defined by any hash function are randomly mapped to $\{0, 1\}$, with equal probability for each bit. Thus, points are projected to the Hamming cube of dimension d' . Thus, we obtain binary strings

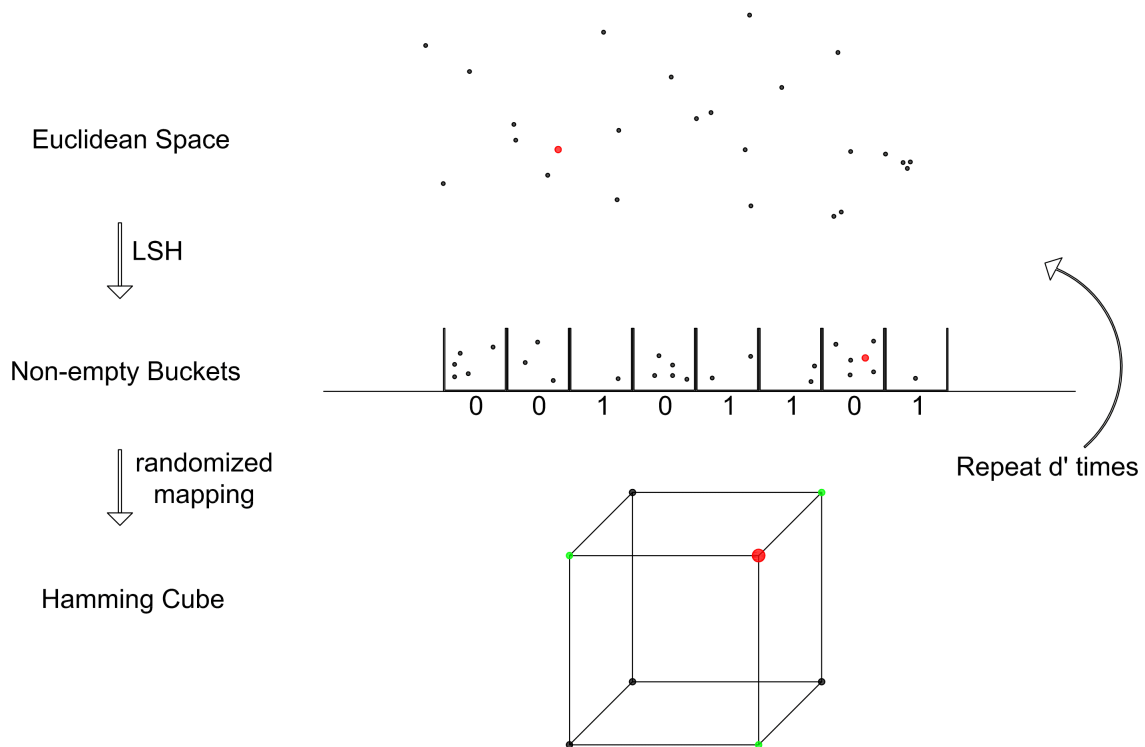


Figure 2.1: In this figure we illustrate the algorithmic idea of the proposed data structure. At first, we partition the pointset into buckets using an LSH family and then we randomly project them to the Hamming hypercube.

serving as keys for buckets containing the input points. The query algorithm projects a given point, and tests points assigned to the same or nearby vertices on the hypercube. To achieve the desired complexities, it suffices to choose $d' = \log n$.

In Algorithms 1 and 2, we present the preprocessing and query algorithms.

The main lemma below describes the general ANN data structure whose complexity and performance depends on the LSH family that we assume is available. The proof details the data structure construction.

Lemma 2.1 (Main) *Given a (p_1, p_2, r, cr) -sensitive hash family F for some metric $(\mathcal{M}, d_{\mathcal{M}})$ and input dataset $P \subseteq \mathcal{M}$, there exists a data structure for the (c, r) -ANN problem with space $O(dn)$, time preprocessing $O(dn)$, and query time $O(dn^{1-\delta} + n^{H((1-p_1)/2)})$, where*

$$\delta = \delta(p_1, p_2) = \frac{(p_1 - p_2)^2}{(1 - p_2)} \cdot \frac{\lg e}{4},$$

where e denotes the basis of the natural logarithm, and $H(\cdot)$ is the binary entropy function. The bounds hold assuming that computing $d_{\mathcal{M}}(\cdot)$ and computing the hash function cost $O(d)$. Given some query $q \in \mathcal{M}$, the building process succeeds with constant probability.

Proof. The first step is a random projection to the Hamming space of dimension d' , for d' to be specified in the sequel. We first sample $h_1 \in F$. We denote by $h_1(P)$ the image of P under h_1 , which is a set of nonempty buckets. Now each nonempty bucket $x \in h_1(P)$ is mapped to $\{0, 1\}$: with probability $1/2$, set $f_1(x) = 0$, otherwise set $f_1(x) = 1$.

This is repeated d' times, and eventually for $p \in \mathcal{M}$, we compute the function

$$f(p) = (f_1(h_1(p)), \dots, f_{d'}(h_{d'}(p)))$$

, where $f : P \rightarrow \{0, 1\}^{d'}$. Now, observe that

$$\begin{aligned} d_{\mathcal{M}}(p, q) \leq r &\implies \mathbb{E}[\|f_i(h_i(p)) - f_i(h_i(q))\|_1] \leq \\ &\leq 0.5(1 - p_1), \quad i = 1, \dots, d' \implies \\ &\implies \mathbb{E}[\|f(p) - f(q)\|_1] \leq 0.5 \cdot d' \cdot (1 - p_1), \\ d_{\mathcal{M}}(p, q) \geq cr &\implies \mathbb{E}[\|f_i(h_i(p)) - f_i(h_i(q))\|_1] \geq \\ &\geq 0.5(1 - p_2), \quad i = 1, \dots, d' \implies \\ &\implies \mathbb{E}[\|f(p) - f(q)\|_1] \geq 0.5 \cdot d' \cdot (1 - p_2). \end{aligned}$$

We distinguish two cases.

First, consider the case $d_{\mathcal{M}}(p, q) \leq r$. Let $\mu = \mathbb{E}[\|f(p) - f(q)\|_1]$. Then,

$$\Pr[\|f(p) - f(q)\|_1 \geq \mu] \leq \frac{1}{2},$$

since $\|f(p) - f(q)\|_1$ follows the binomial distribution.

Second, consider the case $d_{\mathcal{M}}(p, q) \geq cr$. By typical Chernoff bounds, $\Pr[\|f(p) - f(q)\|_1 \leq \frac{1-p_1}{1-p_2} \cdot \mu] \leq \exp(-0.5 \cdot \mu \cdot (p_1 - p_2)^2 / (1 - p_2)^2) \leq \exp(-d' \cdot (p_1 - p_2)^2 / 4(1 - p_2))$.

After mapping the query $q \in \mathcal{M}$ to $f(q)$ in the d' -dimensional Hamming space we search for all “near” Hamming vectors $f(p)$ s.t. $\|f(p) - f(q)\|_1 \leq 0.5 \cdot d' \cdot (1 - p_1)$. This search costs $\binom{d'}{1} + \binom{d'}{2} + \dots + \binom{d'}{\lfloor d' \cdot (1 - p_1) / 2 \rfloor} \leq O(d' \cdot 2^{d' \cdot H((1 - p_1) / 2)})$, where $H(\cdot)$ is the binary entropy function. The inequality is obtained from standard bounds on binomial coefficients, e.g. [28]. Now, the expected number of points $p \in P$, for which $d_{\mathcal{M}}(p, q) \geq cr$ but are mapped “near” q is $\leq n \cdot \exp(-d' \cdot (p_1 - p_2)^2 / 4(1 - p_2))$. If we set $d' = \log n$, we obtain expected query time

$$O(n^{H((1 - p_1) / 2)}) + dn^{1 - \delta},$$

where

$$\delta = \frac{(p_1 - p_2)^2}{(1 - p_2)} \cdot \frac{\lg e}{4}.$$

If we stop searching after having seen, say $10n^{1 - \delta}$ points for which $d_{\mathcal{M}}(p, q) \geq cr$, then we obtain the same time with constant probability of success. Notice that “success” translates to successful preprocessing for a fixed query $q \in \mathcal{M}$. The space required is $O(dn)$. \square

The value of δ could be somewhat larger, but we have used simplified Chernoff bounds to keep our exposition simple.

Discussion on parameters. We set the dimension $d' = \log n$ (which denotes the binary logarithm), since it minimizes the expected number of candidates under the linear space restriction. We note that it is possible to set $d' < \log n$ and still have sublinear query time. This choice of d' is interesting in practical applications since it improves space requirement. The number of candidate points is set to $n^{1 - \delta}$ for the purposes of Lemma 2.1 and under worst case assumptions on the input. In practice, this should be a user-defined parameter and hence it is denoted by the parameter *StopSearch* in Algorithm 2.

2.3 The ANN under Euclidean and Manhattan metrics

In this section, we study some classical LSH families which are also simple to implement. At first, we examine the case where the distance is the Euclidean metric and we consider two LSH families, the projection on random lines [16] and the hyperplane LSH [13]. Then, we study

Algorithm 1 Dolphinn: Preprocessing (data structure)

Input: Metric $(\mathcal{M}, d_{\mathcal{M}})$, radius $r > 0$, approximation factor $c > 1$, LSH family $F = F(c, r)$, data set $P \subset \mathcal{M}$, parameter d' .

Initialize empty hashtable T .

for $i = 1$ to d' **do**

 Sample $h_i \in F$ u.a.r.

for each $x \in h_i(P)$ **do**

 Flip a fair coin and assign the result to $f_i(x)$.

end for

end for

For all $p \in P$, $f(p) = (f_1(h_1(p)), \dots, f_{d'}(h_{d'}(p)))$.

For all $p \in P$, add p to the bucket of T with key $f(p)$.

the approximate near neighbor problem under the l_1 metric, where we consider an LSH family whose buckets correspond to cells of a randomly shifted grid [5].

2.3.1 The ℓ_2 case

In this subsection, we consider the (c, r) -ANN problem when the dataset consists of n points $P \subset \mathbb{R}^d$, the query is $q \in \mathbb{R}^d$, and the distance is the Euclidean metric.

We may assume, without loss of generality, that $r = 1$, since we can uniformly scale $(\mathbb{R}^d, \|\cdot\|_2)$.

We shall consider two LSH families, for which we obtain slightly different results. The first is based on projecting points to random lines, and it is the algorithm used in our implementation, see Section 2.4. The second family relies on reducing the Euclidean problem to points on the sphere, and then partitioning the sphere with random hyperplanes.

Project on random lines

Let p, q two points in \mathbb{R}^d and η the distance between them. Let $w > 0$ be a real parameter, and let t be a random number distributed uniformly in the interval $[0, w]$. In [16], they present the following LSH family. For $p \in \mathbb{R}^d$, consider the random function

$$h(p) = \left\lfloor \frac{\langle p, v \rangle + t}{w} \right\rfloor, \quad p, v \in \mathbb{R}^d, \quad (2.1)$$

Algorithm 2 Dolphinn: Query Algorithm

Input: Metric $(\mathcal{M}, d_{\mathcal{M}})$, LSH family F , data set $P \subset \mathcal{M}$, parameter d' , integer StopSearch , query q . (We assume that this algorithm has access to the ANN data structure created in Algorithm 1)

Output: Point $p \in P$ or "no"

for $i = 1$ to d' **do**

if $f_i(h_i(q))$ is not defined in Algorithm 1 **then**

 Flip a fair coin and assign the result to $f_i(h_i(q))$.

else

 Compute $f_i(h_i(q))$.

end if

end for

$i=0$

for each x in $f(P)$ s.t. $\|x - f(q)\|_1 \leq 0.5 \cdot d' \cdot (1 - p_1)$ **do**

for each point p inside the bucket with key x **do**

if $d_{\mathcal{M}}(p, q) \leq c \cdot r$ **then**

return p .

end if

$i \leftarrow i + 1$

if $i > \text{StopSearch}$ **then**

return "no".

end if

end for

end for

return "no"

where v is a vector randomly distributed with the d -dimensional normal distribution. This function describes the projection on a random line, where the parameter t represents the random shift and the parameter w the discretization of the line. For this LSH family, the probability of collision is

$$\alpha(\eta, w) = \int_{t=0}^w \frac{2}{\sqrt{2\pi\eta}} \exp\left(-\frac{t^2}{2\eta^2}\right) \left(1 - \frac{t}{w}\right) dt.$$

Lemma 2.2 *Given a set of n points $P \subseteq \mathbb{R}^d$, there exists a data structure for the (c, r) -ANN problem under the Euclidean metric, requiring space $O(dn)$, time preprocessing $O(dn)$, and query time $O(dn^{1-\delta} + n^{0.9})$, where*

$$\delta \geq 0.03(c - 1)^2.$$

Given some query point $q \in \mathbb{R}^d$, the building process succeeds with constant probability.

Proof. In the sequel we use the standard Gauss error function, denoted by $\text{erf}(\cdot)$. For probabilities p_1, p_2 , it holds that

$$p_1 = \alpha(1, w) = \int_{t=0}^w \frac{2}{\sqrt{2\pi}} \exp\left(-\frac{t^2}{2}\right) \left(1 - \frac{t}{w}\right) dt =$$

$$\text{erf}\left(\frac{w}{\sqrt{2}}\right) - \sqrt{\frac{2}{\pi}} \frac{1}{w} \left(1 - \exp\left(-\frac{w^2}{2}\right)\right),$$

and also that

$$p_2 = \alpha(c, w) = \int_{t=0}^w \frac{2}{\sqrt{2\pi}c} \exp\left(-\frac{t^2}{2c^2}\right) \left(1 - \frac{t}{w}\right) dt =$$

$$= \text{erf}\left(\frac{w}{\sqrt{2}c}\right) - \sqrt{\frac{2}{\pi}} \frac{c}{w} \left(1 - \exp\left(-\frac{w^2}{2c^2}\right)\right).$$

The LSH scheme is parameterized by w . One possible value is $w = 3$, as we have checked on a computer algebra system. On the other hand, $w = c$ gives similar results, and they are simpler to obtain. In particular, we have

$$p_1 - p_2 = \text{erf}\left(\frac{c}{\sqrt{2}}\right) - \sqrt{\frac{2}{\pi}} \frac{1}{c} \left(1 - \exp\left(-\frac{c^2}{2}\right)\right) -$$

$$- \text{erf}\left(\frac{1}{\sqrt{2}}\right) + \sqrt{\frac{2}{\pi}} \left(1 - \exp\left(-\frac{1}{2}\right)\right).$$

We shall prove that, given $w = c$, for $c \in (1, 2]$, it holds that $p_1 - p_2 > \frac{5(c-1)}{21}$. Let us define

$$g(c) = p_1 - p_2 - \frac{5(c-1)}{21} = \text{erf}\left(\frac{c}{\sqrt{2}}\right) - \text{erf}\left(\frac{1}{\sqrt{2}}\right) -$$

$$- \sqrt{\frac{2}{\pi}} \frac{1}{c} \left(1 - \exp\left(-\frac{c^2}{2}\right)\right) + \sqrt{\frac{2}{\pi}} \left(1 - \exp\left(-\frac{1}{2}\right)\right) - \frac{5(c-1)}{21},$$

$c \in (1, 2]$. Using elementary calculus, it is easy to show that $g(c)$ is concave over $c \in (1, 2]$. Also, $g(1) = 0$ and $g(2) > 0$, thus $\forall c \in (1, 2]$, $g(c) > 0$ and consequently $p_1 - p_2 > \frac{5(c-1)}{21}$. In addition, $w = c$ implies $1 - p_2 = 1 - \text{erf}\left(\frac{1}{\sqrt{2}}\right) + \sqrt{\frac{2}{\pi}} \left(1 - \exp\left(-\frac{1}{2}\right)\right) < 0.64$, and $H\left(\frac{1-p_1}{2}\right) < 0.9$. Hence, for $w = c$ and $c \in (1, 2]$, $\delta > 0.03(c-1)^2$. \square

Hyperplane LSH

This section reduces the Euclidean ANN to an instance of ANN for which the points lie on a unit sphere. The latter admits an LSH scheme based on partitioning the space by randomly selected halfspaces.

