National Technical University of Athens
School of Electrical and Computer Engineering
Department of Computer Science

High dimensional Approximate r-nets with emphasis on vectors on a unit hypercube

Loukas Kavouras

Submitted to the Department Of Mathematics
Graduate Program in Logic, Algorithms and Computation
$\mu\Pi\lambda\forall$, November 2016

Abstract

The construction of r-nets offers a powerful tool in computational and metric geometry. We focus on high-dimensional spaces and present a new randomized algorithm which efficiently computes 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. More specifically, the best previously known LSH-based construction of Eppstein et al. [EHS15] is improved in terms of complexity by reducing the dependence on ϵ, provided that ϵ is sufficiently small. Our method does not require LSH but, instead, follows Valiant's [Val15] 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 kth nearest neighbor distance in time subquadratic in the size of the input.

Περίληψη

Σε αυτή τη διπλωματική, παρουσιάζουμε έναν αλγόριθμο για την κατασκευή προσεγγιστικών r-nets σε Ευκλείδιο χώρο υψηλής διάστασης. Δεδομένου ενός μετρικού χώρου $X, |X| = n$, ένα r-net είναι ένα υποσύνολο N των αρχικών σημείων, τέτοιο ώστε τα σημεία που ανήκουν στο N έχουν απόσταση τουλάχιστον r, και όλα τα υπόλοιπα σημεία του σημειοσυνόλου απέχουν απόσταση από τα σημεία του N το πολύ r. Για την κατασκευή r-net, έχουν προταθεί διάφοροι αλγόριθμοι, οι οποίοι έχουν χρόνο τερματισμού τετραγωνικό στο πλήθος του σημειοσυνόλου ή εκθετικό στη διάσταση του μετρικού χώρου, με ανάλυση χειρότερης περίπτωσης. Οι τεχνικές που χρησιμοποιούνται συχνότερα είναι αυτή της άπληστης μεθόδου, καθώς και της δημιουργίας πλεγμάτων σε συνδυασμό με κατακερματισμό και κουβάδιασμα. Τέτοιοι αλγόριθμοι δεν μπορούν να θεωρηθούν αποδοτικοί σε περιπτώσεις μεγάλου πλήθους σημείων και σε περιπτώσεις μετρικών χώρων με υψηλή διάσταση. Μια αποδοτική προσέγγιση για το πρόβλημα της κατασκευής r-net σε υψηλή διάσταση είναι ο αλγόριθμος των [EHS15], οποίος βασίζεται στο LSH (Locality Sensitive Hashing). Ο αλγόριθμός τους είναι πιθανοκρατικός και υπολογίζει προσεγγιστικά r-net, με μεγάλη πιθανότητα. Ο προσεγγιστικός λόγος είναι $1 + \epsilon$, για κάθε $\epsilon > 0$, και η χρονική πολυπλοκότητα είναι $\tilde{O}(dn^{2-\Theta(\epsilon)})$, για κατάλληλα μικρά ϵ, όπου το $\tilde{O}$ κρύβει πολυλογαριθμικούς παράγοντες. Ο αλγόριθμος που αναπτύσσουμε για την κατασκευή r-nets βελτιώνει το αποτέλεσμα των [EHS15] όσο αφορά την εξάρτηση από το ϵ, για κατάλληλα μικρά ϵ. Συγκεκριμένα, η πολυπλοκλότητα του αλγορίθμου είναι $\tilde{O}(dn^{2-\Theta(\sqrt{\epsilon})})$ και υπολογίζει $(1 + \epsilon)r$-nets με μεγάλη πιθανότητα. Επιπλέον, η μέθοδος που χρησιμοποιούμε δεν βασίζεται στο LSH, αντιθέτως εκμεταλλεύεται φαινόμενα που εμφανίζονται σε υψηλές διαστάσεις. Η προσέγγισή μας ακολουθεί αυτή του Valiant [Val15], για την επίλυση του προβλήματος του προσεγγιστικά κοντινότερου γείτονα. Αρχικά ανάγουμε το πρόβλημά του υπολογισμολού του r-net για αυθαίρετα διανύσματα με Ευκλείδια απόσταση στο ίδιο πρόβλημα για μοναδιαία διανύσματα και ακολουθούν διάφορες μετατροπές του προβλήματος όπως μετασχηματισμοί των μοναδιαίων διανυσμάτων σε διανύσματα με στοιχεία 1 ή -1, μετάφραση της Ευκλείδιας απόστασης σε εσωτερικό γινόμενο, και εμβάπτιση του σημειοσυνόλου έτσι ώστε να μπορούμε να ξεχωρίσουμε ¨μακρινά' και ¨κοντινά' σημεία. Όλες αυτές οι αναγωγές απαιτούν αποδείξεις ορθότητας, που εγγυώνται ότι θα έχουμε το επιθυμητό αποτέλεσμα, με μεγάλη πιθανότητα, και ότι το συσσωρευτικό σφάλμα, που προκύπτει από την ακολουθία των μετασχηματισμών, είναι στα επιτρεπτά όρια. Στο τελικό στάδιο του αλγορίθμου εκμεταλλευόμαστε γρήγορο πολλαπλασιασμό πινάκων. Ο αλγόριθμός μας μπορεί να χρησιμοποιηθεί σαν υπορουτίνα στο πλαίσιο Net and Prune και να επιλύσει αποδοτικά σε χώρο υψηλής διάστάσης προβλήματα, όπως το k-center και k-th nearest neighbor distance.

