Low-Quality Dimension Reduction and High-Dimensional Approximate Nearest Neighbor*

Evangelos Anagnostopoulos[1], Ioannis Z. Emiris[2], and Ioannis Psarros[1]

1 **Department of Mathematics, University of Athens, Athens, Greece**
aneva@math.uoa.gr, i.psarros@di.uoa.gr

2 **Department of Informatics & Telecommunications, University of Athens, Athens, Greece**
emiris@di.uoa.gr

Abstract

The approximate nearest neighbor problem (ϵ-ANN) in Euclidean settings is a fundamental question, which has been addressed by two main approaches: Data-dependent space partitioning techniques perform well when the dimension is relatively low, but are affected by the curse of dimensionality. On the other hand, locality sensitive hashing has polynomial dependence in the dimension, sublinear query time with an exponent inversely proportional to $(1+\epsilon)^2$, and subquadratic space requirement.

We generalize the Johnson-Lindenstrauss Lemma to define "low-quality" mappings to a Euclidean space of significantly lower dimension, such that they satisfy a requirement weaker than approximately preserving all distances or even preserving the nearest neighbor. This mapping guarantees, with high probability, that an approximate nearest neighbor lies among the k approximate nearest neighbors in the projected space. These can be efficiently retrieved while using only linear storage by a data structure, such as BBD-trees. Our overall algorithm, given n points in dimension d, achieves space usage in $O(dn)$, preprocessing time in $O(dn \log n)$, and query time in $O(dn^{\rho} \log n)$, where ρ is proportional to $1 - 1/\log\log n$, for fixed $\epsilon \in (0,1)$. The dimension reduction is larger if one assumes that pointsets possess some structure, namely bounded expansion rate. We implement our method and present experimental results in up to 500 dimensions and 10^6 points, which show that the practical performance is better than predicted by the theoretical analysis. In addition, we compare our approach with E2LSH.

1998 ACM Subject Classification F.2.2 [Analysis of algorithms and problem complexity] Geometrical problems and computations

Keywords and phrases Approximate nearest neighbor, Randomized embeddings, Curse of dimensionality, Johnson-Lindenstrauss Lemma, Bounded expansion rate, Experimental study

Digital Object Identifier 10.4230/LIPIcs.SOCG.2015.436

1 Introduction

Nearest neighbor searching is a fundamental computational problem. Let X be a set of n points in $\mathbb{R}^d$ and let $d(p,p')$ be the (Euclidean) distance between any two points p and p'. The

* This research has been co-financed by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) – Research Funding Program: THALIS-UOA (MIS 375891). This work was done in part while Ioannis Z. Emiris was visiting the Simons Institute for the Theory of Computing, at UC Berkeley.

problem consists in reporting, given a query point q, a point $p \in X$ such that $d(p,q) \leq d(p',q)$, for all $p' \in X$ and p is said to be a "nearest neighbor" of q. For this purpose, we preprocess X into a structure called NN-structure. However, an exact solution to high-dimensional nearest neighbor search, in sublinear time, requires prohibitively heavy resources. Thus, many techniques focus on the less demanding task of computing the approximate nearest neighbor (ϵ-ANN). Given a parameter $\epsilon \in (0,1)$, a $(1+\epsilon)$-approximate nearest neighbor to a query q is a point p in X such that $d(q,p) \leq (1+\epsilon) \cdot d(q,p')$, $\forall p' \in X$. Hence, under approximation, the answer can be any point whose distance from q is at most $(1+\epsilon)$ times larger than the distance between q and its nearest neighbor.

Our contribution

Tree-based space partitioning techniques perform well when the dimension is relatively low, but are affected by the curse of dimensionality. To address this issue, randomized methods such as Locality Sensitive Hashing are more efficient when the dimension is high. One may also apply the Johnson-Lindenstrauss Lemma followed by standard space partitioning techniques, but the properties guaranteed are stronger than what is required for efficient approximate nearest neighbor search (cf. 2).

We introduce a "low-quality" mapping to a Euclidean space of dimension $O(\log \frac{n}{k}/\epsilon^2)$, such that an approximate nearest neighbor lies among the k approximate nearest neighbors in the projected space. This leads to our main Theorem 10, which offers a new randomized algorithm for approximate nearest neighbor search with the following complexity: Given n points in $\mathbb{R}^d$, the data structure, which is based on Balanced Box-Decomposition (BBD) trees, requires $O(dn)$ space, and reports an $(1+\epsilon)^2$-approximate nearest neighbor in time $O(dn^\rho \log n)$, where function $\rho < 1$ is proportional to $1 - 1/\ln\ln n$ for fixed $\epsilon \in (0,1)$ and shall be specified in Section 4. The total preprocessing time is $O(dn \log n)$. For each query $q \in \mathbb{R}^d$, the preprocessing phase succeeds with probability $> 1-\delta$ for any constant $\delta \in (0,1)$. The low-quality embedding is extended to pointsets with bounded expansion rate c (see Section 5 for definitions). The pointset is now mapped to a Euclidean space of dimension roughly $O(\log c/\epsilon^2)$, for large enough k.

We also present experiments, based on synthetic datasets that validate our approach and our analysis. One set of inputs, along with the queries, follow the "planted nearest neighbor model" which will be specified in Section 6. In another scenario, we assume that the near neighbors of each query point follow the Gaussian distribution. Apart from showing that the embedding has the desired properties in practice, we also implement our overall approach for computing ϵ-ANN using the `ANN` library and we compare with a LSH implementation, namely E2LSH.

The notation of key quantities is the same throughout the paper.

The paper extends and improves ideas from [25].

Paper organization

The next section offers a survey of existing techniques. Section 3 introduces our embeddings to dimension lower than predicted by the Johnson-Linderstrauss Lemma. Section 4 states our main results about ϵ-ANN search. Section 5 generalizes our discussion so as to exploit bounded expansion rate, and Section 6 presents experiments to validate our approach. We conclude with open questions.

2 Existing work

As it was mentioned above, an exact solution to high-dimensional nearest neighbor search, in sublinear time, requires heavy resources. One notable solution to the problem [21] shows that nearest neighbor queries can be answered in $O(d^5 \log n)$ time, using $O(n^{d+\delta})$ space, for arbitrary $\delta > 0$.