In Euclidean space \mathbb{R}^d , let us assume that the dimension is $d = O(\log n \cdot \log \log n)$, since one can project points à la Johnson-Lindenstrauss [15], and preserve pairwise distances up to multiplicative factors of $1 \pm o(1)$. Then, we partition \mathbb{R}^d using a randomly shifted grid, with cell edge of length $O(\sqrt{d}) = O((\log n \cdot \log \log n)^{1/2})$. Any two points $p, q \in \mathbb{R}^d$ for which $\|p - q\|_2 \leq 1$ lie in the same cell with constant probability. Let us focus on the set of points lying inside one cell. This set of points has diameter bounded by $O((\log n \cdot \log \log n)^{1/2})$. Now, a reduction of [34], reduces the problem to an instance of ANN for which all points lie on a unit sphere \mathbb{S}^{d-1} , and the search radius is roughly $r' = \Theta((\log n \cdot \log \log n)^{-1/2})$. These steps have been also used in [8], as a data-independent reduction to the spherical instance.

Let us now consider the LSH family introduced in [13]. Given n unit vectors $P \subset \mathbb{S}^{d-1}$, we define, for each $q \in \mathbb{S}^{d-1}$, hash function $h(q) = \text{sign}\langle q, v \rangle$, where v is a random unit vector. Obviously, $\Pr[h(p) = h(q)] = 1 - \frac{\theta(p, q)}{\pi}$, where $\theta(p, q)$ denotes the angle formed by the vectors $p \neq q \in \mathbb{S}^{d-1}$. Instead of directly using the family of [13], we employ its amplified version, obtained by concatenating $k \approx 1/r'$ functions $h(\cdot)$, each chosen independently and uniformly at random from the underlying family. The amplified function $g(\cdot)$ shall be fully defined in the proof below. This procedure leads to the following.

Lemma 2.3 *Given a set of n points $P \subset \mathbb{R}^d$, there exists a data structure for the (c, r) -ANN problem under the Euclidean metric, requiring space $O(dn)$, time preprocessing $O(dn)$, and query time $O(dn^{1-\delta} + n^{0.91})$, where*

$$\delta \geq 0.05 \cdot \left(\frac{c-1}{c}\right)^2.$$

Given some query $q \in \mathbb{R}^d$, the building process succeeds with constant probability.

Proof. We exploit the reduction described above that translates the Euclidean ANN to a spherical instance of ANN with search radius $r' = \Theta((\log n \cdot \log \log n)^{-1/2})$. The latter is

handled by a hyperplane LSH scheme based on [13] as detailed immediately below.

Let us denote by F the aforementioned LSH family of [13]. We build a new (amplified) family of functions $G_k = \{g(x) = (h_1(x), \dots, h_k(x)) : i = 1 \dots k, h_i \in F\}$. Now, obviously, for any two unit vectors $p \neq q$, we have

$$\Pr_{g \in G}[g(p) = g(q)] = \left(1 - \frac{\theta(p, q)}{\pi}\right)^k.$$

Hence, $\|p - q\|_2 \leq r' \implies 2 \sin\left(\frac{\theta(p, q)}{2}\right) \leq r' \implies \theta(p, q) \leq 2 \arcsin\left(\frac{r'}{2}\right) = \theta_r$, which defines θ_r .

Moreover, $\|p - q\|_2 \geq cr' \implies 2 \sin\left(\frac{\theta(p, q)}{2}\right) \geq cr' \implies \theta(p, q) \geq 2 \arcsin\left(\frac{cr'}{2}\right)$.

By using elementary calculus, it is easy to prove that $2 \arcsin\left(\frac{cr'}{2}\right) \geq 2c \cdot \arcsin\left(\frac{r'}{2}\right) \implies \theta(p, q) \geq c \cdot \theta_r$. Hence, for $k = \lceil \pi/\theta_r \rceil$ and since $r' = \Theta((\log n \cdot \log \log n)^{-1/2}) \implies \theta_r = o(1)$,

$$p_1 = \Pr[g(p) = g(q) \mid \|p - q\|_2 \leq r] \geq \left(1 - \frac{\theta_r}{\pi}\right)^k \geq$$

$$\geq \exp\left(-\frac{\pi}{\pi - \theta_r}\right) \geq \frac{1}{e^{1+o(1)}},$$

$$p_2 = \Pr[g(p) = g(q) \mid \|p - q\|_2 \geq c \cdot r] \leq \left(1 - \frac{c \cdot \theta_r}{\pi}\right)^k \leq$$

$$\leq \exp\left(-\frac{c\theta_r}{\pi} \cdot \left(\frac{\pi}{\theta_r} - 1\right)\right) \leq \frac{1}{c \cdot e^{1-o(1)}}.$$

Now applying Lemma 2.1 yields

$$\delta \geq \frac{1}{e^{2+o(1)}} \cdot \left(1 - \frac{e^{o(1)}}{c}\right)^2 \cdot \frac{1}{1 - (c \cdot e)^{-1}} \cdot \frac{\log(e)}{4} \geq$$

$$\geq 0.059 \cdot \left(1 - \frac{1}{c}\right)^2, \quad \text{for } c \in (1, 2].$$

The space required is $O(dn + n(d' + k))$ where d' is defined in Lemma 2.1. Since $k \ll d$, and $d' \ll d$ the total space is $O(dn)$. Notice also that $H\left(\frac{1-p_1}{2}\right) \leq 0.91$. \square

The data structure of Lemma 2.3 provides slightly better query time than that of Lemma 2.2, when c is small enough.

2.3.2 The ℓ_1 case

In this section, we study the (c, r) -ANN problem under the ℓ_1 metric. The dataset consists again of n points $P \subset \mathbb{R}^d$ and the query point is $q \in \mathbb{R}^d$.

For this case, let us consider the following LSH family, introduced in [5]. A point p is hashed as follows:

$$h(p) = \left(\left\lfloor \frac{p_1 + t_1}{w} \right\rfloor, \left\lfloor \frac{p_2 + t_2}{w} \right\rfloor, \dots, \left\lfloor \frac{p_d + t_d}{w} \right\rfloor \right),$$

where $p = (p_1, p_2, \dots, p_d)$ is a point in P , $w = \alpha r$, and the t_i are drawn uniformly at random from $[0, \dots, w)$. Buckets correspond to cells of a randomly shifted grid.

Now, in order to obtain a better lower bound, we employ an amplified hash function, defined by concatenation of $k = \alpha$ functions $h(\cdot)$ chosen uniformly at random from the above family.

Lemma 2.4 *Given a set of n points $P \subseteq \mathbb{R}^d$, there exists a data structure for the (c, r) -ANN problem under the ℓ_1 metric, requiring space $O(dn)$, time preprocessing $O(dn)$, and query time $O(dn^{1-\delta} + n^{0.91})$, where*

$$\delta \geq 0.05 \cdot \left(\frac{c-1}{c} \right)^2.$$

Given some query point $q \in \mathbb{R}^d$, the building process succeeds with constant probability.

Proof. We denote by F the previously introduced LSH family of [5], which is $(1 - \frac{1}{\alpha}, 1 - \frac{c}{c+\alpha}, 1, c)$ -sensitive. We build the amplified family of functions $G_k = \{g(x) = (h_1(x), \dots, h_k(x)) : i = 1, \dots, k, h_i \in F\}$. Setting $\alpha = k = \log n$, we have:

$$\begin{aligned} p_1 &= \left(1 - \frac{1}{\alpha}\right)^k = \left(1 - \frac{1}{\log n}\right)^{\log n} \geq \\ &\geq \left(\exp\left(-\frac{1}{\log n - 1}\right)\right)^{\log n} \geq \frac{1}{e^{1+o(1)}}, \\ p_2 &= \left(1 - \frac{c}{\alpha + c}\right)^k = \left(1 - \frac{c}{\log n + c}\right)^{\log n}. \end{aligned}$$

Hence,

$$p_2 \geq \exp(-c) \geq \frac{1}{e \cdot (2c - 1)},$$

and

$$\begin{aligned} p_2 &\leq \exp\left(-\frac{c}{1+\frac{c}{\log n}}\right) = \exp\left(-\frac{c}{1+o(1)}\right) \leq \\ &\leq \exp(-c+o(1)) \leq \frac{e^{o(1)}}{ec}. \end{aligned}$$

Therefore, for n large enough, it holds that

$$\begin{aligned} \delta &= \frac{(p_1 - p_2)^2}{(1 - p_2)} \cdot \frac{\log e}{4} \geq \frac{1}{e^{2+o(1)}} \cdot \frac{(1 - \frac{1}{c})^2}{1 - \frac{1}{e(2c-1)}} \cdot \frac{\log e}{4} \geq \\ &\geq 0.055 \cdot (1 - \frac{1}{c})^2, \quad \text{for } c \in (1, 2]. \end{aligned}$$

Notice that $H((1 - p_1)/2) \leq 0.91$. □

2.4 Implementation and experimental results

The algorithm presented has been implemented. Our C++ library, named `Dolphinn` is available online ¹. The project is open source, under the BSD 2-clause license.

Our implementation is based on the algorithm from Subsection 2.3.1 and supports similarity search under the Euclidean metric. We compare `Dolphinn` with the state-of-the-art LSH-based `FALCONN`² library, and the straightforward brute force approach. More information about the parameters of the implementation are available in [12], where important implementation issues focusing on efficiency are discussed and the experimental results are thoroughly analyzed.

Lets summarize the experimental results in order to illustrate the contribution of our implementation. We examine build and search times, as well as memory consumption. `Dolphinn` and `FALCONN`, both, have equal memory consumption. The preprocessing time of `Dolphinn` has a linear dependence in n and d , as expected. Moreover, `Dolphinn` is observed to be competitive with `FALCONN` when employing a specific LSH family. In general, our algorithm is significantly faster than brute force as expected, and scales well, namely sublinearly in n and linearly in d . Last but not least, `Dolphinn` either outperforms or has comparable performance with the implementation of `FALCONN`, regarding query times.

¹<https://github.com/Anonymous-ICML/Dolphinn>

²<https://falconn-lib.org/>

Chapter 3

High-dimensional approximate r -nets

3.1 Introduction

In this section, we present a new randomized algorithm that computes approximate r -nets in time subquadratic in n and polynomial in D , and improves upon the complexity of the best known algorithm. Our method does not employ LSH and, with probability $1 - o(1)$, returns $R \subseteq X$, which is a $(1 + \epsilon)r$ -net of X .

We reduce the problem of an approximate r -net for arbitrary vectors (points) under Euclidean distance to the same problem for vectors on the unit sphere. Then, depending on the magnitude of distance r , either an algorithm handling “small” distances or an algorithm handling “large” distances is called. These algorithms reduce the Euclidean problem of r -nets on unit vectors to finding an r -net for unit vectors under inner product (Section 3.3). This step requires that the multiplicative $1 + \epsilon$ approximation of the distance corresponds to an additive $c\epsilon$ approximation of the inner product, for suitable constant $c > 0$.

We convert the vectors having unit norm into vectors with entries $\{-1, +1\}$ (Section 3.2). This transformation is necessary in order to apply the Chebyshev embedding of [34], an embedding that damps the magnitude of the inner product of “far” vectors, while preserving the magnitude of the inner product of “close” vectors. For the final step of the algorithm, we first apply a procedure that allows us to efficiently compute $(1 + \epsilon)$ -nets when the number of “small” distances is large. Then, we apply a modified version of the **Vector Aggregation** algorithm of [34] that

exploits fast matrix multiplication in order to achieve the desired running time.

In short, we extend Valiant’s framework [34] and compute r -nets in time $\tilde{O}(dn^{2-\Theta(\sqrt{\epsilon})})$, thus improving on the exponent of the LSH-based construction in [17], when ϵ is sufficiently small. This improvement by $\sqrt{\epsilon}$ in the exponent is the same as the complexity improvement obtained in [34] over the LSH-based algorithm for the approximate closest-pair problem.

Our study is motivated by the observation that computing efficiently an r -net leads to efficient solutions for several geometric problems, specifically by means of approximation algorithms. In particular, our extension of r -nets in high dimensional Euclidean space can serve in the framework of [22]. This framework has many applications, notably to computing the distance to the k -th nearest neighbor, for which we obtain a $1 + \epsilon$ approximation, in time $\tilde{O}(dn^{2-\Theta(\sqrt{\epsilon})})$. Other applications may include preprocessing for finding the approximate nearest neighbor to ANN, see e.g. [3].

This work offers a complete version of our results, initially presented in [11], including full proofs and discussion. Section 3.2 presents an algorithm for computing an approximate net with respect to the inner product for a set of unit vectors. Section 3.3 translates the problem of finding r -nets under Euclidean distance to the same problem under inner product. In Section 3.4, we discuss applications of our construction and possible future work.

Throughout this work, we use $\|\cdot\|$ to denote the Euclidean norm $\|\cdot\|_2$.

3.2 Points on a sphere under inner product

In this section, we design an algorithm for constructing an approximate ρ -net of vectors on the sphere under inner product. To that end, we reduce the problem to constructing an approximate net under absolute inner product for vectors that lie on the vertices of a unit hypercube.