Acknowledgements

I would first like to thank my supervisor Ioannis Z. Emiris, Georgia Avarikioti and Ioannis Psarros, since this thesis is based on a joint work with them. This accomplishment would not have been possible without them.

I would also like to thank to my parents for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis.

Contents

Chapter 1

Introduction

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, nets provide 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. In particular, net-trees can be interpreted as an extension of (compressed) quadtrees to more abstract settings.

Computing nets is closely related to k-center clustering. In the metric k-center problem, one is given a metric space X, $(X, \|\cdot\|)$, $|X| = n$ and an integer k, and the objective is to place k centers so as to minimize the maximum distance of a point to a center. A reduction from the dominating set problem shows that it is NP-hard to approximate k-center within a factor < 2. However, there is a simple greedy algorithm for this problem which uses the concept of r-nets to achieve the best possible approximation ratio of 2. The algorithm constructs an approximate r-net with k net points, which is also a 2 approximation to the k-center problem. We define approximate r-nets analogously. Formally,

Definition 1. *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. $\|p-q\|_2 \leq (1+\epsilon)r$.*

In this thesis, we consider the efficient construction of r-nets on high dimensional spaces. The efficiency comes in terms of time complexity as well as in terms of solution quality. Time complexity can be affected by big data phenomena and the curse of dimensionality, which refers to how certain algorithms may perform poorly in high dimensional data. Towards this end, we present a new randomized algorithm with complexity polynomial in the dimension and subquadratic in the number of points.

In terms of solution quality, our algorithm computes approximate r-nets with respect to Euclidean distance and succeeds with high probability. For any fixed $\epsilon > 0$, the approximation factor is $1+\epsilon$. The best previously known algorithm of Eppstein et al. [EHS15] also computes $(1+\epsilon)$-approximate r-nets with high probability and has running time $\tilde{O}(dn^{1+\frac{1}{(1+\epsilon)^2}})$, where $\tilde{O}$ hides polylogarithmic factors. For quite small $\epsilon > 0$, this is roughly $\tilde{O}(dn^{2-\Theta(\epsilon)})$ and we improve this result by reducing the dependence on ϵ to $\tilde{O}(dn^{2-\Theta(\sqrt{\epsilon})})$. Moreover, our algorithm is not based on locality sensitive hashing.

Introduced in work of Indyk and Motwani [IM98], the concept of locality sensitive hashing (LSH) uses a series of hashing functions that all have the property that close points have a higher probability of hashing to the same bucket. To perform a given query, one simply hashes the query point, and then checks the subset of the n data points that have also been hashed to those buckets.

However, our approach follows Valiant's [Val15] 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. Substantial work of this thesis forms part of the paper "High-dimensional approximate r-nets" [$SODA2017$], together with Georgia Avarikioti, Ioannis Z.Emiris, and Ioannis Psarros [AEKP17].

1.1 Existing Work

Finding r-nets can be addressed naively by considering all points of X unmarked and, while there remains an unmarked point p, the algorithm adds it to R and marks all other points within distance r from p. A similar approach was used by Gonzalez [Gon85] to 2-approximate the k-center problem in time $O(nk)$. The greedy algorithm chooses the first center arbitrarily, and at each step selects the

point, which is the farthest from all preceding centers as the new center. This was later improved to $O(n)$ time, for low dimensional Euclidean space [Har04], if k is sufficiently small ,using grids and hashing.

However, their 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 [HR15].

The aforementioned technique used by Gonzalez, can be generalized, when $k = n$, to lead to a permutation of the point set , called the greedy permutation. Formally, a permutation $\Pi =< \pi_1, \dots \pi_n >$ of the vertices of a metric space (X, d) is a greedy permutation (also called a farthest-first traversal or farthest point sampling) if each vertex π_i is the farthest in X from the set $\Pi_i = \{\pi_1, \dots, \pi_{i-1}\}$ of preceding vertices. Each prefix of a greedy permutation is an r-net, for r equal to the minimum distance between points in the prefix, and for every r, an r-net may be obtained as a prefix of a greedy permutation.

Greedy permutations may be computed for metric spaces in $O(n^2)$ time, and for graphs in the same time as all pairs shortest paths. The only previous improvement on the naive algorithm, by Har-Peled and Mendel [HP05] defines a concept of approximation for greedy permutations. They showed that $(1+\epsilon)$-greedy permutations can be computed in $O(n \log n)$ time in metric spaces ; these are permutations $\Pi =< \pi_1, \dots \pi_i >$ for which there exists a sequence of numbers $r_1 \geq r_2 \geq \dots$such that

1. the maximum distance of a point of X from Π_i is in the range $[r_i, (1 + \epsilon)r_i]$,
2. the distance between every two points $u, v \in \Pi_i$ is at least r_i.

This is satisfactory when doubling dimension of the metric space is constant, but requires a vast amount of resources when it is high. More specifically, the overall running time of the algorithm is $O(2^{ddim} n \log n)$, thus suffers from the curse of dimensionality.

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 [EHS15], which uses the Locality Sensitive Hashing (LSH) method of [AI08]. The resulting time complexity is $\tilde{O}(dn^{2-\Theta(\epsilon)})$, where $\epsilon > 0$ is quite small and $\tilde{O}$ 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 [GIV01]. Recently, Valiant [Val12], [Val15] 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 [Val15] is inspired by high dimensional phenomena. One main step of the algorithm is that of lifting the pointset up to a higher dimension.