One class of methods for ϵ-ANN may be called data-dependent, since the decisions taken for partitioning the space are affected by the given data points. In [8], they introduced the Balanced Box-Decomposition (BBD) trees. The BBD-trees data structure achieves query time $O(c \log n)$ with $c \leq d/2\lceil 1 + 6d/\epsilon \rceil^d$, using space in $O(dn)$, and preprocessing time in $O(dn \log n)$. BBD-trees can be used to retrieve the $k \geq 1$ approximate nearest-neighbors at an extra cost of $O(d \log n)$ per neighbor. BBD-trees have proved to be very practical, as well, and have been implemented in software library `ANN`.

Another data structure is the Approximate Voronoi Diagrams (AVD). They are shown to establish a tradeoff between the space complexity of the data structure and the query time it supports [7]. With a tradeoff parameter $2 \leq \gamma \leq \frac{1}{\epsilon}$, the query time is $O(\log(n\gamma) + 1/(\epsilon\gamma)^{\frac{d-1}{2}})$ and the space is $O(n\gamma^{d-1} \log \frac{1}{\epsilon})$. They are implemented on a hierarchical quadtree-based subdivision of space into cells, each storing a number of representative points, such that for any query point lying in the cell, at least one of the representatives is an approximate nearest neighbor. Further improvements to the space-time trade offs for ANN, are obtained in [6].

One might directly apply the celebrated Johnson-Lindenstrauss Lemma and map the points to $O(\frac{\log n}{\epsilon^2})$ dimensions with distortion equal to $1 + \epsilon$ in order to improve space requirements. In particular, AVD combined with the Johnson-Lindenstrauss Lemma require $n^{O(\log \frac{1}{\epsilon}/\epsilon^2)}$ space which is prohibitive if $\epsilon \ll 1$ and query time polynomial in $\log n$, d and $1/\epsilon$. Notice that we relate the approximation error with the distortion for simplicity. Our approach (Theorem 10) requires $O(dn)$ space and has query time sublinear in n and polynomial in d.

In high dimensional spaces, data dependent data structures are affected by the curse of dimensionality. This means that, when the dimension increases, either the query time or the required space increases exponentially. An important method conceived for high dimensional data is locality sensitive hashing (LSH). LSH induces a data independent space partition and is dynamic, since it supports insertions and deletions. It relies on the existence of locality sensitive hash functions, which are more likely to map similar objects to the same bucket. The existence of such functions depends on the metric space. In general, LSH requires roughly $O(dn^{1+\rho})$ space and $O(dn^{\rho})$ query time for some parameter $\rho \in (0, 1)$. In [4] they show that in the Euclidean case, one can have $\rho = \frac{1}{(1+\epsilon)^2}$ which matches the lower bound of hashing algorithms proved in [23]. Lately, it was shown that it is possible to overcome this limitation with an appropriate change in the scheme which achieves $\rho = \frac{1}{2(1+\epsilon)^2-1} + o(1)$ [5]. For comparison, in Theorem 10 we show that it is possible to use $O(dn)$ space, with query time roughly $O(dn^{\rho})$ where $\rho < 1$ is now higher than the one appearing in LSH. One different approach [24] achieves near linear space but query time proportional to $O(dn^{\frac{2}{1+\epsilon}})$.

Exploiting the structure of the input is an important way to improve the complexity of nearest neighbor search. In particular, significant amount of work has been done for pointsets with low doubling dimension. In [14], they provide an algorithm for ANN with expected preprocessing time $O(2^{\dim(X)} n \log n)$, space $O(2^{\dim(X)} n)$ and query time $O(2^{\dim(X)} \log n + \epsilon^{-O(\dim(X))})$ for any finite metric space X of doubling dimension $\dim(X)$. In [16] they provide randomized embeddings that preserve nearest neighbor with constant probability, for points lying on low doubling dimension manifolds in Euclidean settings. Naturally, such an approach can be easily combined with any known data structure for ϵ-ANN.

In [10] they present random projection trees which adapt to pointsets of low doubling dimension. Like kd-trees, every split partitions the pointset into subsets of roughly equal cardinality; in fact, instead of splitting at the median, they add a small amount of "jitter". Unlike kd-trees, the space is split with respect to a random direction, not necessarily parallel to the coordinate axes. Classic *kd*-trees also adapt to the doubling dimension of randomly rotated data [26]. However, for both techniques, no related theoretical arguments about the efficiency of ϵ-ANN search were given.

In [19], they introduce a different notion of intrinsic dimension for an arbitrary metric space, namely its expansion rate c; it is formally defined in Section 5. The doubling dimension is a more general notion of intrinsic dimension in the sense that, when a finite metric space has bounded expansion rate, then it also has bounded doubling dimension, but the converse does not hold [13]. Several efficient solutions are known for metrics with bounded expansion rate, including for the problem of exact nearest neighbor. In [20], they present a data structure which requires $c^{O(1)}n$ space and answers queries in $c^{O(1)}\ln n$. Cover Trees [9] require $O(n)$ space and each query costs $O(c^{12}\log n)$ time for exact nearest neighbors. In Theorem 13, we provide a data structure for the ϵ-ANN problem with linear space and $O((C^{1/\epsilon^3}+\log n)d\log n/\epsilon^2)$ query time, where C depends on c. The result concerns pointsets in the d-dimensional Euclidean space.

3 Low Quality Randomized Embeddings

This section examines standard dimensionality reduction techniques and extends them to approximate embeddings optimized to our setting. In the following, we denote by $\|\cdot\|$ the Euclidean norm and by $|\cdot|$ the cardinality of a set.

Let us start with the classic Johnson-Lindenstrauss Lemma:

▶ **Proposition 1.** [18] *For any set $X\subset\mathbb{R}^d$, $\epsilon\in(0,1)$ there exists a distribution over linear mappings $f:\mathbb{R}^d\longrightarrow\mathbb{R}^{d'}$, where $d'=O(\log|X|/\epsilon^2)$, such that for any $p,q\in X$,*

$$(1-\epsilon)\|p-q\|^2\leq\|f(p)-f(q)\|^2\leq(1+\epsilon)\|p-q\|^2.$$

In the initial proof [18], they show that this can be achieved by orthogonally projecting the pointset on a random linear subspace of dimension d'. In [11], they provide a proof based on elementary probabilistic techniques. In [15], they prove that it suffices to apply a gaussian matrix G on the pointset. G is a $d\times d'$ matrix with each of its entries independent random variables given by the standard normal distribution $N(0,1)$. Instead of a gaussian matrix, we can apply a matrix whose entries are independent random variables with uniformly distributed values in $\{-1,1\}$ [2].