Since our ultimate goal is a solution to computing r -nets with respect to Euclidean distance, we allow additive error in the approximation, which under certain assumptions, translates to multiplicative error in Euclidean distance. In the following, we define rigorously the notion of approximate ρ -nets under inner product.

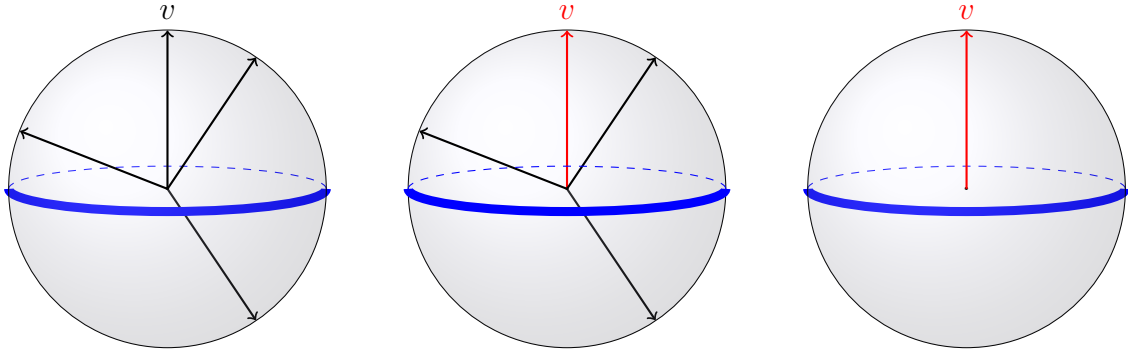


Figure 3.1: The red vector is ρ -correlated with the black vectors, but is not ρ -correlated with the blue points on the equator, since it is almost orthogonal to them.

Definition 3.1 For any $X \subset \mathbb{S}^{d-1}$, an approximate ρ -net for $(X, \langle \cdot, \cdot \rangle)$, with additive approximation parameter $\epsilon > 0$, is a subset $C \subseteq X$ which satisfies the following properties:

- for any two $p \neq q \in C$, $\langle p, q \rangle < \rho$, and
- for any $x \in X$, there exists $p \in C$ s.t. $\langle x, p \rangle \geq \rho - \epsilon$.

One relevant notion is that of ϵ -kernels [1]. In ϵ -kernels, one is interested in finding a subset of the input pointset, which approximates its directional width. Such constructions have been extensively studied when the dimension is low, due to their relatively small size.

3.2.1 Crude approximate nets.

In this subsection we develop our basic tool, which is based on the Vector Aggregation Algorithm by [34]. This tool aims to compute approximate ρ -nets with multiplicative error, in contrast with our final goal for this section, namely to bound additive error. Moreover, in the context of this subsection, two vectors are close to each other when the magnitude of their inner product is large, and two vectors are far from each other when the magnitude of their inner product is small. Let $|\langle \cdot, \cdot \rangle|$ denote the magnitude of the inner product of two vectors.

Definition 3.2 For any $X = [x_1, \dots, x_n], X' = [x'_1, \dots, x'_n] \subset \mathbb{R}^{d \times n}$, a crude approximate ρ -net for $(X, X', |\langle \cdot, \cdot \rangle|)$, with multiplicative approximation factor $c > 1$, is a subset $C \subseteq [n]$ which satisfies the following properties:

- for any two $i \neq j \in C$, $|\langle x_i, x'_j \rangle| < c\rho$, and
- for any $i \in [n]$, there exists $j \in C$ s.t. $|\langle x_i, x'_j \rangle| \geq \rho$.

In Figure 3.1, we illustrate the notion of ρ -correlated vectors. **Vector Aggregation** follows the exposition of [34]. The main difference is that, instead of the “compressed” matrix $Z^T Z$, we use the form $X^T Z$, where Z derives from vector aggregation. Both forms encode the information in the Gram matrix $X^T X$. The matrix $X^T Z$ is better suited for our purposes, since each row corresponds to an input vector instead of an aggregated subset; this extra information may be useful in further problems.

Vector Aggregation

Input: $X = [x_1, \dots, x_n] \in \mathbb{R}^{d \times n}$, $X' = [x'_1, \dots, x'_n] \in \mathbb{R}^{d \times n}$, $\alpha \in (0, 1)$, $\tau > 0$.

Output: $n \times n^{1-\alpha}$ matrix W and random partition $S_1, \dots, S_{n^{1-\alpha}}$ of $\{x'_1, \dots, x'_n\}$.

- Randomly partition $\{x'_1, \dots, x'_n\}$ into $n^{1-\alpha}$ disjoint subsets, each of size n^α , denoting the sets $S_1, \dots, S_{n^{1-\alpha}}$.
- For each $i = 1, 2, \dots, 78 \log n$:
 - Select n coefficients $q_1, \dots, q_n \in \{-1, +1\}$ at random.
 - Form the $d \times n^{1-\alpha}$ matrix Z^i with entries $z_{j,k}^i = \sum_{l: x'_l \in S_k} q_l \cdot x'_{j,l}$, where $x'_{j,l}$ are entries of X' .
 - $W^i = X^T Z^i$.
- Define the $n \times n^{1-\alpha}$ matrix W with $w_{i,j} = \text{quartile}(|w_{i,j}^1|, \dots, |w_{i,j}^{78 \log n}|)$ (quartile is the smallest value greater than the lowest 75% of the entries).
- Output W and $S_1, \dots, S_{n^{1-\alpha}}$.

Theorem 3.1 *Let $X = [x_1, \dots, x_n] \in \mathbb{R}^{d \times n}$, $X' = [x'_1, \dots, x'_n] \in \mathbb{R}^{d \times n}$, $\alpha \in (0, 1)$, $\tau > 0$ the input of Vector Aggregation. Then, the algorithm returns a matrix W of size $n \times n^{1-\alpha}$ and*

a random partition $S_1, \dots, S_{n^{1-\alpha}}$ of $\{x'_1, \dots, x'_n\}$, which with probability $1 - O(1/n^3)$ satisfies the following:

- For all $j \in [n]$ and $k \in [n^{1-\alpha}]$, if $\forall u \in S_k$, $|\langle x_j, u \rangle| \leq \tau$ then $|w_{j,k}| < 3 \cdot n^\alpha \tau$.
- For all $j \in [n]$ and $k \in [n^{1-\alpha}]$ if $\exists u \in S_k$, $|\langle x_j, u \rangle| \geq 3n^\alpha \tau$ then $|w_{j,k}| \geq 3 \cdot n^\alpha \tau$.

Moreover, the algorithm runs in time $\tilde{O}(dn + n^{2-\alpha} + \text{MatrixMul}(n \times d, d \times n^{1-\alpha}))$.

The following anti-concentration Lemma is crucial for the proof of Theorem 3.1, since it argues that if an entry of W^i contains a contribution from a pair of columns with large inner product, then with reasonable probability over the random choice of q_1, \dots, q_n this entry will not be too small. In other words, after applying **Vector Aggregation** algorithm, we can still distinguish pairs of vectors which were initially correlated from pairs of vectors which were uncorrelated in the first place. The proof of the Lemma shows that by randomly flipping the signs of each vector we can achieve pairwise independence between the inner products of different pairs of vectors, thus roughly preserving the magnitude of inner products between vectors after the aggregation step.

Lemma 3.2 (Anti-concentration) *Let $q_1, \dots, q_t \in \{-1, 1\}$ be chosen independently and uniformly at random, and let $a_1, \dots, a_t \in \mathbb{R}$ s.t. $|a_1| = \max_i |a_i|$. Then,*

$$\Pr\left[\left|\sum_{i=1}^t q_i \cdot a_i\right| \geq |a_1|\right] \geq 1/2.$$

Proof. Consider a given assignment for q_2, \dots, q_t . Then if

$$\sum_{i=2}^t q_i \cdot a_i = 0 \implies \left|\sum_{i=1}^t q_i \cdot a_i\right| = |q_1 \cdot a_1| = |a_1|.$$

Otherwise,

$$\Pr\left[\left|\sum_{i=1}^t q_i \cdot a_i\right| \geq |a_1|\right] \geq$$

$$\geq \Pr[\text{sign}(q_1 \cdot a_1) = \text{sign}(\sum_{i=2}^t q_i \cdot a_i = 0)] = 1/2.$$

□

Proof of Theorem 3.1. Notice that

$$w_{j,k}^i = \sum_{x'_i \in S_k} q_i \cdot \langle x_j, x'_i \rangle$$

and since $q_1, \dots, q_{|S_k|} \in \{-1, 1\}$ are independent and chosen uniformly at random, we obtain

$$\mathbb{E}[w_{j,k}^i] = 0.$$

If $\forall u \in S_k, |\langle x_j, u \rangle| \leq \tau$, then

$$\text{Var}(w_{j,k}^i) = \mathbb{E}[(w_{j,k}^i)^2] \leq n^{2\alpha} \tau^2$$

By Chebyshev's inequality:

$$\Pr[|w_{j,k}^i| \geq 3 \cdot n^\alpha \tau] \leq 1/9$$

With m repetitions, the number of successes N , that is the number of indices i for which $|w_{j,k}^i| \leq 3 \cdot n^\alpha \tau$, follows the binomial distribution. Hence,

$$\Pr[N \leq 3m/4] \leq \exp(-m/26).$$

We consider as bad event the event that for some j, k , more than 25% of the repetitions fail, that is $|w_{j,k}^i| \geq 3 \cdot n^\alpha \tau$. By the union bound, this probability is $\leq n^{2-\alpha} \cdot \exp(-m/26)$, which for $m \geq 78 \log n$ implies a probability of failure $\leq 1/n^3$.

Now consider x_j , and $x'_l \in S_k$ s.t. $|\langle x_j, x'_l \rangle| \geq 3 \cdot n^\alpha \tau$, then by Lemma 3.2, with probability $1/2$, $|w_{j,k}^i| \geq 3 \cdot n^\alpha \tau$. We consider as bad event the event that for j, l , more than 75% of the repetitions fail, that is $|w_{j,k}^i| \leq 3 \cdot n^\alpha \tau$. Hence,

$$\Pr[N \leq m/4] \leq \exp(-m/8),$$

which for $m \geq 78 \log n$ implies a probability of failure $\leq 1/n^3$.

The runtime of the algorithm is dominated, up to polylogarithmic factors, by the computation of matrix Z , taking time $O(dn)$, the computation of matrix W , taking time n^{2-a} , or the computation of the product W^i , taking time $MatrixMul(n \times d, d \times n^{1-a})$. \square

For the case of pointsets with many “small” distances, we rely crucially on the fact that the expected number of near neighbors for a randomly chosen point is large. So, if we iteratively choose random points and delete these and their neighbors, we will end up with a pointset which satisfies the property of having sufficiently few “small” distances. Then, we apply **Vector Aggregation**.

Crude ApprxNet

Input: $X = [x_1, \dots, x_n] \in \mathbb{R}^{d \times n}$, $X' = [x'_1, \dots, x'_n] \in \mathbb{R}^{d \times n}$, $\alpha \in (0, 1)$, $\tau > 0$.

Output: $C' \subseteq [n]$, $F' \subseteq [n]$.

- $C \leftarrow \emptyset$, $F_1 \leftarrow \emptyset$, $F_2 \leftarrow \{x_1, \dots, x_n\}$
- Repeat $n^{0.5}$ times:
 - Choose a column x_i uniformly at random.
 - $C \leftarrow C \cup \{x_i\}$.
 - Delete column i from matrix X and column i from matrix X' .
 - Delete each column k from matrix X , X' s.t. $|\langle x_i, x'_k \rangle| \geq \tau$.
 - If there is no column k from matrix X s.t. $|\langle x_i, x'_k \rangle| \geq \tau$, then $F_1 \leftarrow F_1 \cup \{x_i\}$
- Run **Vector Aggregation** with input X , X' , α , τ and output W , $S_1, \dots, S_{n^{1-\alpha}}$.
- For each of the remaining rows $i = 1, \dots$:
 - For any $|w_{i,j}| \geq 3n^\alpha \tau$:
 - * If more than $n^{1.7}$ times in here, output "ERROR".

- * Compute inner products between x_i and vectors in S_j . For each vector $x'_k \in S_j$ s.t. $k \neq i$ and $|\langle x_i, x'_k \rangle| \geq \tau$, delete row k and $F_2 \leftarrow F_2 \setminus \{x_i\}$.
- $C \leftarrow C \cup \{x_i\}$
- Output indices of C and $F \leftarrow \{F_1 \cup F_2\}$.

Theorem 3.3 *On input $X = [x_1, \dots, x_n] \in \mathbb{R}^{d \times n}$, $X' = [x'_1, \dots, x'_n] \in \mathbb{R}^{d \times n}$, $\alpha \in (0, 1)$, $\tau > 0$, Crude ApprxNet, computes a crude $3n^\alpha$ -approximate τ -net for X, X' , following the notation of Definition 3.2. The algorithm costs time:*

$$\tilde{O}(n^{2-\alpha} + d \cdot n^{1.7+\alpha} + \text{MatrixMul}(n \times d, d \times n^{1-\alpha})),$$

and succeeds with probability $1 - O(1/n^{0.2})$. Additionally, it outputs a set $F \subseteq \{x_1, \dots, x_n\}$ with the following property: $\{x_i \mid \forall j \neq i \ |\langle x'_j, x_i \rangle| < \tau\} \subseteq F \subseteq \{x_i \mid \forall j \neq i \ |\langle x'_j, x_i \rangle| < 3n^\alpha \tau\}$.

Proof. We perform $n^{0.5}$ iterations and for each, we compare the inner products between the randomly chosen vector and all other vectors. Hence, the time needed is $O(dn^{1.5})$.

In the following, we denote by X_i the number of vectors which have “large” magnitude of the inner product with the randomly chosen point in the i th iteration. Towards proving correctness, suppose first that $\mathbb{E}[X_i] > 2n^{0.5}$ for all $i = 1, \dots, n^{0.5}$. The expected number of vectors we delete in each iteration of the algorithm is more than $2n^{0.5} + 1$. So, after $n^{0.5}$ iterations, the expected total number of deleted vectors will be greater than n . This means that if the hypothesis holds for all iterations we will end up with a proper net.

Now suppose that there is an iteration j where $\mathbb{E}[X_j] \leq 2n^{0.5}$. After all iterations, the number of “small” distances are at most $n^{1.5}$ on expectation. By Markov’s inequality, when the Vector Aggregation algorithm is called, the following is satisfied with probability $1 - n^{-0.2}$:

$$|\{(i, k) \mid |\langle x_i, x'_k \rangle| \geq \tau, i \neq k\}| \leq n^{1.7}.$$

By Theorem 3.1 and the above discussion, the number of entries in the matrix W that we need

to visit is at most $n^{1.7}$. For each entry, we perform brute force which costs dn^α .