Efficient construction of r-nets in high dimensional spaces may lead to efficient solutions for several problems. In [HR15], they describe a framework that contains problems, satisfying specific properties, that can be solved efficiently by algorithms, using r-nets as a subroutine. Our extension of r-nets achieves the best high dimensional solution for the kth nearest neighbor distance problem and leads to efficient solutions for the k-center problem. However, the further extension of net and prune framework to other problems is challenging, due to the hardness of finding efficient deciders for these problems in high dimensional spaces.

1.2 Our Contribution.

We present a new randomized algorithm that computes approximate r-nets in time subquadratic in the number of points and polynomial in the dimension, and improves upon the complexity of the best known algorithm. Our method does not employ LSH and, with probability $1 - o(1)$, it returns $R \subset X$, 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. The reduction can be relatively easily accomplished by adding a rather large randomly chosen vector v to all other vectors, then normalizing the vectors so as to have unit norm. Provided the vector v has magnitude significantly more than the maximum magnitude of all the vectors of interest, and the dimensionality of the space is sufficiently high so as to guarantee that v is nearly orthogonal to the chords connecting all pairs of the vectors of interest, this operation will simply scale all distances by roughly the same factor.

Then, depending on the magnitude of the distance r, an algorithm handling "small" distances or an algorithm handling "large" distances is called. These algorithms reduce the problem of computing

r-nets on unit vectors under Euclidean distance to that of finding an r-net for unit vectors under inner product. The "law of cosines" will allow us to translate between multiplicative $1+\epsilon$ bounds on the distance r and additive $c\epsilon$ bounds on the inner product, for c constant, provided that r is not too small.

If r is too small for an additive guarantee on the inner product to correspond to a meaningful multiplicative guarantee on r we distinguish between the cases $r \geq \frac{1}{n^{0.9}}$ and $r < \frac{1}{n^{0.9}}$. In the first case, we can achieve a multiplicative gap amplification between distances which immediately reduces the problem to the case that can be solved by the "law of cosines".

The second case is not straightforward and the approach of [Val15] can not be adapted. It 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.

Next, we convert the vectors having unit norm into vectors with entries $\{-1, +1\}$. This transformation is necessary in order to apply the Chebyshev embedding of [Val15], an embedding that reduces 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 in the case where the number of "small" distances is large. Then, we apply a modified version of the `Vector Aggregation` algorithm of [Val15], that exploits fast matrix multiplication, so as to achieve the desired running time.

In short, we extend Valiant's framework [Val15] and we compute r-nets in time $\tilde{O}(dn^{2-\Theta(\sqrt{\epsilon})})$, thus improving on the exponent of the LSH-based construction [EHS15], when ϵ is small enough. This improvement by $\sqrt{\epsilon}$ in the exponent is the same as the complexity improvement obtained in [Val15] 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 in approximation algorithms. In particular, our extension of r-nets in high dimensional Euclidean space can be plugged in the framework of [HR15]. The new framework has many applications, notably the kth nearest neighbor distance problem, which we solve in $\tilde{O}(dn^{2-\Theta(\sqrt{\epsilon})})$.

Chapter 2

Approximate ρ-net for vectors on a unit hypercube

In this chapter, we present an algorithm for computing an approximate net with respect to the inner product for a set of unit vectors. We begin by defining the notion of approximate ρ-nets formally:

Definition 2. *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. $\langle x, p \rangle \geq \rho - \epsilon$.*

Note that "far" points tend to have small inner product and "close" points tend to have "big" inner product. 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.

Our basic tool is based on the Vector Aggregation Algorithm by [Val15]. Their approach is motivated by the simple observation that if some columns of X have "big" inner product (the corresponding vectors are close), then we can compress X, by simply aggregating sets of columns. If one randomly partitions the n columns into, say, $n^{2/3}$ sets, each of size $n^{1/3}$, and then replaces each set of columns by a single vector, each of whose entries is given by the sum (over the real numbers) of the corresponding entries of the columns in the set, then we have shrunk the size of the matrix from $d \times n$, to a $d \times n^{2/3}$ matrix,

Z. This step is required to reduce the runtime of the algorithm which uses fast matrix multiplication of the matrices Z^T, Z forming matrix W, which still contains the information of close and far vectors.

Following the exposition of [Val15], 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. *For any $X = [x_1, \ldots, x_n], X' = [x'_1, \ldots, x'_n] \subset \mathbb{R}^{d \times n}$, 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. $|\langle x_i, x'_j \rangle| \geq \rho$.*

We will now present `Vector Aggregation` algorithm, which is a slight modification of Valiant's algorithm. 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: $X = [x_1, \ldots, x_n] \in \mathbb{R}^{d \times n}$, $X' = [x'_1, \ldots, x'_n] \in \mathbb{R}^{d \times n}$, $\alpha \in (0, 1)$, $\tau > 0$.

Output: $n \times n^{1-\alpha}$ matrix W and random partition $S_1, \ldots, S_{n^{1-\alpha}}$ of $\{x_1, \ldots, x_n\}$.