However, it has been realized that this notion of randomized embedding is somewhat stronger than what is required for approximate nearest neighbor searching. The following definition has been introduced in [16] and focuses only on the distortion of the nearest neighbor.

▶ **Definition 2.** Let (Y,d_Y), (Z,d_Z) be metric spaces and $X\subseteq Y$. A distribution over mappings $f:Y\to Z$ is a *nearest-neighbor preserving embedding* with distortion $D\geq 1$ and probability of correctness $P\in[0,1]$ if, $\forall\epsilon\geq 0$ and $\forall q\in Y$, with probability at least P, when $x\in X$ is such that $f(x)$ is a ϵ-ANN of $f(q)$ in $f(X)$, then x is a $(D\cdot(1+\epsilon))$-approximate nearest neighbor of q in X.

While in the ANN problem we search one point which is approximately nearest, in the k approximate nearest neighbors problem (ϵ-kANNs) we seek an approximation of the k

nearest points, in the following sense. Let X be a set of n points in $\mathbb{R}^d$, let $q \in \mathbb{R}^d$ and $1 \leq k \leq n$. The problem consists in reporting a sequence $S = \{p_1, \ldots, p_k\}$ of k distinct points such that the i-th point is an $(1+\epsilon)$-approximation to the i-th nearest neighbor of q. Furthermore, the following assumption is satisfied by the search routine of tree-based data structures such as BBD-trees.

▶ **Assumption 3.** *Let $S' \subseteq X$ be the set of points visited by the ϵ-kANNs search such that $S = \{p_1, \ldots, p_k\} \subseteq S'$ is the (ordered w.r.t. distance from q) set of points which are the k nearest to the query point q among the points in S'. We assume that $\forall x \in X \setminus S'$, $d(x,q) > d(p_k,q)/(1+\epsilon)$.*

Assuming the existence of a data structure which solves ϵ-kANNs, we can weaken Definition 2 as follows.

▶ **Definition 4.** Let (Y, d_Y), (Z, d_Z) be metric spaces and $X \subseteq Y$. A distribution over mappings $f : Y \to Z$ is a *locality preserving embedding* with distortion $D \geq 1$, probability of correctness $P \in [0,1]$ and locality parameter k, if $\forall \epsilon \geq 0$ and $\forall q \in Y$, with probability at least P, when $S = \{f(p_1), \ldots, f(p_k)\}$ is a solution to ϵ-kANNs for q, under Assumption 3 then there exists $f(x) \in S$ such that x is a $(D \cdot (1+\epsilon))$-approximate nearest neighbor of q in X.

According to this definition we can reduce the problem of ϵ-ANN in dimension d to the problem of computing k approximate nearest neighbors in dimension $d' < d$.

We use the Johnson-Lindenstrauss dimensionality reduction technique and more specifically the proof obtained in [11]. As it was previously discussed, there also exist alternative proofs which correspond to alternative randomized mappings.

▶ **Lemma 5.** [11] *There exists a distribution over linear maps $A : \mathbb{R}^d \to \mathbb{R}^{d'}$ s.t., for any $p \in \mathbb{R}^d$ with $\|p\| = 1$:*

- *if $\beta^2 < 1$ then* $\Pr[\|Ap\|^2 \leq \beta^2 \cdot \frac{d'}{d}] \leq exp(\frac{d'}{2}(1 - \beta^2 + 2\ln\beta)$,
- *if $\beta^2 > 1$ then* $\Pr[\|Ap\|^2 \geq \beta^2 \cdot \frac{d'}{d}] \leq exp(\frac{d'}{2}(1 - \beta^2 + 2\ln\beta)$.

We prove the following lemma which will be useful.

▶ **Lemma 6.** *For all $i \in \mathbb{N}$, $\epsilon \in (0,1)$, the following holds:*

$$\frac{(1+\epsilon/2)^2}{(2^i(1+\epsilon))^2} - 2\ln\frac{(1+\epsilon/2)}{2^i(1+\epsilon)} - 1 > 0.05(i+1)\epsilon^2.$$

Proof. For $i = 0$, it can be checked that if $\epsilon \in (0,1)$ then, $\frac{(1+\epsilon/2)^2}{(1+\epsilon)^2} - 2\ln\frac{1+\epsilon/2}{1+\epsilon} - 1 > 0.05\epsilon^2$. This is our induction basis. Let $j \geq 0$ be such that the induction hypothesis holds. Then, to prove the induction step

$$\frac{1}{4}\frac{(1+\epsilon/2)^2}{(2^j(1+\epsilon))^2} - 2\ln\frac{(1+\epsilon/2)}{2^j(1+\epsilon)} + 2\ln 2 - 1 > 0.05(j+1)\epsilon^2 - \frac{3}{4}\frac{(1+\epsilon/2)^2}{(2^j(1+\epsilon))^2} + 2\ln 2 >$$

$$> 0.05(j+1)\epsilon^2 - \frac{3}{4} + 2\ln 2 > 0.05(j+2)\epsilon^2,$$

since $\epsilon \in (0,1)$. ◀

A simple calculation shows the following.

▶ **Lemma 7.** *For all $x > 0$, it holds:*

$$\frac{(1+x)^2}{(1+2x)^2} - 2\ln(\frac{1+x}{1+2x}) - 1 < (1+x)^2 - 2\ln(1+x) - 1. \tag{1}$$

▶ **Theorem 8.** *Under the notation of Definition 4, there exists a randomized mapping $f : \mathbb{R}^d \to \mathbb{R}^{d'}$ which satisfies Definition 4 for $d' = O(\log \frac{n}{\delta k}/\epsilon^2)$, $\epsilon > 0$, distortion $D = 1 + \epsilon$ and probability of success $1 - \delta$, for any constant $\delta \in (0, 1)$.*