Now notice that the first iteration stores centers c and deletes all points p for which $|\langle c, p \rangle| \geq \tau$. Hence, any two centers c, c' satisfy $|\langle c, c' \rangle| < \tau$. In the second iteration, over the columns of W , notice that by Theorem 3.1, for any two centers c, c' we have $|\langle c, c' \rangle| < 3n^\alpha \tau$. \square

Notice that the constants in the analysis above was chosen to facilitate our purposes. No further attempts were made towards optimization.

3.2.2 Approximate inner product nets.

In this subsection, we show that the problem of computing ρ -nets for the inner product of unit vectors reduces to the less natural problem of Definition 3.2, which refers to the magnitude of the inner product.

The first step consists of mapping the unit vectors to vectors in $\{-1, 1\}^{d'}$. The mapping is essentially Charikar's LSH scheme [13] and we include it for reasons of completeness (**Make Uniform**). Then, we apply the Chebyshev embedding of [34] in order to achieve gap amplification, and finally we call algorithm **Crude ApprxNet**, which will now return a proper ρ -net with additive error.

Make Uniform

Input: An $d \times n$ matrix X with entries $x_{i,j} \in \mathbb{R}$ whose columns have unit Euclidean norm and $\delta \in (0, 1)$.

Output: An $d' \times n$ matrix Y with entries $y_{i,j} \in \{\pm 1\}$, where $d' = \frac{10 \log n}{\delta^2}$.

- For each $i = 1, \dots, d'$, select a random unit vector $u \in \mathbb{R}^d$, and let $w = u^t X$. For all $j = 1, \dots, n$ set $y_{i,j} = \text{sign}(w_j)$.

The following theorem describes the performance of the algorithm that converts the set of unit vectors into the set of vectors with entries in $\{-1, 1\}^{d'}$. This transformation is necessary in order to apply Chebyshev embedding and ensures that the entries of the vectors returned by the

embedding have the same magnitudes. Then, we can apply Chernoff bounds to guarantee that the inner products between the returned vectors are concentrated about their expectations.

Theorem 3.4 [34] *There exists an algorithm, **Make Uniform**, with the following properties. Let $d' = O(\frac{\log n}{\delta^2})$ and $Y \in \mathbb{R}^{d' \times n}$ denote its output on input X , and δ , where X is a matrix whose columns have unit norm. Then, with probability $1 - o(1/n^2)$, for all pairs $i, j \in [n]$, the following difference has an absolute value bounded as follows:*

$$\left| \frac{\langle Y_i, Y_j \rangle}{d'} - 1 + 2 \cdot \frac{\cos^{-1}(\langle X_i, X_j \rangle)}{\pi} \right| \leq \delta,$$

where X_i, Y_i denote the i -th column of X and Y respectively. Additionally, the runtime of the algorithm is $O(dn \log n / \delta^2)$.

Next, we present the theorem describing the performance of Chebyshev embedding, while we include the algorithm **Chebyshev Embedding** to facilitate the reader; a randomized embedding that reduces the magnitude of the inner product between “far away” vectors, while preserving the magnitude of the inner product between “close” vectors. The statement is taken from [34, Prop.6], except that we additionally establish an asymptotically better probability of success. Before proceeding to the proof, we define the Chebyshev polynomial of the first kind. The proof is the same, but since we claim stronger guarantees on success probability, we include the complete proof.

Definition 3.3 *Let*

$$T_q(x) = \frac{(x - \sqrt{x^2 - 1})^q + (x + \sqrt{x^2 - 1})^q}{2},$$

be the q th Chebyshev polynomial of the first kind.

Chebyshev Embedding

Input: Two $d \times n$ matrices X, X' with entries $x_{i,j} \in \{\pm 1\}$, real numbers $\tau^-, \tau^+ \in [-1, 1]$ with $\tau^- < \tau^+$ and integers q and d' .

Output: Two $d' \times n$ matrices Y, Y' with entries $y_{i,j} \in \{\pm 1\}$.

- Let T_q denote the degree q Chebyshev polynomial (of the first kind), with roots at $r_1, \dots, r_q \in (0, 1)$.
- Each of the d' rows of the output matrices Y, Z are populated as follows:
 - We will populate two sets of q vectors s_1, \dots, s_q , and t_1, \dots, t_q each of length n as follows. For $i = 1, \dots, q$:
 - * With probability $1/2$, choose a random index $j \in [d]$ and set both s_i to be the j th row of X and t_i to be the j th row of X' .
 - * Let $c_i = \tau^- + \frac{1+r_i}{2}(\tau^+ - \tau^-)$ be the location of the i th root of T_q after the support has been scaled so that the roots lie within $[\tau^-, \tau^+]$ rather than $[1, 1]$.
 - * With probability $\frac{1-c_i}{4}$ set both s_i and t_i to be the all ones vectors.
 - * With probability $\frac{1+c_i}{4}$ set s_i to be the all ones vector, and t_i to be the all minus ones vector.
 - Define the i th rows of Y and Y' to be the component-wise products of the s_i s and t_i s respectively:

$$Y_{i,j} = \prod_{l=1}^q q s_l(j), \quad Y'_{i,j} = \prod_{l=1}^q q t_l(j),$$

where $s_l(j)$ and $t_l(j)$ denote the j th entries of the vectors s_l and t_l , respectively.

Theorem 3.5 *Let Y, Y' be the matrices output by algorithm Chebyshev Embedding on input $X, X' \in \{-1, 1\}^{d \times n}$, $\tau^+ \in [-1, 1], \tau^- \in [-1, 1]$ with $\tau^- < \tau^+$, integers q, d' . With probability $1 - o(1/n)$ over the randomness in the construction of Y, Y' , for all $i, j \in [n]$, $\langle Y_i, Y'_j \rangle$ is within $\sqrt{d'} \log n$ from the value*

$$T_q \left(2 \frac{\langle X_i, X'_j \rangle / d' - \tau^-}{\tau^+ - \tau^-} - 1 \right) \cdot d' \cdot \frac{(\tau^+ - \tau^-)^q}{2^{3q-1}},$$

where T_q is the degree- q Chebyshev polynomial of the first kind. The algorithm runs in time $O(d' \cdot n \cdot q)$.

For the proof of Theorem 3.5 we refer to [34, Algorithm 3: Chebyshev Embedding], which we included for completeness. The proof is the same with that of [34], apart from indicating that the probability of success is actually $1 - o(1/n)$ instead of $1 - o(1)$ as stated in [34]. While $1 - o(1/n)$ probability of success is enough for our purposes, even better probability bounds can be achieved.

Proof. The fact that all inner products are concentrated within $\pm\sqrt{d}\log n$ about their expectations follows from the fact that each row of Y, Y' is generated identically and independently from the other rows, and all entries of these matrices are ± 1 ; thus, each inner product is a sum of independent and identically distributed random ± 1 random variables, and we can apply the basic Chernoff bound to each inner product, and then a union bound over the $O(n^2)$ inner products. Let $X_i \in \pm 1$ i.i.d. random variables. The basic chernoff bound gives probability,

$$\begin{aligned} \Pr\left[\left|\sum_{i=1}^{d'} X_i - \mathbb{E}\left[\sum_{i=1}^{d'} X_i\right]\right| > \sqrt{d'} \log n\right] &\leq \\ &\leq 2 \cdot \exp(-\Theta(\log^2 n)) = o(1/n^3). \end{aligned}$$

Given this concentration, we now analyze the expectation of the inner products. Let u, u' be columns of X, X' , respectively, and v, v' the corresponding columns of Y, Y' . Letting $x = \langle u, u' \rangle / d$, we argue that by [34, Lemma 3.3], $\mathbb{E}[v, v'] = d' \sum_{i=1}^q \frac{x - c_i}{2}$ (1), where c_i is the location of the i th root of the q th Chebyshev polynomial after the roots have been scaled to lie in the interval $[\tau^-, \tau^+]$. To see why this is the case, note that each coordinate of u, u' , is generated by computing the product of q random variables that are all ± 1 ; namely, a given entry of u is given by $\prod_{l=1}^q s_v(l)$, with the corresponding entry of u' given by $\prod_{l=1}^q s_{v'}(l)$. Note that for $i \neq j$, $s_v(i)$ is independent of $s_v(j)$ and $t_{v'}(j)$, although by construction, $s_v(i)$ and $t_{v'}(i)$ are not independent. We now argue that $\mathbb{E}[s_v(i)t_{v'}(i)] = \frac{x - c_i}{2}$, from which Eq. (1) will follow by the fact that the expectation of the product of independent random variables is the product of their expectations.

By construction, in Step (1) of the inner loop of the algorithm, with probability $1/2$, $\mathbb{E}[s_v(i)t_{v'}(i)] = \langle v, v' \rangle / d = x$. Steps (2)(4) ensure that with the remaining $1/2$ probability, $\mathbb{E}[s_v(i)t_{v'}(i)] = \frac{1 - c_i}{2}(1) - \frac{1 + c_i}{2}(-1) = -c_i$. Hence, in aggregate over the randomness of Steps (1)(4), $\mathbb{E}[s_v(i)t_{v'}(i)] =$

$x/2 - c_i/2i$, as claimed, establishing Eq. (1).

To show that Eq. (1) yields the statement of the proposition, we simply reexpress the polynomial $\prod_{i=1}^q \frac{x-c_i}{2}$ in terms of the q th Chebyshev polynomial T_q . Note that the q th Chebyshev polynomial has leading coefficient 2^{q-1} , whereas this expression (as a polynomial in x) has leading coefficient $1/2^q$, disregarding the factor of the dimension m' . If one has two monic degree q polynomials, P and Q where the roots of Q are given by scaling the roots of P by a factor of α , then the values at corresponding locations differ by a multiplicative factor of $1/\alpha^q$; since the roots of T_q lie between $[-1, 1]$ and the roots of the polynomial constructed in the embedding lie between $[\tau^-, \tau^+]$, this corresponds to taking $\alpha = \frac{2}{\tau^+ - \tau^-}$. \square

Inner product ApprxNet

Input: $X = [x_1, \dots, x_n]$ with each $x_i \in \mathbb{S}^{d-1}$, $\rho \in [-1, 1]$, $\epsilon \in (0, 1/2]$.

Output: Sets $C, F \subseteq [n]$.

- If $\rho \leq \epsilon$, then:

- $C \leftarrow \emptyset, F \leftarrow \emptyset, W \leftarrow \{x_1, \dots, x_n\}$
- While $W \neq \emptyset$:
 - * Choose arbitrary vector $x \in W$.
 - * $W \leftarrow W \setminus \{y \in W \mid \langle x, y \rangle \geq \rho - \epsilon\}$
 - * $C \leftarrow C \cup \{x\}$
 - * If $\forall y \in W, \langle x, y \rangle < \rho - \epsilon$ then $F \leftarrow F \cup \{x\}$
- Return indices of C, F .

- Apply Theorem 3.4 for input X , $\delta = \epsilon/2\pi$ and output $Y \in \{-1, 1\}^{d' \times n}$ for $d' = O(\log n/\delta^2)$.

- Apply Theorem 3.5 for input Y , $d'' = n^{0.2}$, $q = 50^{-1} \log n$, $\tau^- = -1$, $\tau^+ = 1 - \frac{2 \cos^{-1}(\rho - \epsilon)}{\pi} + \delta$ and output Z, Z' .

- Run algorithm `Crude ApprxNet` with input $\tau = 3n^{0.16}$, $\alpha = \sqrt{\epsilon}/500$, Z, Z' and output C, F .
- Return C, F .

Theorem 3.6 *The algorithm `Inner product ApprxNet`, on input $X = [x_1, \dots, x_n]$ with each $x_i \in \mathbb{S}^{d-1}$, $\rho \in [-1, 1]$ and $\epsilon \in (0, 1/2]$, computes an approximate ρ -net with additive error ϵ , using the notation of definition 3.1. The algorithm runs in time $\tilde{O}(dn + n^{2-\sqrt{\epsilon}/600})$ and succeeds with probability $1 - O(1/n^{0.2})$. Additionally, it computes a set F with the following property: $\{x_i \mid \forall x_j \neq x_i \langle x_j, x_i \rangle < \rho - \epsilon\} \subseteq F \subseteq \{x_i \mid \forall x_j \neq x_i \langle x_j, x_i \rangle < \rho\}$.*

The proof of Theorem 3.6 relies on fast matrix multiplication and specifically it employs Coppersmith's result presented below.

Theorem 3.7 [14] For any positive $\gamma > 0$, provided that $\beta < 0.29$, the product of a $k \times k^\beta$ with a $k^\beta \times k$ matrix can be computed in time $O(k^{2+\gamma})$.

Corollary 3.8 For any positive $\gamma > 0$, provided that $\beta < 0.29 \cdot \alpha < 1$, the product of a $n \times n^\beta$ by a $n^\beta \times n^\alpha$ matrix can be computed in time $O(n^{1+\alpha+\alpha\gamma})$.

Proof. The idea is to perform $n^{1-\alpha}$ multiplications of matrices of size $n^\alpha \times n^\beta$ and $n^\beta \times n^\alpha$.

Hence, by Theorem 3.7, the total cost is:

$$O(n^{1-\alpha}(n^{\alpha(2+\gamma)})) = O(n^{1+\alpha+\alpha\gamma}).$$

□

The following fact describes important properties of the Chebyshev polynomial (of the first kind) that we will rely on, specifically for the proof of theorem 3.6.

Fact 3.9 Let $T_q(x)$ denote the q th Chebyshev polynomial of the first kind, then the following hold:

- For $x \in [-1, 1]$, $|T_q(x)| \leq 1$.
- For $\delta \in (0, 1/2]$, $T_q(1 + \delta) \geq \frac{1}{2}e^{q\sqrt{\delta}}$.

Claim 3.1 For $\rho \in [-1, 1]$, $\epsilon \in (0, 1)$, it holds $\cos^{-1}(\rho - \epsilon) - \cos^{-1}(\rho) \geq \epsilon/2$.