- Randomly partition $[n]$ into $n^{1-\alpha}$ disjoint subsets, each of size n^α , denoting the sets $S_1, \ldots, S_{n^{1-\alpha}}$.
- For each $i = 1, 2, \ldots, 78 \log n$:
  - Select n coefficients $q_1, \ldots, q_n \in \{-1, +1\}$ at random.
  - Form the $d \times n^{1-\alpha}$ matrix Z^i with entries $z^i_{j,k} = \sum_{l \in S_k} q_l \cdot x'_{j,l}$
  - $W^i = X^T Z^i$

- Define the $n \times n^{1-\alpha}$ matrix W with $w_{i,j} = quartile(|w_{i,j}^1|, \dots |w_{i,j}^{78 \log n}|)$.
- Output W and $S_1, \dots, S_{n^{1-\alpha}}$.

Theorem 4. *Let $X \in \mathbb{R}^{d \times n}$, $X' \in \mathbb{R}^{d \times n}$, $\alpha \in (0,1)$, $\tau > 0$ the input of* `Vector Aggregation`*. Then, the algorithm returns a matrix W of size $n \times n^{1-\alpha}$ and a random partition $S_1, \dots, S_{n^{1-\alpha}}$, 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 $\tilde{O}(dn + n^{2-\alpha} + MatrixMul(n \times d, d \times n^{1-\alpha}))$.

The proof Theorem 4 relies on the following extremely crude anti-concentration lemma to argue that if an entry $w_{j,k}^i$ of W contains a contribution from a pair of columns with large inner product, then with a reasonable probability over the random choice of $q_1, \dots, q_n$, the entry $w_{j,k}^i$ will not be too small.

Lemma 5 (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,*

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

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

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

Otherwise,

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

□

Proof of Theorem 4. Notice that

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

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

$$\mathrm{E}[w^i_{j,k}] = 0.$$

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

$$Var(w^i_{j,k}) = \mathrm{E}[(w^i_{j,k})^2] \leq n^{2\alpha}\tau^2$$

By Chebyshev's inequality:

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

With m repetitions, the number of successes N, that is the number of indices i for which $|w^i_{j,k}| \leq 3 \cdot n^\alpha \tau$, 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 $|w^i_{j,k}| \geq 3 \cdot n^\alpha \tau$. 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 5, with probability 1/2, $|w^i_{j,k}| \geq 3 \cdot n^\alpha \tau$. We consider as bad event the event that for j, l, more than 75% of the repetitions fail, that is $|w^i_{j,k}| \leq 3 \cdot n^\alpha \tau$. 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 $MatrixMul(n \times d, d \times n^{1-a})$.

□

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

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

- $C \leftarrow \emptyset$, $F_1 \leftarrow \emptyset, F_2 \leftarrow \{x_1, \ldots, x_n\}$
- 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, $S_1, \ldots, S_{n^{1-\alpha}}$.
- For each of the remaining rows $i = 1, \ldots$:
  - 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. $x'_k \neq x_i$ and $|\langle x_i, x'_k \rangle| \geq \tau$, delete row k and $F_2 \leftarrow F_2 \backslash \{x_i\}$.
  - $C \leftarrow C \cup \{x_i\}$
- Output indices of C and $F \leftarrow \{F_1 \cup F_2\}$.

Algorithm `Crude ApprxNet` takes as input the output of a powerful embedding, namely the Chebyshev embedding. This embedding allows us to distinguish between inner products of "close" and "far" vectors.

In order for `Vector Aggregation` to give the desired results, we must preprocess the pointset with

a "sparsifying" step, which helps us to compute the centers of a crude approximate r-net and the far points with high probability. We begin by choosing a point x_i uniformly at random and adding it to the net. We delete all columns j, such that $|\langle x_i, x'_j\rangle|$ is "big", since this means that these two points are close and we should add them both to the net. If there is not such a column, the point is definitely "far" from all the other points and we add it to the set F. This step is repeated for $n^{0.5}$ times.

Suppose that all points chosen in all repetitions have many neighbors. Then, the expected number of points we delete is greater than n and we end up with an exact crude net. Now, suppose that the previous hypothesis does not hold. Then, the number of points with "big" inner product will be less than $n^{1.7}$, with high probability, before `Vector Aggregation` is called. Thus, the pointset will have sufficiently "small" distances.

Theorem 6. *On input $X = [x_1, \ldots, x_n] \in \mathbb{R}^{d\times n}$, $X' = [x'_1, \ldots, x'_n] \in \mathbb{R}^{d\times n}$, $\alpha \in (0, 1)$, $\tau > 0$,* `Crude ApprxNet`*, computes a crude $3n^{\alpha}$-approximate τ-net for X, X', following the notation of Definition 3. The algorithm costs time:*

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

and succeeds with probability $1 - O(1/n^{0.2})$. Additionally, it outputs a set $F \subseteq C$ with the following property: $\{x_i \mid \forall x_j \neq x_i \ |\langle x_j, x_i\rangle| < \tau\} \subseteq F \subseteq \{x_i \mid \forall x_j \neq x_i \ |\langle x_j, x_i\rangle| < n^a\tau\}$.

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 ith iteration. Towards proving correctness, suppose first that $\mathrm{E}[X_i] > 2n^{0.5}$ for all $i = 1, \ldots n^{0.5}$. 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 $\mathrm{E}[X_j] \leq 2n^{0.5}$. 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 4 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 a 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 $|\langle c, p\rangle| < \tau$. In the second iteration, over the columns of W, notice that by Theorem 4, for any two centers c, c' we have $|\langle c, c'\rangle| < 3n^{\alpha}\tau$. □

The problem of computing ρ-nets for the inner product of unit vectors reduces to the less natural problem of Definition 3, 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 [Cha02]. Then, we apply the Chebyshev embedding of [Val15] in order to achieve gap amplification, and finally we call algorithm `Crude ApprxNet`, which will now return a proper ρ-net with additive error.