Proof. Let X be a set of n points in $\mathbb{R}^d$ and consider map

$$f : \mathbb{R}^d \to \mathbb{R}^{d'} : v \mapsto \sqrt{d/d'} \cdot A\ v,$$

where A is a matrix chosen from a distribution as in Lemma 5. Wlog the query point q lies at the origin and its nearest neighbor u lies at distance 1 from q. We denote by $c \geq 1$ the approximation ratio guaranteed by the assumed data structure. That is, the assumed data structure solves the $(c-1)$-kANNs problem. For each point x, $L_x = \|Ax\|^2/\|x\|^2$. Let N be the random variable whose value indicates the number of "bad" candidates, that is

$$N = |\{x \in X\ :\ \|x - q\| > \gamma \wedge L_x \leq \frac{\beta^2}{\gamma^2} \cdot \frac{d'}{d}\}|,$$

where we define $\beta = c(1 + \epsilon/2)$, $\gamma = c(1 + \epsilon)$. Hence, by Lemma 5,

$$\mathbb{E}[N] \leq n \cdot exp(\frac{d'}{2}(1 - \frac{\beta^2}{\gamma^2} + 2\ln\frac{\beta}{\gamma})).$$

By Markov's inequality,

$$\Pr[N \geq k] \leq \frac{\mathbb{E}[N]}{k} \implies \Pr[N \geq k] \leq n \cdot exp(\frac{d'}{2}(1 - \frac{\beta^2}{\gamma^2} + 2\ln\frac{\beta}{\gamma}))/k.$$

The event of failure is defined as the disjunction of two events:

$$[\,N \geq k\,] \vee [\,L_u \geq (\beta/c)^2\frac{d'}{d}\,], \tag{2}$$

and its probability is at most equal to

$$\Pr[N \geq k] + exp(\frac{d'}{2}(1 - (\beta/c)^2 + 2\ln(\beta/c))),$$

by applying again Lemma 5. Now, we bound these two terms. For the first one, we choose d' such that

$$d' \geq 2\,\frac{\ln\frac{2n}{\delta k}}{\frac{\beta^2}{\gamma^2} - 1 - 2\ln\frac{\beta}{\gamma}}. \tag{3}$$

Therefore,

$$\frac{exp(\frac{d'}{2}(1 - \frac{\beta^2}{\gamma^2} + 2\ln\frac{\beta}{\gamma}))}{k} \leq \frac{\delta}{2n} \implies \Pr[N \geq k] \leq \frac{\delta}{2}. \tag{4}$$

Notice that $k \leq n$ and due to expression (1), we obtain $(\beta/\gamma)^2 - 2\ln(\beta/\gamma) - 1 < (\beta/c)^2 - 2\ln(\beta/c) - 1$. Hence, inequality (3) implies the following inequality:

$$d' \geq 2\frac{\ln\frac{2}{\delta}}{(\beta/c)^2 - 1 - 2\ln(\beta/c)}.$$

Therefore, the second term in expression (2) is bounded as follows:

$$exp(\frac{d'}{2}(1-(\frac{\beta}{c})^2+2\ln\frac{\beta}{c}))\leq\frac{\delta}{2}. \tag{5}$$

Inequalities (4), (5) imply that the total probability of failure in expression (2) is at most δ.

Using Lemma 6 for $i=0$, we obtain, that there exists d' such that

$$d'=O(\log\frac{n}{\delta k}/\epsilon^2)$$

and with probability at least $1-\delta$, these two events occur:

- $\|f(q)-f(u)\|\leq(1+\frac{\epsilon}{2})\|u-q\|$.
- $|\{p\in X|\|p-q\|>c(1+\epsilon)\|u-q\| \implies \|f(q)-f(p)\|\leq c(1+\epsilon/2)\|u-q\|\}|<k$.

Now consider the case when the random experiment succeeds and let $S=\{f(p_1),...,f(p_k)\}$ a solution of the $(c-1)$-kANNs problem in the projected space, given by a data-structure which satisfies Assumption 3. We have that $\forall f(x)\in f(X)\setminus S'$, $\|f(x)-f(q)\|>\|f(p_k)-f(q)\|/c$ where S' is the set of all points visited by the search routine.

Now, if $f(u)\in S$ then S contains the projection of the nearest neighbor. If $f(u)\notin S$ then if $f(u)\notin S'$ we have the following:

$$\|f(u)-f(q)\|>\|f(p_k)-f(q)\|/c \implies \|f(p_k)-f(q)\|<c(1+\epsilon/2)\|u-q\|,$$

which means that there exists at least one point $f(p^*)\in S$ s.t. $\|q-p^*\|\leq c(1+\epsilon)\|u-q\|$. Finally, if $f(u)\notin S$ but $f(u)\in S'$ then

$$\|f(p_k)-f(q)\|\leq\|f(u)-f(q)\| \implies \|f(p_k)-f(q)\|\leq(1+\epsilon/2)\|u-q\|,$$

which means that there exists at least one point $f(p^*)\in S$ s.t. $\|q-p^*\|\leq c(1+\epsilon)\|u-q\|$.

Hence, f satisfies Definition 4 for $D=1+\epsilon$. ◀

4 Approximate Nearest Neighbor Search

This section combines tree-based data structures which solve ϵ-kANNs with the results above, in order to obtain an efficient randomized data structure which solves ϵ-ANN.

BBD-trees [8] require $O(dn)$ space, and allow computing k points, which are $(1+\epsilon)$-approximate nearest neighbors, within time $O((\lceil 1+6\frac{d}{\epsilon}\rceil^d+k)d\log n)$. The preprocessing time is $O(dn\log n)$. Notice, that BBD-trees satisfy the Assumption 3. The algorithm for the ϵ-kANNs search, visits cells in increasing order with respect to their distance from the query point q. If the current cell lies at distance more than r_k/c where r_k is the current distance to the kth nearest neighbor, the search terminates. We apply the random projection for distortion $D=1+\epsilon$, thus relating approximation error to the allowed distortion; this is not required but simplifies the analysis.

Moreover, $k=n^\rho$; the formula for $\rho<1$ is determined below. Our analysis then focuses on the asymptotic behaviour of the term $O(\lceil 1+6\frac{d'}{\epsilon}\rceil^{d'}+k)$.

▶ **Lemma 9.** *With the above notation, there exists $k>0$ s.t., for fixed $\epsilon\in(0,1)$, it holds that $\lceil 1+6\frac{d'}{\epsilon}\rceil^{d'}+k=O(n^\rho)$, where $\rho\leq 1-\epsilon^2/\hat{c}(\epsilon^2+\log(\max\{\frac{1}{\epsilon},\log n\}))<1$ for some appropriate constant $\hat{c}>1$.*

Proof. Recall that $d'\leq\frac{\tilde{c}}{\epsilon^2}\ln\frac{n}{k}$ for some appropriate constant $\tilde{c}>0$. The constant δ is hidden in $\tilde{c}$. Since $(\frac{d'}{\epsilon})^{d'}$ is a decreasing function of k, we need to choose k s.t. $(\frac{d'}{\epsilon})^{d'}=\Theta(k)$.