Proof. If $(\rho - \epsilon)^2 \neq 1$ then we have

$$\begin{aligned}
& \cos^{-1}(\rho - \epsilon) - \cos^{-1}(\rho) = \\
&= \int_{\rho-\epsilon}^1 \frac{1}{\sqrt{1-x^2}} dx - \int_{\rho}^1 \frac{1}{\sqrt{1-x^2}} dx = \\
&= \int_{\rho-\epsilon}^{\rho} \frac{1}{\sqrt{1-x^2}} dx = \int_0^{\epsilon} \frac{1}{\sqrt{1-(\rho-\epsilon+y)^2}} dy \geq \\
&\geq \int_0^{\epsilon} \frac{1}{\sqrt{1-(\rho-\epsilon)^2}} dy = \frac{\epsilon}{\sqrt{1-(\rho-\epsilon)^2}} \geq \epsilon.
\end{aligned}$$

Now if $(\rho - \epsilon)^2 \neq 1 \implies \rho - \epsilon = -1$ then,

$$\begin{aligned}
& \cos^{-1}(\rho - \epsilon) - \cos^{-1}(\rho) = \int_{-1}^{-1+\epsilon} \frac{1}{\sqrt{1-x^2}} dx \geq \\
&\geq \frac{\epsilon}{\sqrt{2\epsilon - \epsilon^2}} \geq \epsilon/2.
\end{aligned}$$

□

Proof of Theorem 3.6. If $\rho \leq \epsilon$, our approach ensures that for any $x, y \in C$, it holds $\langle x, y \rangle < \rho - \epsilon \leq 0$. We show that $|C| \leq d + 1$, due to a simple packing argument. Let x_1, \dots, x_{d+2} such that $\forall i \neq j \in [d+2]$ we have $\langle x_i, x_j \rangle < 0$. Then, there exist $\lambda_1, \dots, \lambda_{d+1} \in \mathbb{R}$ not all zero for which $\sum_{i=1}^{d+1} \lambda_i x_i = 0$. Now consider two subsets $I, J \subseteq [d+2]$ of indices such that $\forall i \in I, \lambda_i > 0$ and $\forall j \in J, \lambda_j < 0$. We can write

$$\sum_{i \in I} \lambda_i x_i = \sum_{j \in J} -\lambda_j x_j \implies 0 \leq \left\langle \sum_{i \in I} \lambda_i x_i, -\sum_{j \in J} \lambda_j x_j \right\rangle = - \sum_{i \in I, j \in J} \lambda_i \lambda_j \langle x_i, x_j \rangle < 0$$

which leads to contradiction. If $J = \emptyset$ (or equivalently if $I = \emptyset$), then $0 = \langle x_{d+2}, \sum_{i \in I} \lambda_i x_i \rangle < 0$, which leads again to contradiction.

We now focus on the case $\rho > \epsilon$. By Theorem 3.4, with probability $1 - o(1/n^2)$, the matrix Y returned by the corresponding algorithm will have the property that any pair of columns

$$\langle X_i, X_j \rangle \geq \rho \implies \frac{\langle Y_i, Y_j \rangle}{d'} \geq 1 - \frac{2 \cos^{-1}(\rho)}{\pi} - \delta$$

$$\langle X_i, X_j \rangle \leq \rho - \epsilon \implies \frac{\langle Y_i, Y_j \rangle}{d'} \leq 1 - \frac{2 \cos^{-1}(\rho - \epsilon)}{\pi} + \delta.$$

Hence, according to Claim 3.1, it suffices to set $\delta = \epsilon/3\pi$ in order to distinguish between the two cases:

$$1 - \frac{2 \cos^{-1}(\rho - \epsilon)}{\pi} + 2\delta \leq 1 - \frac{2 \cos^{-1}(\rho)}{\pi} - \delta.$$

Now we set $\tau^+ = 1 - \frac{2 \cos^{-1}(\rho - \epsilon)}{\pi} + \delta > -1$. By Theorem 3.5, with probability $1 - o(1)$,

$$\langle Y_i, Y_j \rangle \leq \tau^+ d' \implies |\langle Z_i, Z_j \rangle| \leq$$

$$\leq d'' \frac{2^q}{2^{3q-1}} + \sqrt{d''} \log n \leq 3n^{0.16}$$

for large enough n . Moreover, let Y_i, Y_j s.t. $\langle Y_i, Y_j \rangle \geq (\tau^+ + \delta)d'$. Then,

$$\begin{aligned} |\langle Z_i, Z'_j \rangle| &\geq d'' \cdot T_q \left(1 + 2 \frac{\delta}{\tau^+ + 1} \right) \frac{2^q}{2^{3q-1}} - \sqrt{d''} \log n > \\ &> \frac{1}{2} \cdot T_q \left(1 + 2 \frac{\delta}{\tau^+ + 1} \right) \cdot n^{0.16} \end{aligned}$$

for large enough n .

Then, by Fact 3.9,

$$\begin{aligned} |\langle Z_i, Z'_j \rangle| \cdot n^{-0.16} &\geq \frac{1}{4} e^{q\sqrt{\delta}} = \frac{1}{4} n^{\sqrt{\delta}/50} \geq \\ &\geq 3n^{\sqrt{\delta}/100} \geq 3n^{\sqrt{\epsilon}/400}, \end{aligned}$$

where some of the inequalities hold for large enough n .

Now, by Theorems 3.4, 3.5, 3.3 and Corollary 3.8 the time complexity is $\tilde{O}(dn + n^{2-\sqrt{\epsilon}/600})$, if we set as γ in Corollary 3.8 a sufficiently small multiple of $\sqrt{\epsilon}$. Finally, the subroutine with the higher probability of failure is **Crude ApprxNet** and by the union bound, it dominates the

total probability of failure. □

3.3 Approximate nets in high dimensions

In this section, we translate the problem of computing r -nets in $(\mathbb{R}^d, \|\cdot\|)$ to the problem of computing ρ -nets for unit vectors under inner product. One intermediate step is that of computing r -nets for unit vectors under Euclidean distance.

3.3.1 From arbitrary to unit vectors.

In this subsection, we show that if one is interested in finding an r -net for $(\mathbb{R}^d, \|\cdot\|)$, it is sufficient to solve the problem for points on the unit sphere. One analogous statement is used in [34], where they prove that one can apply a randomized mapping from the general Euclidean space to points on a unit sphere, while preserving the ratio of distances for any two pairs of points. For reasons of completeness, we include the algorithm, **Standardize**, [34] that applies this randomized mapping. The claim derives by the simple observation that an r -net in the initial space can be approximated by computing an $\epsilon r/c$ -net on the sphere, where c is the maximum norm of any given point envisaged as a vector. Our exposition is simple since we can directly employ the analogous theorem from [34].

Standardize

Input: A $d \times n$ matrix X with entries $x_{i,j} \in \mathbb{R}$, constant $\epsilon \in (0, 1)$.

Output: A $m' \times n$ matrix Y with all columns having unit norm and $m' = \log^3 n$.

- Perform a Johnson-Lindenstrauss transformation of the columns of X into dimension m' to yield matrix X' .
- Let c denote the magnitude of the largest column of X' .
- Choose a random m' -dimensional vector u of length $8c/\epsilon$.

- Let Y be the result of adding u to each column of X' and normalizing all columns so as to have unit norm.

Corollary 3.10 *There exists an algorithm, **Standardize**, which, on input a $d \times n$ matrix X with entries $x_{i,j} \in \mathbb{R}$, a constant $\epsilon \in (0, 1)$ and a distance parameter $r \in \mathbb{R}$, outputs a $m' \times n$ matrix Y , with columns having unit norm and $m' = \log^3 n$, and a distance parameter $\rho \in \mathbb{R}$, such that a ρ -net of Y is an approximate r -net of X , with probability $1 - o(1/\text{poly}(n))$.*

Towards improving the Corollary above we employ an algorithm introduced in [34], which guarantees that there is a randomized mapping from Euclidean space to points on the unit sphere that preserves the ratio of the distances of any two pair of points with only an additive error.

Theorem 3.11 [34] *There exists an algorithm, **Standardize** which on input a $d \times n$ matrix X with entries $x_{i,j} \in \mathbb{R}$ and a constant $\epsilon \in (0, 1)$ outputs a $m' \times n$ matrix Y with columns having unit norm and $m' = \log^3 n$, such that, with probability $1 - o(1/\text{poly}(n))$ for all sets of four columns Y_1, Y_2, Y_3, Y_4 of matrix Y , with X_1, X_2, X_3, X_4 being the corresponding columns of matrix X , it holds that*

$$\frac{\|Y_1 - Y_2\| \|X_3 - X_4\|}{\|Y_3 - Y_4\| \|X_1 - X_2\|} \in \left[1 - \frac{\epsilon}{10}, 1 + \frac{\epsilon}{10}\right].$$

Now, let us define two d -dimensional vectors X_{n+1}, X_{n+2} , s.t. $r' = X_{n+1} - X_{n+2}$ and $\|r'\| = r$, where X is a $d \times n$ matrix with entries $x_{i,j} \in \mathbb{R}$ and $r \in \mathbb{R}$ is the radius of the r -net of X . Also, let matrix X' denote the concatenation of X , X_{n+1} and X_{n+2} with size $d \times (n + 2)$. After applying Theorem 3.11 on input X' and $\epsilon/10$, we define $\rho := \|Y_{n+1} - Y_{n+2}\|$ to be the new radius of Y . Then, we claim that the following hold with probability $1 - o(1/\text{poly}(n))$, which immediately implies Corollary 3.10:

- For all $X_i, X_j \in X$ and their corresponding $Y_i, Y_j \in Y$, if $\|X_i - X_j\| \leq r$ then $\|Y_i - Y_j\| \leq (1 + \epsilon/10)\rho$.

- For all $X_i, X_j \in X$ and their corresponding $Y_i, Y_j \in Y$, if $\|X_i - X_j\| \geq (1 + \epsilon)r$ then $\|Y_i - Y_j\| \geq (1 + \epsilon/2)\rho$.

Proof of Corollary 3.10. From Theorem 3.11, we easily derive that for all $X_i, X_j \in X$ and their corresponding $Y_i, Y_j \in Y$, it holds that

$$\|Y_i - Y_j\| \leq (1 + \epsilon/10) \frac{\|X_i - X_j\|}{r} \rho$$

Therefore, if $\|X_i - X_j\| \leq r$, we have $\|Y_i - Y_j\| \leq (1 + \epsilon/10)\rho$. For the other direction, we use the opposite side of Theorem 3.11, thus we have that for all $X_i, X_j \in X$ and their corresponding $Y_i, Y_j \in Y$:

$$\|Y_i - Y_j\| \geq (1 - \epsilon/10) \frac{\|X_i - X_j\|}{r} \rho.$$

It follows that $\|X_i - X_j\| \geq (1 + \epsilon)r \Rightarrow \|Y_i - Y_j\| \geq (1 - \epsilon/10)(1 + \epsilon)\rho \Rightarrow \|Y_i - Y_j\| \geq (1 + \epsilon/2)\rho$.

□

3.3.2 Approximate nets under Euclidean distance.

In this subsection, we show that one can translate the problem of computing an r -net for points on the unit sphere under Euclidean distance, to finding an r -net for unit vectors under inner product as defined in Section 3.2. Moreover, we identify the subset of the r -net which contains the centers that are approximately far from any other point. Formally,

Definition 3.4 *Given a set of points X and $\epsilon > 0$, a set $F \subseteq X$ of $(1 + \epsilon)$ -approximate r -far points is defined by the following property: $\{x \in X \mid \forall y \in X \setminus \{x\} \|x - y\| > (1 + \epsilon)r\} \subseteq F \subseteq \{x \in X \mid \forall y \in X \setminus \{x\} \|x - y\| > r\}$.*

If r is greater than some constant, the problem can be immediately solved by the law of cosines. If r cannot be considered as constant, we distinguish cases $r \geq 1/n^{0.9}$ and $r < 1/n^{0.9}$. The first case is solved by a simple modification of an analogous algorithm in [34, p.13:28]. The second case is not straightforward and requires partitioning the pointset in a manner which allows computing r -nets for each part separately. Each part has bounded diameter which implies that we need to solve a “large r ” subproblem. Below, we present an algorithm that finds an

$(1 + \epsilon)$ -approximate r -net for points on the unit sphere under Euclidean distance, given that the radius r is appropriately large.

ApprxNet(Large radius)

Input: $X = [x_1, \dots, x_n]^T$ with each $x_i \in \mathbb{S}^{d-1}$ with $d = \log^3 n$, $r > 1/n^{0.9}$, $\epsilon \in (0, 1/2]$.

Output: Sets $R, F \subseteq [n]$.

- If $r > 0.2$ run **Inner Product ApprxNet** with error parameter $\epsilon/25$ and $\rho = 1 - \frac{r^2}{2}$.
- Otherwise, define the $d \times n$ matrix Z as follows: for each $i \in [d]$, select $q = \left\lceil \frac{\pi}{2 \cos^{-1}(1-r^2/2)} \right\rceil$ uniformly random vectors v_1, \dots, v_q and for all $j \in [n]$, set

$$z_{i,j} = \text{sign} \prod_{k=1}^{q} X_j^T v_k,$$

where X_j is the j th column of matrix X .

- Run **Inner Product ApprxNet** with $\rho = \left(1 - \frac{2 \cos^{-1}(1-r^2/2)}{\pi}\right)^q$, error parameter $\epsilon/100$ and input matrix Z with all entries scaled by $1/\sqrt{d}$ to make them have unit norm.

Theorem 3.12 *There exists an algorithm, **ApprxNet(Large radius)**, which, for any constant $\epsilon \in (0, 1/2]$, $X \subset \mathbb{S}^{d-1}$ s.t. $|X| = n$, outputs a $(1 + \epsilon)r$ -net and a set of $(1 + \epsilon)$ -approximate r -far points with probability $1 - O(1/n^{0.2})$. Additionally, provided $r > 1/n^{0.9}$ the runtime of the algorithm is $\tilde{O}(dn^{2-\Theta(\sqrt{\epsilon})})$.*

Proof. In the case of $r > 0.2$ we will show that the $1 + \epsilon$ multiplicative approximation on the distance translates to $c\epsilon$ additive approximation to the inner product. Applying the law of cosines, the first condition yields $\langle p, q \rangle \geq 1 - \frac{r^2}{2}$ and the second condition yields $\langle p, q \rangle \leq 1 - \frac{r^2}{2} - \frac{2\epsilon r^2 + (\epsilon r)^2}{2} < 1 - \frac{r^2}{2} - \frac{\epsilon}{25}$. So, it suffices to take $c = 1/25$.