Theorem 7 ([Val15])**.** *There exists an algorithm 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, δ, where X is a matrix whose columns have unit norm, with probability $1 - o(1/n^2)$, for all pairs $i, j \in [n]$, $\left|\langle Y_i, Y_j\rangle/d' - \left(1 - 2 \cdot \cos^{-1}(\langle X_i, X_j\rangle)/\pi\right)\right| \leq \delta$, where X_i, Y_i denote the ith column of X and Y respectively. Additionally, the runtime of the algorithm is $O(\frac{dn \log n}{\delta^2})$.*

The following theorem provides a randomized embedding, namely the Chebyshev embedding, that reduces the magnitude of the inner product of "far" vectors, while preserving the magnitude of the inner product of "close" vectors. Such transformation could have been achieved by a simple embedding, in which each vector is replaced by its degree q tensor power, leading to an exponent amplification of the multiplicative gap between "big" and "small" inner products. However, with this approach, we will obtain an algorithm with an undesired runtime of $O(n^{2-\Theta(\epsilon)})$, for quite small ϵ.

Valiant [Val15] overcomes this difficulty by suggesting two embeddings f, g, which create two copies of our vectors, and project each according to a different embedding, and then consider inner products across these two sets of vectors. We define this embedding in the setting in which the vectors in

question have values in $\{-1,+1\}$; this uniformity ensures that the entries of the vectors returned by the embedding have the same magnitudes, and hence are amenable to Chernoff bounds to guarantee that the inner products. The statement is almost verbatim that of [Val15, Prop.6] except that we additionally establish $1-o(1/n)$ probability of success instead of $1-o(1)$ as stated in [Val15].

The proof is the same, but since we claim stronger guarantees on success probability, we include the complete proof. While $1-o(1/n)$ probability of success is enough for our purposes, even better probability bounds can be achieved.

Theorem 8. *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 $T_q\Big(\frac{\langle X_i, X'_j\rangle/d' - \tau^-}{\tau^+ - \tau^-}2-1\Big)\cdot d'\cdot(\tau^+-\tau^-)^q/2^{3q-1}$, where T_q is the degree-q Chebyshev polynomial of the first kind. The algorithm runs in time $O(d'\cdot n\cdot q)$.*

Proof. The fact that all inner products are concentrated within $\pm\sqrt{m}\log n$ 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,

$$\Pr[\,|\sum_{i=1}^{m'} X_i - \mathrm{E}[\sum_{i=1}^{m'} X_i]| > \sqrt{m'}\log n] \le 2\cdot exp(-\Theta(\log^2 n)) = o(1/n^3).$$

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 $x = \langle u, u'\rangle/m$, we argue that by [Val15, Lemma 3.3], $\mathrm{E}[v,v'] = m'\sum_{i=1}^{q}\frac{x-c_i}{2}$ (1), where c_i is the location of the ith root of the qth 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 $\mathrm{E}[s_v(i)t_{v'}(i)] = \frac{x-c_i}{2}$, 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, $E[s_v(i)t_{v'}(i)] = \langle v, v' \rangle / m = x$. Steps (2)–(4) ensure that with the remaining 1/2 probability, $\mathrm{E}[s_v(i)t_v(i)] = \frac{1-c_i}{2}(1) - \frac{1+c_i}{2}(-1) = -c_i$. Hence, in aggregate over the randomness of Steps (1)–(4), $\mathrm{E}[s_v(i)t_{v'}(i)] = x/2 - c_i/2i$, 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 qth Chebyshev polynomial T_q. Note that the qth 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 $\alpha = \frac{2}{\tau^+ - \tau^-}$.

□

`Inner product ApprxNet`

Input: $X = [x_1, \ldots, x_n]$ 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:
  - $C \leftarrow \emptyset$, $F \leftarrow \emptyset$, $W \leftarrow \{x_1, \ldots, x_n\}$
  - 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 7 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 8 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 9. *The algorithm* `Inner product ApprxNet`*, on input* $X = [x_1, \ldots, x_n]$ *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 2. 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:* $\{x_i \mid \forall x_j \neq x_i \ \langle x_j, x_i\rangle < \rho - \epsilon\} \subseteq F \subseteq \{x_i \mid \forall x_j \neq x_i \ \langle x_j, x_i\rangle < \rho\}$*.*

Theorem 10 ([Cop97])**.** *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 11. *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 10, the total cost is:

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

□

Fact 12. *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 13. *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

$$\cos^{-1}(\rho-\epsilon)-\cos^{-1}(\rho)=\int_{\rho-\epsilon}^{1}\frac{1}{\sqrt{1-x^2}}\mathrm{d}x-\int_{\rho}^{1}\frac{1}{\sqrt{1-x^2}}\mathrm{d}x=$$

$$=\int_{\rho-\epsilon}^{\rho}\frac{1}{\sqrt{1-x^2}}\mathrm{d}x=\int_{0}^{\epsilon}\frac{1}{\sqrt{1-(\rho-\epsilon+y)^2}}\mathrm{d}y\geq\int_{0}^{\epsilon}\frac{1}{\sqrt{1-(\rho-\epsilon)^2}}\mathrm{d}y=\frac{\epsilon}{\sqrt{1-(\rho-\epsilon)^2}}\geq\epsilon.$$