Let $k = n^\rho$. Obviously $\lceil 1+6\frac{d'}{\epsilon}\rceil^{d'} \leq (c'\frac{d'}{\epsilon})^{d'}$, for some appropriate constant $c' \in (1,7)$. Then, by substituting d', k we have:

$$(c'\frac{d'}{\epsilon})^{d'} = n^{\frac{\tilde{c}(1-\rho)}{\epsilon^2}\ln(\frac{\tilde{c}c'(1-\rho)\ln n}{\epsilon^3})}. \tag{6}$$

We assume $\epsilon \in (0,1)$ is a fixed constant. Hence, it is reasonable to assume that $\frac{1}{\epsilon} < n$. We consider two cases when comparing $\ln n$ to ϵ:

- $\frac{1}{\epsilon} \leq \ln n$. Substituting $\rho = 1 - \frac{\epsilon^2}{2\tilde{c}(\epsilon^2+\ln(c'\ln n))}$ into equation (6), the exponent of n is bounded as follows:

$$\frac{\tilde{c}(1-\rho)}{\epsilon^2}\ln(\frac{\tilde{c}c'(1-\rho)\ln n}{\epsilon^3}) =$$
$$= \frac{\tilde{c}}{2\tilde{c}(\epsilon^2+\ln(c'\ln n))}\cdot[\ln(c'\ln n) + \ln\frac{1}{\epsilon} - \ln(2\epsilon^2 + 2\ln(c'\ln n))] < \rho.$$

- $\frac{1}{\epsilon} > \ln n$. Substituting $\rho = 1 - \frac{\epsilon^2}{2\tilde{c}(\epsilon^2+\ln\frac{c'}{\epsilon})}$ into equation (6), the exponent of n is bounded as follows:

$$\frac{\tilde{c}(1-\rho)}{\epsilon^2}\ln(\frac{\tilde{c}c'(1-\rho)\ln n}{\epsilon^3}) =$$
$$= \frac{\tilde{c}}{2\tilde{c}(\epsilon^2+\ln\frac{c'}{\epsilon})}\cdot[\ln\ln n + \ln\frac{c'}{\epsilon} - \ln(2\epsilon^2 + 2\ln\frac{c'}{\epsilon})] < \rho.$$

◀

Notice that for both cases $d' = O(\frac{\log n}{\epsilon^2+\log\log n})$.

Combining Theorem 8 with Lemma 9 yields the following main theorem.

▶ **Theorem 10** (Main). *Given n points in $\mathbb{R}^d$, there exists a randomized data structure which requires $O(dn)$ space and reports an $(1+\epsilon)^2$-approximate nearest neighbor in time*

$$O(dn^\rho \log n), \text{ where } \rho \leq 1 - \epsilon^2/\hat{c}(\epsilon^2 + \log(\max\{\frac{1}{\epsilon}, \log n\}))$$

for some appropriate constant $\hat{c} > 1$. The preprocessing time is $O(dn\log n)$. For each query $q \in \mathbb{R}^d$, the preprocessing phase succeeds with any constant probability.

Proof. The space required to store the dataset is $O(dn)$. The space used by BBD-trees is $O(d'n)$ where d' is defined in Lemma 9. We also need $O(dd')$ space for the matrix A as specified in Theorem 8. Hence, since $d' < d$ and $d' < n$, the total space usage is bounded above by $O(dn)$.

The preprocessing consists of building the BBD-tree which costs $O(d'n\log n)$ time and sampling A. Notice that we can sample a d'-dimensional random subspace in time $O(dd'^2)$ as follows. First, we sample in time $O(dd')$, a $d \times d'$ matrix where its elements are independent random variables with the standard normal distribution $N(0,1)$. Then, we orthonormalize using Gram-Schmidt in time $O(dd'^2)$. Since $d' = O(\log n)$, the total preprocessing time is bounded by $O(dn\log n)$.

For each query we use A to project the point in time $O(dd')$. Next, we compute its n^ρ approximate nearest neighbors in time $O(d'n^\rho\log n)$ and we check its neighbors with their real coordinates in time $O(dn^\rho)$. Hence, each query costs $O(d\log n + d'n^\rho\log n + dn^\rho) = O(dn^\rho\log n)$ because $d' = O(\log n)$, $d' = O(d)$. Thus, the query time is dominated by the time required for ϵ-kANNs search and the time to check the returned sequence of k approximate nearest neighbors. ◀

To be more precise, the probability of success, which is the probability that the random projection succeeds according to Theorem. 8, is greater than $1-\delta$, for any constant $\delta \in (0,1)$. Notice that the preprocessing time for BBD-trees has no dependence on ϵ.

5 Bounded Expansion Rate

This section models the structure that the data points may have so as to obtain more precise bounds.

The bound on the dimension obtained in Theorem 8 is quite pessimistic. We expect that, in practice, the space dimension needed in order to have a sufficiently good projection is less than what Theorem 8 guarantees. Intuitively, we do not expect to have instances where all points in X, which are not approximate nearest neighbors of q, lie at distance almost equal to $(1+\epsilon)d(q,X)$. To this end, we consider the case of pointsets with bounded expansion rate.