Now suppose that $r < 0.2$. For each random vector v we have that $\mathbb{E}[\text{sign}(X_i^T v \cdot X_j^T v)] = 1 - \frac{2\theta(X_i, X_j)}{\pi}$, where $\theta(X_i, X_j)$ denotes the angle between X_i, X_j . Since expectations of independent random variables multiply, we have that, for each k ,

$$\mathbb{E}[z_{k,i}z_{k,j}] = (1 - 2 \cdot \theta(X_i, X_j)/\pi)^q.$$

Now let $\theta_r = \cos^{-1}(1 - r^2/2)$,

$$\begin{aligned} \|X_i - X_j\| \leq r &\implies \theta(X_i, X_j) \leq \theta_r \implies \\ &\implies \mathbb{E}[\langle Z_i, Z_j \rangle] \geq d(1 - 2\theta_r/\pi)^q \\ \|X_i - X_j\| \geq (1 + \epsilon)r &\implies \theta(X_i, X_j) \geq (1 + \epsilon/2)\theta_r \implies \\ &\implies \mathbb{E}[\langle Z_i, Z_j \rangle] \leq d(1 - 2(1 + \epsilon/2)\theta_r/\pi)^q. \end{aligned}$$

Notice that,

$$\frac{(1 - 2(1 + \epsilon/2)\theta_r/\pi)^q}{(1 - 2\theta_r/\pi)^q} < 1 - \epsilon/10,$$

for $q = \lfloor \pi/(2\theta_r) \rfloor$ and since $n^{-0.9} \leq r \leq 0.2$. Notice that $d(1 - 2\theta_r/\pi)^q \in [0.3d, 0.5d]$. Hence, if $\|X_i - X_j\| \leq r$ and $\|X_l - X_k\| \geq (1 + \epsilon)r$,

$$\begin{aligned} \mathbb{E}[\langle Z_l, Z_k \rangle] &< (1 - \epsilon/10)\mathbb{E}[\langle Z_i, Z_j \rangle] \leq \\ &\leq \mathbb{E}[\langle Z_i, Z_j \rangle] - 0.3d\epsilon/10, \end{aligned}$$

By a union bound over Chernoff bounds, since $d = \log^3 n$, with probability $1 - o(1/\text{poly}(n))$, the inner products between any two columns of Z differs from their expectations by $o(d)$. After performing the scaling procedure, and due to the fact that $d(1 - 2\theta_r/\pi)^q \leq 0.5d$, we conclude that it suffices to compute **Inner Product ApprxNet** with $\rho = (1 - 2\theta_r/\pi)^q$ and approximation error $\epsilon/100$.

The runtime of all components of the algorithm aside from the calls to **Inner Product ApprxNet** is bounded by $\tilde{O}(n/\cos^{-1}(1 - r^2/2)) = \tilde{O}(n^{1.9})$. \square

Let us now present an algorithm which translates the problem of finding an r -net for $r < 1/n^{0.9}$ to the problem of computing an r -net for $r \geq 1/n^{0.9}$. The main idea, illustrated in figure 3.2, is that we compute disjoint subsets S_i , which are far enough from each other, so that we can compute r -nets for each S_i independently. We show that for each S_i we can compute $T_i \subseteq S_i$ which has bounded diameter and $T'_i \subseteq S_i$ such that T_i, T'_i are disjoint, each point in T_i is far

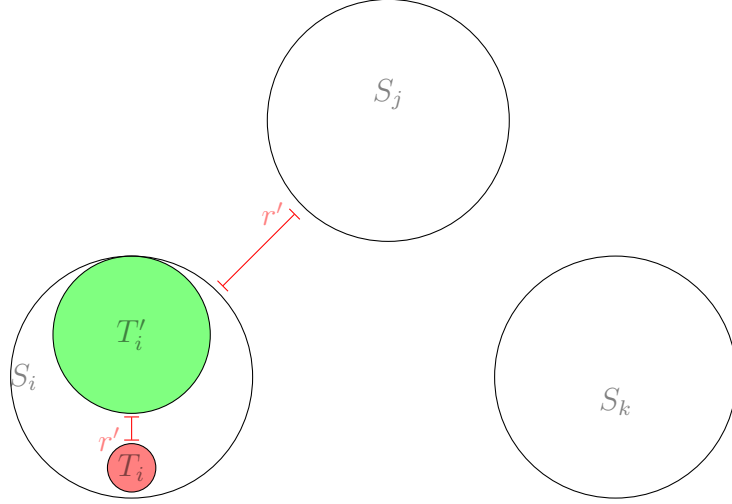


Figure 3.2: Illustration of the algorithm `ApproxNet (Small radius)`. For each disjoint subject S_i, S_j, \dots we compute an r -net independently. The set T_i is bounded in the diameter, hence we construct an r -net by employing `Standardize` and `ApproxNet (Large radius)`. In T'_i 's case we recurse the entire algorithm.

from each point in T'_i , and $|T'_i| \leq 3|S_i|/4$. It is then easy to find r -nets for T_i by employing the `ApproxNet (Large radius)` algorithm. Then, we recurse on T'_i which contains a constant fraction of points from $|S_i|$. Finally, we cover points in $S_i \setminus (T_i \cup T'_i)$ and points which do not belong to any S_i .

`ApproxNet (Small radius)`

Input: $X = [x_1, \dots, x_n]^T$ with each $x_i \in \mathbb{S}^{d-1}$, $r < 1/n^{0.9}$, $\epsilon \in (0, 1/2]$.

Output: Sets $R, F \subseteq [n]$.

1. Project points on a uniform random unit vector and consider projections p_1, \dots, p_n which wlog correspond to $x_1, \dots, x_n \in \mathbb{R}^d$.
2. Traverse the list as follows:
 - If $|\{j \mid p_j \in [p_i - r, p_i]\}| \leq n^{0.6}$ or $i = n$:
 - If $|\{j \mid p_j < p_i\}| \leq n^{0.9}$ remove from the list all points p_j s.t. $p_j < p_i - r$ and save set $K = \{x_j \mid p_j \in [p_i - r, p_i]\}$.

- If $|\{j \mid p_j < p_i\}| > n^{0.9}$ save sets $K_i = \{x_j \mid p_j \in [p_i - r, p_i]\} \cup K$, $S_i = \{x_j \mid p_j < p_i - r\} \setminus K$ and remove projections of S_i and K_i from the list.

3. After traversing the list if we have not saved any S_i go to 5; otherwise for each S_i :

- For each $u \in S_i$, sample $n^{0.1}$ distances between u and randomly chosen $x_k \in S_i$. Stop if for the selected $u \in S_i$, more than $1/3$ of the sampled points are in distance $\leq rn^{0.6}$. This means that one has found u s.t. $|\{x_k \in S_i, \|u - x_k\| \leq rn^{0.6}\}| \geq |S_i|/4$ with high probability. If no such point was found, output "ERROR".
- Let $0 \leq d_1 \leq \dots \leq d_{|S_i|}$ be the distances between u and all other points in S_i . Find $c \in [rn^{0.6}, 2rn^{0.6}]$ s.t. $|\{j \in [n] \mid d_j \in [c, c + r]\}| < n^{0.4}$, store $W_i = \{x_j \mid d_j \in [c, c + r]\}$, and remove W_i from S_i .
- Construct the sets $T_i = \{x_j \in S_i \mid d_j < c\}$ and $T'_i = \{x_j \in S_i \mid d_j > c + r\}$.
 - For T_i , subtract u from all vectors in T_i , run **Standardize**, then **ApprxNet (Large radius)**, both with $\epsilon/4$. Save points which correspond to output at R_i, F_i respectively.
 - Recurse on T'_i the whole algorithm, and notice that $|T'_i| \leq 3|S_i|/4$. Save output at R'_i , and F'_i respectively.

4. Let $R \leftarrow \bigcup_i R_i \cup R'_i$ and $F \leftarrow \bigcup_i F_i \cup F'_i$. Return to the list p_1, \dots, p_n .

- Remove from F all points which cover at least one point from $\bigcup_i W_i$ or $\bigcup_i K_i$.
- Delete all points $(\bigcup_i T_i) \setminus (\bigcup_i R_i)$, and $(\bigcup_i T'_i) \setminus (\bigcup_i R'_i)$.
- For each i delete all points in W_i covered by R_i , or covered by R'_i .
- For each i delete all points in K_i covered by R .
- Finally delete R from the list. Store the remaining points at F' .

5. $R' \leftarrow \emptyset$. Traverse the list as follows: For each p_i , check the distances from all x_j s.t. $p_j \in [p_i - r, p_i]$.

- If $\exists x_j \in R' : \|x_i - x_j\| \leq r$, delete x_i from the list, set $F' \leftarrow F' \setminus \{x_i, x_j\}$ and continue traversing the list.

- If there is no such point x_j then $R \leftarrow R \cup \{x_i\}$ and continue traversing the list.

6. Output indices of $R \leftarrow R \cup R'$ and $F \leftarrow F \cup F'$.

Theorem 3.13 For any constant $\epsilon > 0$, $X \subset \mathbb{S}^{d-1}$ s.t. $|X| = n$, and $r < 1/n^{0.9}$, **ApprxNet (Small radius)** will output a $(1 + \epsilon)r$ -net and a set of $(1 + \epsilon)$ -approximate r -far points in time $\tilde{O}(dn^{2-\Theta(\sqrt{\epsilon})})$, with probability $1 - o(1/n^{0.04})$.

Proof. Note that points in S_i had projections p_i in sets of contiguous intervals of width r ; each interval had $\geq n^{0.6}$ points, hence the diameter of the projection of S_i is $\leq n^{0.4}r$. By the Johnson Lindenstrauss Lemma [15] we have that for $v \in \mathbb{S}^{d-1}$ chosen uniformly at random:

$$\Pr \left[\langle u, v \rangle^2 \leq \frac{\|u\|^2}{n^{0.4}} \right] \leq \frac{\sqrt{d}\sqrt{e}}{n^{0.2}}.$$

Hence, $\mathbb{E}[|\{x_k, x_j \in S_i \mid \|x_k - x_j\| \geq n^{0.6}r \text{ and } \|p_k - p_j\| \leq n^{0.4}r\}|] \leq |S_i|^2 \cdot \frac{\sqrt{ed}}{n^{0.2}}$, and the probability $\Pr[|\{x_k, x_j \in S_i \mid \|x_k - x_j\| \geq n^{0.6}r \text{ and } \|p_k - p_j\| \leq n^{0.4}r\}| \geq |S_i|^{1.95}] \leq |S_i|^{0.05} \cdot \sqrt{ed} \cdot n^{-0.2} \leq \sqrt{ed} \cdot n^{-0.15}$. Taking a union bound over all sets S_i yields a probability of failure $o(1/n^{0.045})$. This implies that (for large enough n , which implies large enough $|S_i|$) at least

$$\binom{|S_i|}{2} - |S_i|^{1.95} \geq \frac{|S_i|^2}{4}$$

distances between points in S_i are indeed small ($\leq n^{0.6}r$). Hence, there exists some point $p_k \in S_i$ which $(n^{0.6}r)$ -covers $|S_i|/2$ points. For each possible p_k we sample $n^{0.1}$ distances to other points, and by Chernoff bounds, if a point $(n^{0.6}r)$ -covers a fraction of more than $1/2$ of the points in S_i , then it covers more than $n^{0.1}/3$ sampled points with high probability. Similarly, if a point $(n^{0.6}r)$ -covers a fraction of less than $1/4$ of the points in S_i , then it covers less than $n^{0.1}/3$ sampled points with high probability. More precisely, for some fixed $u \in S_i$, let $X_j = 1$ when for the j th randomly chosen point $v \in S_i$, it holds $\|u - v\| \leq n^{0.6}r$ and let $X_j = 0$ otherwise. Then, for $Y = \sum_{j=1}^{n^{0.1}} X_j$, it holds:

$$\mathbb{E}[Y] \geq n^{0.1}/2 \implies \Pr[Y \leq n^{0.1}/3] \leq \exp(-\Theta(n^{0.1})),$$

$$\mathbb{E}[Y] \leq n^{0.1}/4 \implies \Pr[Y \geq n^{0.1}/3] \leq \exp(-\Theta(n^{0.1})).$$

Since for any point $x \in T_i$ and any point $y \in T'_i$ we have $\|x - y\| > r$, the packing property of r -nets is preserved when we build r -nets for T_i and T'_i independently. For each T_i , we succeed in building r -nets with probability $1 - O(1/n^{0.2})$. By a union bound over all sets T_i , we have a probability of failure $O(1/n^{0.1})$. Furthermore, points which belong to sets W_i and K_i are possibly covered and need to be checked.

For the analysis of the runtime of the algorithm, notice that step 4b costs time $O(d \cdot (\sum_i |T_i| + \sum_i |T'_i|)) = O(dn)$. Then, step 4c costs time $O(d \cdot \sum_i |W_i| \cdot |T_i| + d \cdot \sum_i |W_i| \cdot |T'_i|) = O(dn^{1.4})$. Finally, notice that we have at most $n^{0.1}$ sets K_i . Each K_i contains at most $2n^{0.6}$ points, hence checking each point in $\bigcup_i K_i$ with each point in R costs $O(dn^{1.7})$.

Now regarding step 5, consider any interval $[p_i - r, p_i]$ in the initial list, where all points are projected. If $|\{j \mid p_j \in [p_i - r, p_i]\}| \leq 2n^{0.9}$ then the i th iteration in step 5 will obviously cost $O(n^{0.9})$, since previous steps only delete points. If $|\{j \mid p_j \in [p_i - r, p_i]\}| > 2n^{0.9}$, we claim that $|\{j < i \mid p_j \in [p_i - r, p_i] \text{ and } K_j \text{ is created}\}| \leq 1$. Consider the smallest $j < i$ s.t. K_j is created and $p_j \in [p_i - r, p_i]$. This means that all points p_k , for $k \leq j$, are deleted when p_j is visited. Now assume that there exists integer $l \in (j, i)$ s.t. K_l is created. This means that the remaining points in the interval $[p_l - r, p_l]$ are $\leq n^{0.6}$ and all of the remaining points $p_k < p_l$ are more than $n^{0.9}$. This leads to contradiction, since by the deletion in the j th iteration, we know that all of the remaining points $p_k < p_l$ lie in the interval $[p_l - r, p_l]$.

Now, assume that there exists one $j < i$ s.t. $p_j \in [p_i - r, p_i]$ and K_j is created. Then, when p_i is visited, there at least $2n^{0.9} - n^{0.6} > n^{0.9}$ remaining points in the interval $[p_i - r, p_i]$. Hence, there exists $l \geq i$ for which the remaining points in the interval $[p_i - r, p_i]$ are contained in $S_l \cup K_l$. Hence in this case, in step 5, there exist at most $O(n^{0.6})$ points which are not deleted and belong to the interval $[p_i - r, p_i]$. Now assume that there does not exist any $j < i$ s.t. $p_j \in [p_i - r, p_i]$ and K_j is created. This directly implies that there exists $l \geq i$ for which the remaining points in the interval $[p_i - r, p_i]$ are contained in $S_l \cup K_l$.