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

$$\cos^{-1}(\rho-\epsilon)-\cos^{-1}(\rho)=\int_{-1}^{-1+\epsilon}\frac{1}{\sqrt{1-x^2}}\mathrm{d}x\geq\frac{\epsilon}{\sqrt{2\epsilon-\epsilon^2}}\geq\epsilon/2.$$

□

Proof of Theorem 9. 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 $x_1,\ldots,x_{d+2}$ such that $\forall i\neq j\in[d+2]$ we have $\langle x_i,x_j\rangle<0$. Then, there exist $\lambda_1,\ldots,\lambda_{d+1}\in\mathbb{R}$ 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 $\sum_{i\in I}\lambda_i x_i=\sum_{j\in J}-\lambda_j x_j \implies 0\leq\langle\sum_{i\in I}\lambda_i x_i,-\sum_{j\in J}\lambda_j x_j\rangle=-\sum_{i\in I,j\in J}\lambda_i\lambda_j\langle x_i,x_j\rangle<0$ 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 7, 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 13, 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 8, with probability $1 - o(1)$,

$$\langle Y_i, Y_j\rangle \le \tau^+ d' \le \Longrightarrow |\langle Z_i, Z_j\rangle| \le d''\frac{2^q}{2^{3q-1}} + \sqrt{d''}\log n \le 3n^{0.16}$$

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

$$|\langle Z_i, Z'_j\rangle| \ge d'' \cdot T_q\Big(1 + 2\frac{\delta}{\tau^+ + 1}\Big)\frac{2^q}{2^{3q-1}} - \sqrt{d''}\log n > \frac{1}{2}\cdot T_q\Big(1 + 2\frac{\delta}{\tau^+ + 1}\Big)\cdot n^{0.16}$$

for large enough n.

Then, by Fact 12,

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

where some of the inequalities hold for large enough n.

Now, by Theorems 7, 8, 6 and Corollary 11 the time complexity is $\tilde{O}(dn + n^{2-\sqrt{\epsilon}/600})$, if we set as γ in Corollary 11 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. □

Chapter 3

Approximate r-nets in high dimension under Euclidean distance

In this chapter, 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.

First, 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 [Val15], 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. 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 vector. Our exposition is even simpler since we can directly employ the analogous theorem from [Val15].

Standardize

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

Output: 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 our algorithm computes an r-net of X given a ρ-net of Y.

- Define two d-dimensional vectors X_{n+1}, X_{n+2}, s.t. $r' = X_{n+1} - X_{n+2}$ and $\|r'\| = r$, and let matrix X' denote the concatenation of X, X_{n+1} and X_{n+2} with size $d \times n + 2$.

- Perform a Johnson-Lindenstrauss transformation on the columns of X', projecting them to dimension m', so as to yield matrix X''.
- Let c denote the magnitude of the largest column of X''. Choose a random m'-dimensional vector u of magnitude $8c/\epsilon$.
- Let matrix Y be the result of adding u to each column of X'' and normalizing all columns so as to have unit norm.
- Define $\rho := \|Y_{n+1} - Y_{n+2}\|$ to be the new distance parameter.

Theorem 14. *[Val15] The algorithm* `Standardize`*, when given as 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 $1 - o(1/poly(n))$ 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*

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

We now show what is implied for our case by Theorem 14.

Corollary 15. *The algorithm* `Standardize`*, when given as 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 $(1+\epsilon)$-net of X, with probability $1 - o(1/poly(n))$.*

Proof. 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 14 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 $1 - o(1/poly(n))$, which immediately implies Corollary 15:

- 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 $\|Y_i - Y_j\| \geq (1+\epsilon/2)\rho$.

□

The remainder of this chapter is dedicated into proving 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 the previous chapter. Moreover, we identify the subset of the r-net which contains the centers that are approximately far from any other point. Formally,

Definition 16. *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:* $\{x \in X \mid \forall x \neq y \in X \ \|x-y\| > (1+\epsilon)r\} \subseteq F \subseteq \{x \in X \mid \forall x \neq y \in X \ \|x-y\| > r\}$.

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 [Val15, 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.

`ApprxNet(Large radius)`

Input: $X = [x_1, \ldots, x_n]^T$ 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 $q = \left\lfloor \frac{\pi}{2\cos^{-1}(1-r^2/2)} \right\rfloor$ uniformly random vectors $v_1, \ldots, v_q$ and for all $j \in [n]$, set

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

  where X_j is the jth 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 17. *For any constant $\epsilon \in (0, 1/2]$, $X \subset \mathbb{S}^{d-1}$ s.t. $|X| = n$, algorithm* `ApprxNet(Large radius)` *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 $\mathrm{E}[sign(X_i^T v \cdot X_j^T v)] = 1 - \frac{2\theta(X_i, X_j)}{\pi}$, 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,

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

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

$$\|X_i - X_j\| \leq r \implies \theta(X_i, X_j) \leq \theta_r \implies \mathrm{E}[\langle Z_i, Z_j\rangle] \geq d(1 - 2\theta_r/\pi)^q$$

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

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 $\|X_l - X_k\| \geq (1+\epsilon)r$,

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

By a union bound over Chernoff bounds, since $d = \log^3 n$, with probability $1 - o(1/poly(n))$, 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 $\rho = (1-2\cdot\theta_r/\pi)^q$ and approximation error $\epsilon/100$.