▶ **Definition 11.** Let M a metric space and $X \subseteq M$ a finite pointset and let $B_p(r) \subseteq X$ denote the points of X lying in the closed ball centered at p with radius r. We say that X has (ρ, c)-expansion rate if and only if, $\forall p \in M$ and $r > 0$,

$$|B_p(r)| \geq \rho \implies |B_p(2r)| \leq c \cdot |B_p(r)|.$$

▶ **Theorem 12.** *Under the notation introduced in the previous definitions, there exists a randomized mapping $f : \mathbb{R}^d \to \mathbb{R}^{d'}$ which satisfies Definition 4 for dimension $d' = O(\frac{\log(c+\frac{\rho}{\delta k})}{\epsilon^2})$, distortion $D = 1+\epsilon$ and probability of success $1-\delta$, for any constant $\delta \in (0,1)$, for pointsets with (ρ, c)-expansion rate.*

Proof. We proceed in the same spirit as in the proof of Theorem 8, and using the notation from that proof. Let r_0 be the distance to the $\rho-$th nearest neighbor, excluding neighbors at distance $\leq 1+\epsilon$. For $i > 0$, let $r_i = 2 \cdot r_{i-1}$ and set $r_{-1} = 1+\epsilon$. Clearly,

$$\begin{aligned}\mathbb{E}[N] &\leq \sum_{i=0}^{\infty} |B_p(r_i)| \cdot exp(\frac{d'}{2}(1 - \frac{(1+\epsilon/2)^2}{r_{i-1}^2} + 2\ln\frac{1+\epsilon/2}{r_{i-1}})) \\ &\leq \sum_{i=0}^{\infty} c^i\rho \cdot exp(\frac{d'}{2}(1 - \frac{(1+\epsilon/2)^2}{2^{2i}(1+\epsilon)^2} + 2\ln\frac{1+\epsilon/2}{2^i(1+\epsilon)})).\end{aligned}$$

Now, using Lemma 6,

$$\mathbb{E}[N] \leq \sum_{i=0}^{\infty} c^i\rho \cdot exp(-\frac{d'}{2}0.05(i+1)\epsilon^2),$$

and for $d' \geq 40 \cdot \ln(c + \frac{2\rho}{k\delta})/\epsilon^2$,

$$\mathbb{E}[N] \leq \rho \cdot \sum_{i=0}^{\infty} c^i \cdot (\frac{1}{c+\frac{2\rho}{k\delta}})^{i+1} = \rho \cdot \sum_{i=0}^{\infty} c^i \cdot (\frac{1}{c})^{i+1} \cdot (\frac{1}{1+\frac{2\rho}{kc\delta}})^{i+1} = \frac{\rho}{c} \cdot \sum_{i=0}^{\infty} (\frac{1}{1+\frac{2\rho}{kc\delta}})^{i+1} = \frac{k\delta}{2}.$$

Finally,

$$\Pr[N \geq k] \leq \frac{\mathbb{E}[N]}{k} \leq \frac{\delta}{2}.$$

◀

Employing Theorem 12 we obtain a result analogous to Theorem 10 which is weaker than those in [20, 9] but underlines the fact that our scheme shall be sensitive to structure in the input data, for real world assumptions.

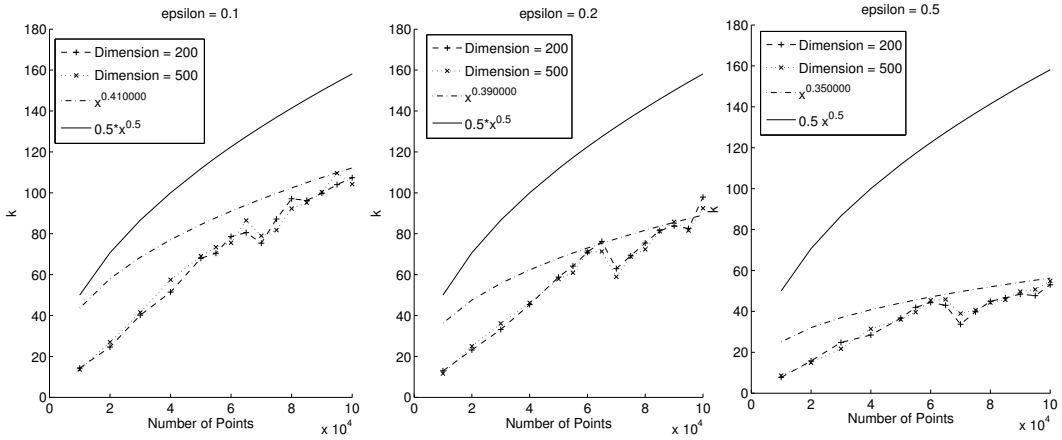

Figure 1 Plot of k as n increases for the "planted nearest neighbor model" datasets. The highest line corresponds to $\frac{\sqrt{n}}{2}$ and the dotted line to a function of the form n^{ρ}, where $\rho = 0.41, 0.39, 0.35$ that best fits the data.

▶ **Theorem 13.** *Given n points in $\mathbb{R}^d$ with $(\log n, c)$-expansion rate, there exists a randomized data structure which requires $O(dn)$ space and reports an $(1+\epsilon)^2$-approximate nearest neighbor in time $O((C^{1/\epsilon^3} + \log n)d \log n/\epsilon^2)$, for some constant C depending on c. The preprocessing time is $O(dn \log n)$. For each query $q \in \mathbb{R}^d$, the preprocessing phase succeeds with any constant probability.*

Proof. Set $k = \log n$. Then $d' = O(\frac{\log c}{\epsilon^2})$ and $(\frac{d'}{\epsilon})^{d'} = O(c^{\frac{1}{\epsilon^2}\log[\frac{\log c}{\epsilon^3}]})$. Now the query time is

$$O((c^{\frac{1}{\epsilon^2}\log[\frac{\log c}{\epsilon^3}]} + \log n)d\frac{\log c}{\epsilon^2}\log n) = O((C^{1/\epsilon^3} + \log n)d\frac{\log n}{\epsilon^2}),$$

for some constant C such that $c^{\log(\log c/\epsilon^3)/\epsilon^2} = O(C^{1/\epsilon^3})$. ◀

6 Experiments

In the following two sections we present and discuss the two experiments we performed. In the first one we computed the average value of k in a worst-case dataset and we validated that it is indeed sublinear. In the second one we made an ANN query time and memory usage comparison against a LSH implementation using both artificial and real life datasets.

6.1 Validation of k

In this section we present an experimental verification of our approach. We show that the number k of the nearest neighbors in the random projection space that we need to examine in order to find an approximate nearest neighbor in the original space depends sublinearly on n. Recall that we denote by $\|\cdot\|$ the Euclidean norm.