At last, the total time of the above algorithm is dominated by the calls to the construction of the partial r -nets of the sets T_i . Thus, the total running time is $O(\sum_i |T_i|^{2-\Theta(\sqrt{\epsilon})} + \sum_i |T_i|'^{2-\Theta(\sqrt{\epsilon})}) = O(\sum_i |T_i|^{2-\Theta(\sqrt{\epsilon})} + \sum_i (3|T_i|/4)^{2-\Theta(\sqrt{\epsilon})}) = \tilde{O}(n^{2-\Theta(\sqrt{\epsilon})})$. Finally, taking a union bound over all recursive calls of the algorithm we obtain a probability of failure $o(1/n^{0.04})$. \square

We now present an algorithm for an $(1 + \epsilon)r$ -net for points in \mathbb{R}^d under Euclidean distance.

ApprxNet

Input: Matrix $X = [x_1, \dots, x_n]$ with each $x_i \in \mathbb{R}^d$, parameter $r \in \mathbb{R}$, constant $\epsilon \in (0, 1/2]$.

Output: $R \subseteq \{x_1, \dots, x_n\}$

- Let Y, r' be the output of algorithm **Standardize** on input X, r with parameter $\epsilon/4$.
- If $r' \geq 1/n^{0.9}$ run **ApprxNet(Large radius)** on input $Y, \epsilon/4, r'$ and return points which correspond to the set R .
- If $r' < 1/n^{0.9}$ run **ApprxNet(Small radius)** on input $Y, \epsilon/4, r'$ and return points which correspond to the set R .

Theorem 3.14 *Given n points in \mathbb{R}^d , a distance parameter $r \in \mathbb{R}$ and an approximation parameter $\epsilon \in (0, 1/2]$, with probability $1 - o(1/n^{0.04})$, **ApprxNet** will return a $(1 + \epsilon)r$ -net, R , in $\tilde{O}(dn^{2-\Theta(\sqrt{\epsilon})})$ time.*

Proof. The theorem is a direct implication of Theorems 3.12, 3.13, and Corollary 3.10. \square

Moreover, we present a randomized approximation algorithm which, given a pointset in \mathbb{R}^d and distance parameter r , returns the points that have at least one neighbor at distance at most r , as shown in the figure 3.3. This algorithm will be used on the applications on Section 4.

DelFar

Input: Matrix $X = [x_1, \dots, x_n]$ with each $x_i \in \mathbb{R}^d$, parameter $r \in \mathbb{R}$, constant $\epsilon \in (0, 1/2]$.

Output: $F' \subseteq \{x_1, \dots, x_n\}$.

- Let Y, r' be the output by algorithm **Standardize** on input X, r with parameter $\epsilon/4$.

- If $r \geq 1/n^{0.9}$ run `ApprxNet(Large radius)` on input Y , $\epsilon/4, r$ and return points which correspond to the set $F' \leftarrow X \setminus F$.
- If $r < 1/n^{0.9}$ run `ApprxNet(Small radius)` on input Y , $\epsilon/4, r$ and return points which correspond to the set $F' \leftarrow X \setminus F$.

Theorem 3.15 *Given $X \subset \mathbb{R}^d$ such that $|X| = n$, a distance parameter $r \in \mathbb{R}$ and an approximation parameter $\epsilon \in (0, 1/2]$, there exists an algorithm, `DelFar`, that will return, with probability $1 - o(1/n^{0.04})$, a set F' with the following properties in $\tilde{O}(dn^{2-\Theta(\sqrt{\epsilon})})$ time:*

- *If for a point $p \in X$ it holds that $\forall q \neq p, q \in X$ we have $\|p - q\| > (1 + \epsilon)r$, then $p \notin F'$.*
- *If for a point $p \in X$ it holds that $\exists q \neq p, q \in X$ s.t. $\|p - q\| \leq r$, then $p \in F'$.*

Proof. By Theorems 3.12, 3.13, 3.10, both `ApprxNet(Large radius)` and `ApprxNet(Small radius)` return a set F , the subset of the centers of r -net that are isolated, i.e. the points that do not have any neighbor at distance $(1 + \epsilon)r$. Also, both procedures run in $\tilde{O}(dn^{2-\Theta(\sqrt{\epsilon})})$. Thus, `DelFar` on input a $d \times n$ matrix X , a radius $r \in \mathbb{R}$ and a fixed constant $\epsilon \in (0, 1/2]$ returns a set $F' \subseteq \{x_1, \dots, x_n\}$, which contains all the points (vectors) of X that have at least one neighbor at distance r . Additionally, the algorithm costs $\tilde{O}(dn^{2-\Theta(\sqrt{\epsilon})})$ time and succeeds with probability $1 - o(n^{0.04})$. \square

3.4 Applications

Concerning applications, in [22], they design an approximation scheme, which solves various distance optimization problems. The technique employs a grid-based construction of r -nets which is linear in n , but exponential in d . The main prerequisite of the method is the existence of a linear-time decider (formally defined in Appendix 3.4.1). The framework is especially interesting when the dimension is constant, since the whole algorithm costs time linear in n which, for some problems, improves upon previously known near-linear algorithms. When the dimension is high, we aim for polynomial dependency on d , and subquadratic dependency on

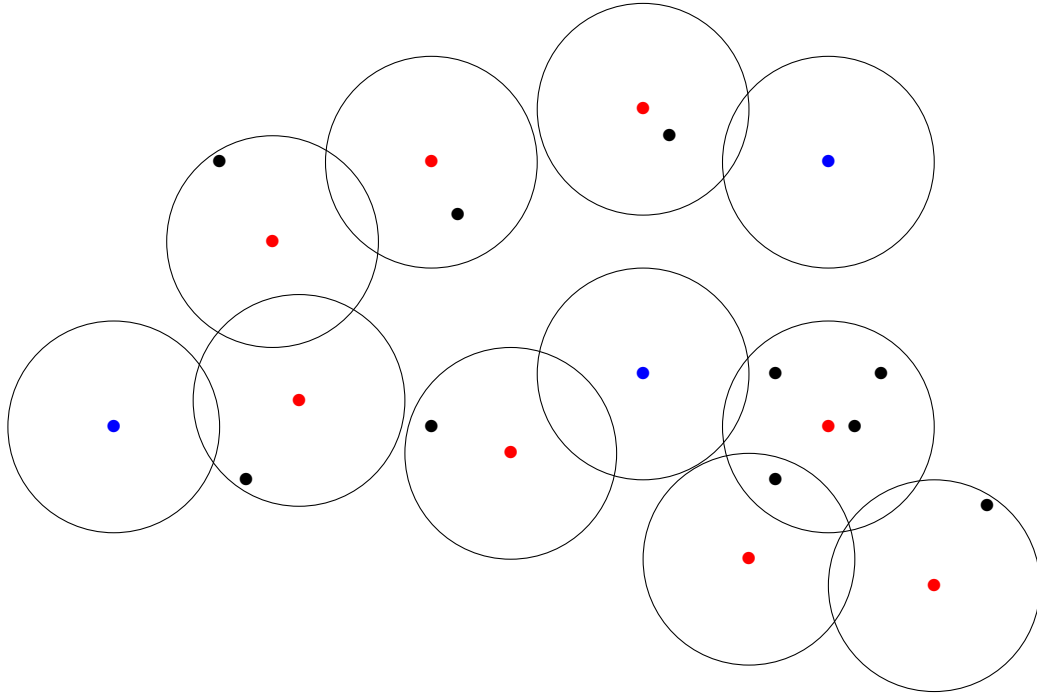


Figure 3.3: The `ApprxNet` algorithm returns both red and blue points of this figure, while the `DelFar` algorithm returns only the red points that have at least one r -close neighbor.

n . In the first subsection, we present the modified framework for high dimensional data sets, employing the algorithms `ApprxNet` and `DelFar` from this work. In the last subsection, we discuss more possible applications and other research directions in order to improve this result even more.

3.4.1 A general framework for high dimensional distance problems

In this subsection, we modify a framework originally introduced by [22], which provides an efficient way for constructing approximation algorithms for a variety of well known distance problems. We present the algorithm `Net` and `Prune` of [22], modified to call the algorithms `ApprxNet` and `DelFar`. We claim that this algorithm computes, with high probability, a constant spread interval and costs $O(dn^{1.999999})$ time.

We assume the existence of a fast approximate decider procedure for the problems we want to address using this framework, specifically an algorithm that runs in $\tilde{O}(dn^{2-\Theta(\sqrt{\epsilon})})$, where ϵ is

the approximation factor. Formally,

Definition 3.5 Given a function $f : X \rightarrow \mathbb{R}$, we call a decider procedure a $(1 + \epsilon)$ -decider for f , if for any $x \in X$ and $r > 0$, $\text{decider}(r, x)$ returns one of the following: (i) $f(x) \in [\alpha, (1 + \epsilon)\alpha]$, where α is some real number, (ii) $f(x) < r$, or (iii) $f(x) > r$.

Additionally, we assume the problems we seek to improve with this method have the following property: if the decider returns that the optimal solution is smaller than a fixed value r , we can efficiently remove all points that do not have any neighbor at distance at most r and this does not affect the optimal solution. Let us denote $f(X)$ the optimal solution of a problem for input X .

Net & Prune

Input: An instance (X, Γ) s.t. $X \subseteq \mathbb{R}^d$.

Output: An interval $[x, y]$ containing the optimal value.

- $X_0 = X, i = 0$
- While TRUE do
 - Choose at random a point $x \in X_i$ and compute its nearest neighbor distance, l_i
 - Call $\frac{3}{2}$ -decider($2l_i/3, X_i$) and $\frac{3}{2}$ -decider(cl_i, X_i). Do one of the following:
 - * If $\frac{3}{2}$ -decider($2l_i/3, X_i$) returns $f(X_i) \in [x, y]$, return $f(X) \in [x/2, 2y]$
 - * If $\frac{3}{2}$ -decider(cl_i, X_i) returns $f(X_i) \in [x', y']$, return $f(X) \in [x'/2, 2y']$
 - * If $2l_i/3$ is too small and cl_i too large, return $[l_i/3, 2cl_i]$
 - * If $2l_i/3$ is too large, call $X_{i+1} = \text{DelFar}(2l_i/3, X_i, \frac{3}{2})$
 - * If cl_i is too small, $X_{i+1} = \text{ApprxNet}(4l_i, X_i, \frac{3}{2})$
 - $i = i + 1$,

Let us denote as $|X_i^{<l}|$ and $|X_i^{\geq l}|$ the set of points in X , whose nearest neighbor distance is smaller than l and greater than l , respectively.

Theorem 3.16 *Assume that the DelFar algorithm and the ApprxNet algorithm succeed with probability $1 - \frac{1}{n^{0.01}}$. The algorithm Net & Prune (X, Γ) runs in expected $O(dn^{1.999999})$ time.*

Proof. In each iteration of the while loop the algorithm calls on input X_i the $\frac{3}{2}$ -decider procedure and either ApprxNet or DelFar, all of which cost $O(d|X_i|^{1.999999})$ time. Thus, the total running time of the algorithm is $O(\sum_{i=0}^{i=k-1} d|X_i|^{1.999999})$, where k denotes the last iteration of the while loop.

In the $(i + 1)$ th iteration of the while loop, where $(i + 1 < k)$, let's assume that x_1, x_2, \dots, x_m is the points' labels in increasing order of their nearest neighbor distance in X_i . If j is the index of the chosen point on the first step of the algorithm and $X_i^{\geq j}$ and $X_i^{\leq j}$ are the subsets of points with index $\geq j$ and $\leq j$, respectively, then we call i a successful iteration when $j \in [m/4, 3m/4]$. Then, we have that $|X_i^{\geq j}| \geq |X_{i+1}|/4$ and $|X_i^{\leq j}| \geq |X_{i+1}|/4$ for a successful iteration. The probability that $i + 1$ is a successful iteration is $1/2$.

At each iteration, but the last, either ApprxNet or DelFar gets called. Thus, for any successful iteration, a constant fraction of the point set is removed (it follows from Lemma 3.2.3 in [22] and Theorem 3.15). Also, the algorithms $(1 + \epsilon)$ -decider, ApprxNet and DelFar succeed at every call with probability $1 - \frac{O(\log n)}{n^{0.01}} = 1 - o(1)$, since the expected number of iterations is $O(\log n)$. Hence, the expected running time of the algorithm is $O(dn^{1.999999})$, given the above algorithms succeed. \square

At every step, either far points are being removed or we net the points. If the DelFar algorithm is called, then with small probability we remove a point which is not far. This obviously affects the optimal value, thus we will prove the correctness of the algorithm with high probability. On the other hand, if the ApprxNet algorithm is called, the net radius is always significantly smaller than the optimal value, so the accumulated error in the end, which is proportional to the radius of the last net computation, is also much smaller than the optimal value. For the following proofs we assume both DelFar and ApprxNet algorithms succeed, which occurs with probability $1 - o(1)$.

Lemma 3.17 *For every iteration i , we have $|f(X_i) - f(X_0)| \leq 16l_i$.*

Proof. Let I be the set of indices of the **ApprxNet** iterations up to the i th iteration. Similarly, let I' be the set of iterations where **DelFar** is called.

If **ApprxNet** was called in the j th iteration, then X_j is at most a $6l_j$ -drift of X_{j-1} , therefore $|f(X_j) - f(X_{j-1})| \leq 12l_j$. Also, if **DelFar** is called in the j th iteration, then $f(X_j) = f(X_{j-1})$ (by Theorem 3.15). Let $m = \max I$, we have that,

$$\begin{aligned} |f(X_i) - f(X_0)| &\leq \sum_{j=1}^i |f(X_j) - f(X_{j-1})| = \\ &= \sum_{j \in I} |f(X_j) - f(X_{j-1})| + \sum_{j \in I'} |f(X_j) - f(X_{j-1})| \\ &\leq \sum_{j \in I} 12l_j + \sum_{j \in I'} 0 \leq 12l_m \sum_{j=0}^{\infty} \left(\frac{1}{4}\right)^j \leq 16l_m \leq 16l_i, \end{aligned}$$

where the second inequality holds since for every $j < i$, in the beginning of the j th iteration of the while loop, the set of points X_{j-1} is a subset of the net points of a $4l_i$ -net, therefore $l_j \geq 4l_i$.

□

Lemma 3.18 *For any iteration i of the while loop such that **ApprxNet** gets called, we have $l_i \leq f(X_0)/\eta$, where $\eta = c - 16$.*

Proof. We will prove this with induction. Let m_1, m_2, \dots, m_t be the indices of the iterations of the while loop in which **ApprxNet** gets called.

Base: In order for **ApprxNet** to get called we must have $\eta l_{m_1} < cl_{m_1} < f(X_{m_1-1})$ and since this is the first time **ApprxNet** gets called we have $f(X_{m_1-1}) = f(X_0)$. Therefore, $\eta l_{m_1} < f(X_0)$.