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

□

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 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 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 ApprxNet(Large radius) algorithm. Then, we recurse on T_i' which contains a constant fraction of points from $|S_i|$. Then, we cover points in $S_i \setminus (T_i \cup T_i')$ and points which do not belong to any S_i.

`ApprxNet(Small radius)`

Input: $X = [x_1, \ldots, x_n]^T$ 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 $p_1, \ldots, p_n$ which wlog correspond to $x_1, \ldots, x_n \in \mathbb{R}^d$.
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 $0 \leq d_1 \leq \ldots \leq d_{|S_i|}$ 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 $p_1, \ldots, p_n$.
   (a) Remove from F all points which cover at least one point from $\bigcup_i W_i$ or $\bigcup_i K_i$.
   (b) Delete all points $(\bigcup_i T_i) \setminus (\bigcup_i R_i)$, and $(\bigcup_i T_i') \setminus (\bigcup_i R_i')$.
   (c) For each i delete all points in W_i covered by R_i, or covered by R_i'.
   (d) For each i delete all points in K_i covered by R.
   (e) 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 18. *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 [Das03] we have that for $v \in \mathbb{S}^{d-1}$ chosen uniformly at random:

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

Hence, $\mathrm{E}[|\{x_k, x_j \in S_i \mid \|x_k - x_j\| \geq n^{0.6}r \text{ and } \|p_k - p_j\| \leq n^{0.4}r\}|] \leq |S_i|^2 \cdot \frac{\sqrt{ed}}{n^{0.2}}$, and the probability

$$\Pr[|\{x_k, x_j \in S_i \mid \|x_k - x_j\| \geq n^{0.6}r \text{ and } \|p_k - p_j\| \leq n^{0.4}r\}| \geq |S_i|^{1.95}] \leq |S_i|^{0.05} \cdot \frac{\sqrt{ed}}{n^{0.2}} \leq \frac{\sqrt{ed}}{n^{0.15}}.$$

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 jth 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:

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

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

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 ith 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 $|\{j < i \mid p_j \in [p_i - r, p_i]$ and K_j 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 jth 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 $o(1/n^{0.04})$. □

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

`ApprxNet`

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

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

- 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 19. *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 17, 18, 14. □

Before we proceed to the next chapter, we show how to store and delete the set F of far points. This is a direct implication of the previous results.

Theorem 20. *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`*, which 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 17, 18, 15, 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 $F' \subseteq \{x_1, \ldots, x_n\}$, 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 $1 - o(n^{0.04})$. □

Chapter 4

Applications and future work

Concerning applications, in [HR15], 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 . 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 n.

The cornerstone of this framework is to find an algorithm that can bound the optimal solution of a problem in an appropriate interval in order to perform in it binary search for the desired approximation of the optimal solution. We present the algorithm Net and Prune of [HR15], 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 21. *Given a function $f : X \to \mathbb{R}$, we call a decider procedure a $(1+\epsilon)$-decider for f, if for any $x \in X$ and $r > 0$, 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 $X_{i+1} =$`DelFar`$(2l_i/3, X_i, \frac{3}{2})$
    - If cl_i is too small, $X_{i+1} =$`ApprxNet`$(4l_i, X_i, \frac{3}{2})$
  - $i = i + 1$,

Let us denote as $|X_i^{\leq l}|$ 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 22. *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 $(i + 1 < k)$, lets assume that $x_1, x_2, \ldots, x_m$ 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 [HR15] and Theorem 20). 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. □

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

Proof. Let I be the set of indices of the `ApprxNet` iterations up to the ith iteration. Similarly, let I' be the set of iterations where `DelFar` is called.
If `ApprxNet` was called in the jth iteration, then X_j is at most a $6l_j$-drift of X_{j-1}, therefore $|f(X_j) - f(X_{j-1})| \leq 12 l_j$. Also, if `DelFar` is called in the jth iteration, then $f(X_j) = f(X_{j-1})$ (by Theorem 20). Let m=maxI, we have that,

$$|f(X_i) - f(X_0)| \leq \sum_{j=1}^{i} |f(X_j) - f(X_{j-1})| = \sum_{j \in I} |f(X_j) - f(X_{j-1})| + \sum_{j \in I'} |f(X_j) - f(X_{j-1})|$$

$$\leq \sum_{j \in I} 12 l_j + \sum_{j \in I'} 0 \leq 12 l_m \sum_{j=0}^{\infty} \left(\frac{1}{4}\right)^j \leq 16 l_m \leq 16 l_i$$

,where the second inequality holds since for every $j < i$, in the beginning of the jth 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 $l_j \geq 4 l_i$. □

Lemma 24. *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 $m_1, m_2, \ldots, m_t$ 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 $\eta l_{m_1} < c l_{m_1} < f(X_{m_1-1})$ 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 $c l_{m_i} < f(X_{(m_i)-1}) = f(X_{m_{i-1}})$. Thus, by the induction hypothesis and Lemma 23 we have,

$$l_{m_i} < \frac{f(X_{m_{i-1}})}{c} \leq \frac{f(X_0) + 16 l_{m_{i-1}}}{c} \leq \frac{f(X_0) + 16 f(X_0)/\eta}{c} = \frac{1 + 16/\eta}{c} f(X_0) = f(X_0)/\eta$$

□

Therefore, if we set $c = 64$ we have $\eta = 48$, thus by Lemma 23 and Lemma 24,

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

Corollary 25. *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 26. *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 25 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 $c l_i$ too large, then the returned interval is $[\frac{l_i}{3}, 2c l_i]$ and its spread is 384. □

4.1 k-th nearest neighbor distance

Let us focus on the problem of approximating the *kth nearest neighbor distance*.