Dataset

We generated our own synthetic datasets and query points to verify our results. We decided to follow two different procedures for data generation in order to be as complete as possible. First of all, as in [12], we followed the "planted nearest neighbor model" for our datasets. This model guarantees for each query point q the existence of a few approximate nearest

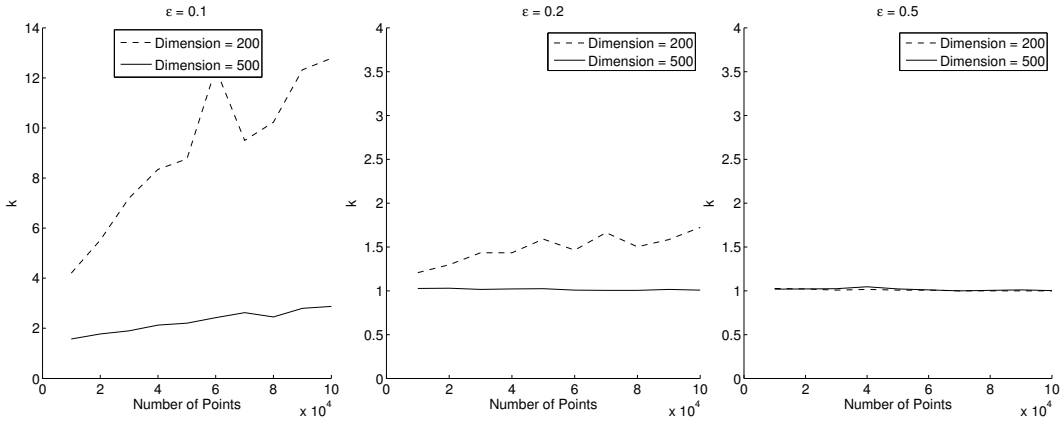

Figure 2 Plot of k as n increases for the gaussian datasets. We see how increasing the number of approximate nearest neighbors in this case decreases the value of k.

neighbors while keeping all others points sufficiently far from q. The benefit of this approach is that it represents a typical ANN search scenario, where for each point there exist only a handful approximate nearest neighbors. In contrast, in a uniformly generated dataset, all the points will tend to be equidistant to each other in high dimensions, which is quite unrealistic.

In order to generate such a dataset, first we create a set Q of query points chosen uniformly at random in $\mathbb{R}^d$. Then, for each point $q \in Q$, we generate a single point p at distance R from q, which will be its single (approximate) nearest neighbor. Then, we create more points at distance $\geq (1+\epsilon)R$ from q, while making sure that they shall not be closer than $(1+\epsilon)R$ to any other query point q'. This dataset now has the property that every query point has exactly one approximate nearest neighbor, while all other points are at distance $\geq (1+\epsilon)R$.

We fix $R = 2$, let $\epsilon \in \{0.1, 0.2, 0.5\}, d = \{200, 500\}$ and the total number of points $n \in \{10^4, 2\times 10^4, \ldots, 5\times 10^4, 5.5\times 10^4, 6\times 10^4, 6.5\times 10^4, \ldots, 10^5\}$. For each combination of the above we created a dataset X from a set Q of 100 query points where each query coordinate was chosen uniformly at random in the range $[-20, 20]$.

The second type of datasets consisted again of sets of 100 query points in $\mathbb{R}^d$ where each coordinate was chosen uniformly at random in the range $[-20, 20]$. Each query point was paired with a random variable σ_q^2 uniformly distributed in $[15, 25]$ and together they specified a gaussian distribution in $\mathbb{R}^d$ of mean value $\mu = q$ and variance σ_q^2 per coordinate. For each distribution we drew n points in the same set as was previously specified.

Scenario

We performed the following experiment for the "planted nearest neighbor model". In each dataset X, we consider, for every query point q, its unique (approximate) nearest neighbor $p \in X$. Then we use a random mapping f from $\mathbb{R}^d$ to a Euclidean space of lower dimension $d' = \frac{\log n}{\log\log n}$ using a gaussian matrix G, where each entry $G_{ij} \sim N(0,1)$. This matrix guarantees a low distortion embedding [15]. Then, we perform a range query centered at $f(q)$ with radius $\|f(q) - f(p)\|$ in $f(X)$: we denote by $rank_q(p)$ the number of points found. Then, exactly $rank_q(p)$ points are needed to be selected in the worst case as k-nearest neighbors of $f(q)$ in order for the approximate nearest neighbor $f(p)$ to be among them, so $k = rank_q(p)$.

For the datasets with the gaussian distributions we compute again the maximum number of points k needed to visit in the lower-dimensional space in order to find an ϵ-approximate nearest neighbor of each query point q in the original space. In this case the experiment

works as follows: we find all the ϵ-approximate nearest neighbors of a query point q. Let S_q be the set containing for each query q its ϵ-kANNs. Next, let $p_q = \arg\min_{p \in S} \|f(p) - f(q)\|$. Now as before we perform a range query centered at $f(q)$ with radius $\|f(q) - f(p_q)\|$. We consider as k the number of points returned by this query.

Results

The "planted nearest neighbor model" datasets constitute a worst-case input for our approach since every query point has only one approximate nearest neighbor and has many points lying near the boundary of $(1+\epsilon)$. We expect that the number of k approximate nearest neighbors needed to consider in this case will be higher than in the case of the gaussian distributions, but still expect the number to be considerably sublinear.

In Figure 1 we present the average value of k as we increase the number of points n for the planted nearest neighbor model. We can see that k is indeed significantly smaller than n. The line corresponding to the averages may not be smooth, which is unavoidable due to the random nature of the embedding, but it does have an intrinsic concavity, which shows that the dependency of k on n is sublinear. For comparison we also display the function $\sqrt{n}/2$, as well as a function of the form $n^\rho, \rho < 1$ which was computed by SAGE that best fits the data per plot. The fitting was performed on the points in the range $[50000, 100000]$ as to better capture the asymptotic behaviour. In Figure 2 we show again the average value of k as we increase the number of points n for the gaussian distribution datasets. As expected we see that the expected value of k is much smaller than n and also smaller than the expected value of k in the worst-case scenario, which is the planted nearest neighbor model.

6.2 ANN experiments

In this section we present a naive comparison between our algorithm and the E2LSH [3] implementation of the LSH framework for approximate nearest neighbor queries.

Experiment Description

We projected all the "planted nearest neighbor" datasets, down to $\frac{\log n}{\log \log n}$ dimensions. We remind the reader that these datasets were created to have a single approximate nearest neighbor for each query at distance R and all other points at distance $> (1+\epsilon)R$. We then built a BBD-tree data structure on the projected space using the `ANN` library [22] with the default settings. Next, we measured the average time needed for each query q to find its ϵ-kANNs, for $k = \sqrt{n}$, using the BBD-Tree data structure and then to select the first point at distance $\leq R$ out of the k in the original space. We compare these times to the average times reported by E2LSH range queries for $R = 2$, when used from its default script for probability of success 0.95. The script first performs an estimation of the best parameters for the dataset and then builds its data structure using these parameters. We required from the two approaches to have accuracy > 0.90, which in our case means that in at least 90 out of the 100 queries they would manage to find the approximate nearest neighbor. We also measured the maximum resident set size of each approach which translates to the maximum portion of the main memory (RAM) occupied by a process during its lifetime. This roughly corresponds to the size of the dataset plus the size of the data structure for the E2LSH implementation and to hte size of the dataset plus the size of the embedded dataset plus the size of the data structure for our approach.