Inductive step: Suppose that $l_{m_j} \leq f(X_0)/\eta$, for all $m_j < m_i$. If a call to $\frac{3}{2}$ -rNet is made in iteration m_i then again $cl_{m_i} < f(X_{(m_i)-1}) = f(X_{m_i-1})$. Thus, by the induction hypothesis and Lemma 3.17 we have,

$$\begin{aligned} l_{m_i} &< \frac{f(X_{m_i-1})}{c} \leq \frac{f(X_0) + 16l_{m_i-1}}{c} \leq \\ &\leq \frac{f(X_0) + 16f(X_0)/\eta}{c} = \frac{1 + 16/\eta}{c} f(X_0) = f(X_0)/\eta. \end{aligned}$$

□

Therefore, if we set $c = 64$ we have $\eta = 48$, thus by Lemma 3.17 and Lemma 3.18,

$$|f(X_i) - f(X_0)| \leq 16l_i \leq 16f(X_0)/\eta = f(X_0)/3.$$

Corollary 3.19 *For $c \geq 64$ and for any iteration i we have:*

- $(2/3)f(X_0) \leq f(X_i) \leq (4/3)f(X_0)$,
- if $f(X_i) \in [x, y]$, then $f(X_0) \in [(3/4)x, (3/2)y] \subseteq [x/2, 2y]$,
- if $f(X_0) > 0$ then $f(X_i) > 0$.

Theorem 3.20 *For $c \geq 64$, the Net & Prune algorithm computes in $O(dn^{1.999999})$ time a constant spread interval containing the optimal value $f(X)$, with probability $1 - o(1)$.*

Proof. Consider the iteration of the while loop at which Net & Prune terminates. If the interval $[x, y]$ was computed by the $\frac{3}{2}$ -decider, then it has spread $\leq \frac{3}{2}$. Thus, by Corollary 3.19 the returned interval $[x', y'] = [x/2, 2y]$ contains the optimal value and its spread is ≤ 6 . Similarly, if $2l_i/3$ is too small and cl_i too large, then the returned interval is $[\frac{l_i}{3}, 2cl_i]$ and its spread is 384. \square

3.4.2 k -th nearest neighbor distance

Let us focus on the problem of approximating the k th nearest neighbor distance.

Definition 3.6 *Let $X \subset \mathbb{R}^d$ be a set of n points, approximation error $\epsilon > 0$, and let $d_1 \leq \dots \leq d_n$ be the nearest neighbor distances. The problem of computing an $(1 + \epsilon)$ -approximation to the k th nearest neighbor distance asks for a pair $x, y \in X$ such that $\|x - y\| \in [(1 - \epsilon)d_k, (1 + \epsilon)d_k]$.*

Now we present an approximate decider for the problem above. This procedure combined with the framework we mentioned earlier, which employs our net construction, results in an efficient solution for this problem in high dimension.

kth NND Decider

Input: $X \subseteq \mathbb{R}^d$, constant $\epsilon \in (0, 1/2]$, integer $k > 0$.

Output: An interval for the optimal value $f(X, k)$.

- Call $\text{DelFar}(X, \frac{r}{1+\epsilon/4}, \epsilon/4)$ and store its output in W_1 .
- Call $\text{DelFar}(X, r, \epsilon/4)$ and store its output in W_2 .
- Do one of the following:
 - If $|W_1| > k$, then output “ $f(X, k) < r$ ”.
 - If $|W_2| < k$, then output “ $f(X, k) > r$ ”.
 - If $|W_1| \leq k$ and $|W_2| \geq k$, then output “ $f(X, k) \in [\frac{r}{1+\epsilon/4}, \frac{1+\epsilon/4}{r}]$ ”.

Theorem 3.21 *Given a pointset $X \subseteq \mathbb{R}^d$, one can compute a $(1 + \epsilon)$ -approximation to the k -th nearest neighbor in $\tilde{O}(dn^{2-\Theta(\sqrt{\epsilon})})$, with probability $1 - o(1)$.*

Proof. For this particular problem, the optimal solution is not affected by the DelFar 's removal of the points with no other point at distance at most r . Also, each time the ApprxNet algorithm is called, for a fixed distance r , the drift of the optimal solution is at most $2r$. Thus, Theorem 3.20 holds, and we compute a constant spread interval $[x, y]$ containing the optimal value, with high probability. We then apply binary search on values $x, (1 + \epsilon)x, (1 + \epsilon)^2x, \dots, y$ using the algorithm k th NND Decider. We perform $O(1/\log(1 + \epsilon)) = O(1/\epsilon^2)$ iterations, hence the total amount of time needed is $\tilde{O}(dn^{2-\Theta(\sqrt{\epsilon})})$ and the algorithm succeeds with high probability $1 - o(1)$. \square

To the best of our knowledge, this is the best high dimensional solution for this problem, when ϵ is sufficiently small. Setting $k = n$ and applying Theorem 3.21 one can compute the *farthest nearest neighbor* in $\tilde{O}(dn^{2-\Theta(\sqrt{\epsilon})})$ with high probability.

Chapter 4

Conclusion

4.1 Summary of Thesis Achievements

The novel results of the thesis are thoroughly presented in chapters 2 and 3. In chapter 2 we propose a simple data structure for approximating the near neighbor decision problem, while in chapter 3 we present an improved -in terms of time complexity- algorithm for the construction of r -nets. Both those problems are examined when the dimension is high.

To be more specific, we have designed a conceptually simple method for a fast and compact approach to Near Neighbor queries, and have tested it experimentally. This offers a competitive approach to Approximate Nearest Neighbor search. Our method is optimal in space, with sublinear query time for any constant approximation factor $c > 1$. The algorithm randomly projects points to the Hamming hypercube. The query algorithm simply projects the query point, then examines points which are assigned to the same or nearby vertices on the hypercube. We have analyzed the query time for the Euclidean and Manhattan metrics.

However, the critical contribution of the thesis' author concerns the algorithm for constructing r -nets, presented in chapter 3. In this work, we focus on high-dimensional spaces and present a new randomized algorithm which efficiently computes approximate r -nets with respect to Euclidean distance. The complexity of the algorithm is polynomial in the dimension and subquadratic in the number of points, while the the approximation factor is $1 + \epsilon$, for any fixed $\epsilon > 0$. The algorithm succeeds with high probability and our result improves upon the best

previously known (LSH-based) construction of Eppstein et al. [17] in terms of complexity, by reducing the dependence on ϵ , provided that ϵ is sufficiently small. Moreover, our method does not require LSH but follows Valiant’s [34] approach in designing a sequence of reductions of our problem to other problems in different spaces, under Euclidean distance or inner product, for which r -nets are computed efficiently and the error can be controlled. Our result immediately implies efficient solutions to a number of geometric problems in high dimension, such as finding the $(1 + \epsilon)$ -approximate k -th nearest neighbor distance in time subquadratic in the size of the input.

4.2 Future Work

Regarding our work on approximate nearest neighbor search, we have focused only on data-independent methods for ANN, while data-dependent methods achieve better guarantees in theory. Hence, designing a practical data-dependent variant of our method will be a challenging step. Moreover, since our implementation easily extends to other LSH families, it would be interesting to implement and conduct experiments for other metrics.

Concerning future work on our results on approximate r -nets on high dimensional data sets, let us start with the problem of finding a greedy permutation. A permutation $\Pi = \langle \pi_1, \pi_2, \dots \rangle$ of the vertices of a metric space $(X, \|\cdot\|)$ is a *greedy permutation* if each vertex π_i is the farthest in X from the preceding vertices $\Pi_{i-1} = \langle \pi_1, \dots, \pi_{i-1} \rangle$. The computation of r -nets is closely related to that of the greedy permutation.

Furthermore, our algorithm can be plugged into the framework of [22] to achieve a $(4 + \epsilon)$ approximation for the k -center problem in time $\tilde{O}(dn^{2-\Theta(\sqrt{\epsilon})})$. By [17], a simple modification of our net construction implies an algorithm for the $(1 + \epsilon)$ approximate greedy permutation in time $\tilde{O}(dn^{2-\Theta(\sqrt{\epsilon})} \log \Phi)$ where Φ denotes the spread of the pointset. Then, approximating the greedy permutation implies a $(2 + \epsilon)$ approximation algorithm for k -center clustering problem. We expect that one can avoid any dependencies on Φ .

Recently, [2] improved on the algorithm by [34] for the approximate closest pair problem, by presenting an algorithm with randomized time near $dn + n^{2-\Theta(\epsilon^{1/3}/\log(1/\epsilon))}$. It is possible that a similar improvement can be achieved for the problem of approximating r -nets.

Bibliography

- [1] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Geometric approximation via coresets. In *Combinatorial and Computational Geometry, MSRI*, pages 1–30. University Press, 2005.
- [2] J. Alman, T. M. Chan, and R. Williams. Polynomial representations of threshold functions and algorithmic application. In *Proc. 57th IEEE Symp. on Foundations of Computer Science (FOCS)*, pages 467–476, 2016.
- [3] E. Anagnostopoulos, I. Z. Emiris, and I. Psarros. Low-quality dimension reduction and high-dimensional approximate nearest neighbor. In *31st International Symposium on Computational Geometry, SoCG 2015, June 22-25, 2015, Eindhoven, The Netherlands*, pages 436–450, 2015.
- [4] E. Anagnostopoulos, I.Z. Emiris, and I. Psarros. Randomized embeddings with slack, and high-dimensional approximate nearest neighbor. *Trans. of Algorithms, under revision*, 2014–2016.
- [5] A. Andoni and P. Indyk. Efficient algorithms for substring near neighbor problem. In *Proc. 17th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1203–1212, 2006.
- [6] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Commun. ACM*, 51(1):117–122, 2008.
- [7] A. Andoni, P. Indyk, T. Laarhoven, I.P. Razenshteyn, and L. Schmidt. Practical and optimal LSH for angular distance. In *Advances Neural Information Proc. Systems 28: Annual Conf. Neural Information Processing Systems*, pages 1225–1233, 2015.

- [8] A. Andoni, T. Laarhoven, I. Razenshteyn, and E. Waingarten. Optimal hashing-based time-space trade-offs for approximate near neighbors. In *Proc. Symp. Discrete Algorithms (SODA)*, 2017. Also arxiv.org/abs/1608.03580.
- [9] A. Andoni, I. Razenshteyn, and N. Shekel Nosatzki. LSH forest: Practical algorithms made theoretical. In *Proc. Symp. Discrete Algorithms (SODA)*, 2017.
- [10] I. Andoni, A. and Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *Proc. 47th ACM Symp. Theory of Computing, STOC'15*, 2015.
- [11] G. Avarikioti, I. Z. Emiris, L. Kavouras, and I. Psarros. High-dimensional approximate r -nets. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 16–30, 2017.
- [12] G. Avarikioti, I. Z. Emiris, I. Psarros, and G. Samaras. Practical linear-space approximate near neighbors in high dimension. In *Proc. EuroCG 2017, CoRR*, [abs/1612.07405](https://arxiv.org/abs/1612.07405), 2016.
- [13] M. Charikar. Similarity estimation techniques from rounding algorithms. In *Proc. 34th Annual ACM Symp. on Theory of Computing, 2002, Montréal, Canada*, pages 380–388, 2002.
- [14] D. Coppersmith. Rectangular matrix multiplication revisited. *J. Complexity*, 13(1):42–49, March 1997.
- [15] S. Dasgupta and A. Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Struct. Algorithms*, 22(1):60–65, 2003.
- [16] M. Datar, N. Immorlica, P. Indyk, and V.S. Mirrokni. Locality-sensitive hashing scheme based on p -stable distributions. In *Proc. 20th Annual Symp. Computational Geometry, SCG'04*, pages 253–262, 2004.
- [17] D. Eppstein, S. Har-Peled, and A. Sidiropoulos. Approximate greedy clustering and distance selection for graph metrics. *CoRR*, [abs/1507.01555](https://arxiv.org/abs/1507.01555), 2015.
- [18] A. Goel, P. Indyk, and K.R. Varadarajan. Reductions among high dimensional proximity problems. In *Proc. 12th Symp. on Discrete Algorithms (SODA)*, pages 769–778, January 2001.

- [19] S. Har-Peled. Clustering motion. *Discrete & Computational Geometry*, 31(4):545–565, 2004.
- [20] S. Har-Peled, P. Indyk, and R. Motwani. Approximate nearest neighbor: Towards removing the curse of dimensionality. *Theory of Computing*, 8(1):321–350, 2012.
- [21] S. Har-Peled and M. Mendel. Fast construction of nets in low dimensional metrics, and their applications. In *Proc. 21st Annual Symp. Computational Geometry*, pages 150–158, 2005.
- [22] S. Har-Peled and B. Raichel. Net and prune: A linear time algorithm for euclidean distance problems. *J. ACM*, 62(6):44, 2015.
- [23] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. 30th Annual ACM Symp. Theory of Computing*, STOC’98, pages 604–613, 1998.
- [24] P. N. Klein, editor. *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*. SIAM, 2017.
- [25] E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM J. Comput.*, 30(2):457–474, 2000.
- [26] K. Li and J. Malik. Fast k-nearest neighbour search via dynamic continuous indexing. In *Proc. of the 33rd International Conference on Machine Learning - Volume 48, ICML’16*, pages 671–679. JMLR.org, 2016.
- [27] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Multi-probe lsh: Efficient indexing for high-dimensional similarity search. In *Proc. of the 33rd International Conf. on Very Large Data Bases, VLDB ’07*, pages 950–961. VLDB Endowment, 2007.
- [28] M. Mitzenmacher and E. Upfal. *Probability and computing - randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.
- [29] R. Motwani, A. Naor, and R. Panigrahy. Lower bounds on locality sensitive hashing. *SIAM J. Discrete Math.*, 21(4):930–935, 2007.

- [30] R. O’Donnell, Y. Wu, and Y. Zhou. Optimal lower bounds for locality-sensitive hashing (except when q is tiny). *ACM Trans. Comput. Theory*, 6(1):5:1–5:13, March 2014.
- [31] R. Panigrahy. Entropy based nearest neighbor search in high dimensions. In *Proc. 17th Annual ACM-SIAM Symp. Discrete Algorithms*, SODA’06, pages 1186–1195, 2006.
- [32] Y. Sun, W. Wang, J. Qin, Y. Zhang, and X. Lin. Srs: Solving c -approximate nearest neighbor queries in high dimensional euclidean space with a tiny index. *Proc. VLDB Endow.*, 8(1):1–12, September 2014.
- [33] G. Valiant. Finding correlations in subquadratic time, with applications to learning parities and juntas. In *53rd Annual IEEE Symp. Foundations of Computer Science (FOCS)*, pages 11–20, 2012.
- [34] G. Valiant. Finding correlations in subquadratic time, with applications to learning parities and the closest pair problem. *J. ACM*, 62(2):13, 2015.