Definition 27. *Let $X \subset \mathbb{R}^d$ be a set of n points, approximation error $\epsilon > 0$, and let $d_1 \leq \ldots \leq d_n$ be the nearest neighbor distances. The problem of computing an $(1+\epsilon)$-approximation to the kth nearest neighbor distance asks for a pair $x, y \in X$ such that $\|x - y\| \in [(1-\epsilon)d_k, (1+\epsilon)d_k]$.*

Before we present the approximate decider for the k-th nearest neighbor problem, we show a direct implication of the approximate net construction from previous chapters. More specifically, we prove how to identify and delete far points in high dimensional Euclidean space.

`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 `DelFar`$(X, \frac{r}{1+\epsilon/4}, \epsilon/4)$ and store its output in W_1.
- Call `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 28. *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 26 holds, and we compute a constant spread interval $[x, y]$ containing the optimal value, with high probability. We then

apply binary search on values $x, (1+\epsilon)x, (1+\epsilon)^2x, \ldots, y$ 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 $1 - o(1)$. □

To the best of our knowledge, this is the first high dimensional solution for this problem. Setting $k = n$ and applying Theorem 28 one can compute the *farthest nearest neighbor* in $\tilde{O}(dn^{2-\Theta(\sqrt{\epsilon})})$ with high probability.

4.2 *k*-center clustering and greedy permutations

Concerning future work, let us start with the problem of finding a greedy permutation. A permutation $\Pi =< \pi_1, \pi_2, \cdots >$ 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 set $\Pi_{i-1} =< \pi_1, \ldots, \pi_{i-1} >$ of preceding vertices. The computation of r-nets is closely related to that of the greedy permutation.

The k-center clustering problem asks the following: given a set $X \subseteq \mathbb{R}^d$ and an integer k, find the smallest radius r such that X is contained within k balls of radius r. By [EHS15], 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.

The Corollaries below follow from Theorem 19, Lemma 3.5[EHS15] and Lemma 2.1[EHS15].

Corollary 29. *Let X be a set of n points in $\mathbb{R}^d$, $\epsilon \in (0,1)$ an error parameter and let Φ be the spread of the Euclidean metric on X. Then, one can compute in $O(dn^{2-\Theta(\sqrt{\epsilon})} \log \Phi)$ expected time a sequence that is a $(1+\epsilon)$-greedy permutation for the Euclidean metric on X, with high probability.*

Corollary 30. *Given a set X of n points in $\mathbb{R}^d$, an integer k and an error parameter $\epsilon \in (0,1)$, one can compute with high probability a $(2+\epsilon)$-approximation to the optimal k-center clustering in $O(dn^{2-\Theta(\sqrt{\epsilon})} \log \Phi)$, where Φ is the spread of the Euclidean metric on X.*

Bibliography

[AEKP17] A. Avarikioti, I. Z. Emiris, L. Kavouras, and I. Psarros. High-dimensional approximate r-nets. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, 2017.

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

[Cha02] M. Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 380–388, 2002.

[Cop97] D. Coppersmith. Rectangular matrix multiplication revisited. *J. Complex.*, 13(1):42–49, March 1997.

[Das03] A. Dasgupta, S. and Gupta. An elementary proof of a theorem of Johnson and Lindenstrauss. *Random Struct. Algorithms*, 22(1):60–65, 2003.

[EHS15] D. Eppstein, S. Har-Peled, and A. Sidiropoulos. Approximate greedy clustering and distance selection for graph metrics. *CoRR*, abs/1507.01555, 2015.

[GIV01] A. Goel, P. Indyk, and K.R. Varadarajan. Reductions among high dimensional proximity problems. In *Proc. 12th Symposium on Discrete Algorithms (SODA)*, pages 769–778, January 2001.

[Gon85] T. F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoret.Comp.Sci.*, 38(2-3):293–306, 1985.

[Har04] S. Har-Peled. Clustering motion. *Discrete & Computational Geometry*, 31(4):545–565, 2004.

[HP05] M. Har-Peled, S. and Mendel. Fast construction of nets in low dimensional metrics, and their applications. In *Proc. 21st Annual Symp. Computational Geometry*, SCG'05, pages 150–158, 2005.

[HR15] S. Har-Peled and B. Raichel. Net and prune: A linear time algorithm for euclidean distance problems. *J. ACM*, 62(6):44, 2015.

[IM98] P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. *In Proceedings of the ACM Symposium on Theory of Computing (STOC)*, 1998.

[Val12] G. Valiant. Finding correlations in subquadratic time, with applications to learning parities and juntas. In *53rd Annual IEEE Symp. on Foundations of Computer Science, FOCS 2012, 20-23, 2012*, pages 11–20, 2012.

[Val15] G. Valiant. Finding correlations in subquadratic time, with applications to learning parities and the closest pair problem. *J. ACM*, 62(2):13, 2015.