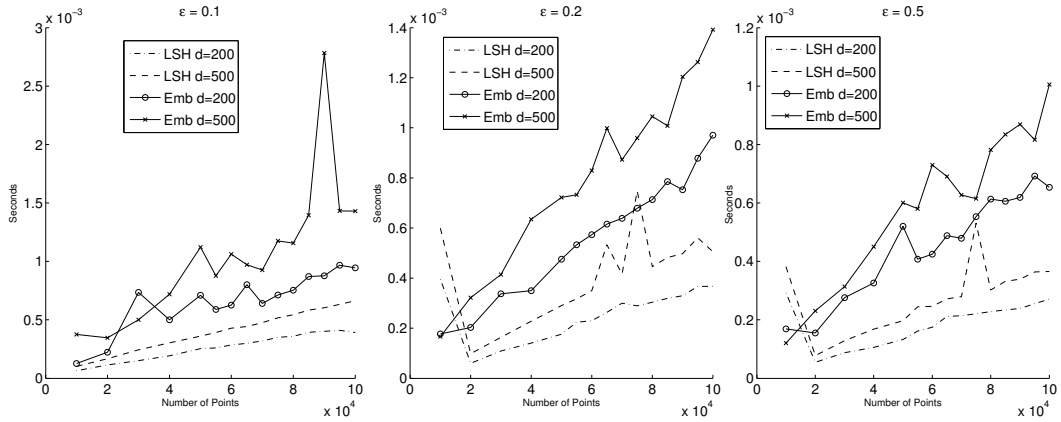

Figure 3 Comparison of average query time of our embedding approach against the E2LSH implementation.

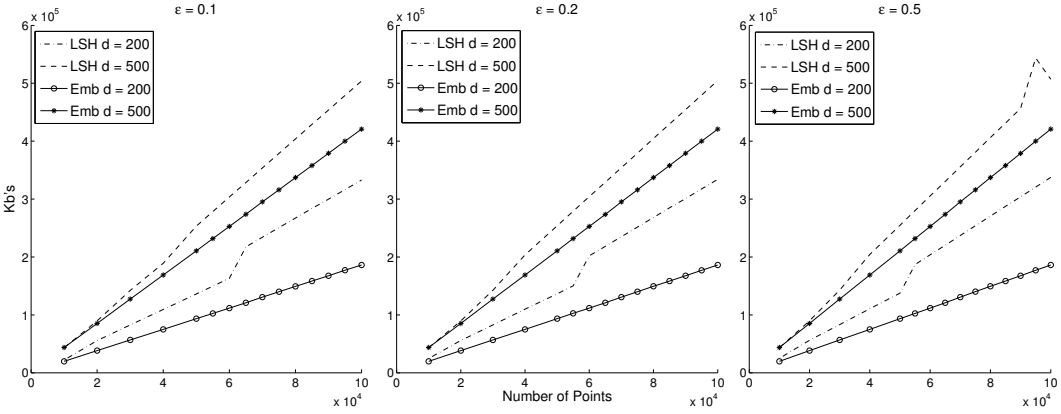

Figure 4 Comparison of memory usage of our embedding approach against the E2LSH implementation.

ANN Results

It is clear from Figure 3 that E2LSH is faster than our approach by a factor of 3. However in Figure 4, where we present the memory usage comparison between the two approaches, it is obvious that E2LSH also requires more space. Figure 4 also validates the linear space dependency of our embedding method. A few points can be raised here. First of all, we supplied the appropriate range to the LSH implementation, which gave it an advantage, because typically that would have to be computed empirically. To counter that, we allowed our algorithm to stop its search in the original space when it encountered a point that was at distance $\leq R$ from the query point. Our approach was simpler and the bottleneck was in the computation of the closest point out of the k returned from the BBD-Tree. We conjecture that we can choose better values for our parameters d' and k. Lastly, the theoretical guarantees for the query time of LSH are better than ours, but we did perform better in terms of space usage as expected.

Real life dataset

We also compared the two approaches using the ANN_SIFT1M [17] dataset which contains a collection of 1000000 vectors in 128 dimensions. This dataset also provides a query file containing 10000 vectors and a groundtruth file, which contains for each query the IDs of its 100 nearest neighbors. These files allowed us to estimate the accuracy for each approach, as the fraction $\frac{\#hits}{10000}$ where $\#hits$ denotes, for some query, the number of times one of its 100 nearest neighbors were returned. The parameters of the two implementations were chosen empirically in order to achieve an accuracy of about 85%. For our approach we set the projection dimension $d' = 25$ and for the BBD-trees we specified 100 points per leaf and $\epsilon = 0.5$ for the ϵ-kANNs queries. We also used $k = \sqrt{n}$. For the E2LSH implementation we specified the radius $R = 240$, $k = 18$ and $L = 250$. As before we measured the average query time and the maximum resident set size. Our approach required an average of 0.171588s per query, whilst E2LSH required 0.051957s. However our memory footprint was 1255948 kbytes and E2LSH used 4781400 kbytes.

7 Open questions

In terms of practical efficiency it is obvious that checking the real distance to the neighbors while performing an ϵ-kANNs search in the reduced space, is more efficient in practice than naively scanning the returned sequence of k-approximate nearest neighbors and looking for the best in the initial space. Moreover, we do not exploit the fact that BBD-trees return a sequence and not simply a set of neighbors.

Our embedding possibly has further applications. One possible application is the problem of computing the k-th approximate nearest neighbor. The problem may reduce to computing all neighbors between the i-th and the j-th nearest neighbors in a space of significantly smaller dimension for some appropriate $i < k < j$. Other possible applications include computing the approximate minimum spanning tree or the closest pair of points.

Our embedding approach could be possibly applied in other metrics or exploit other properties of the pointset. We also intend to seek connections between our work and the notion of local embeddings introduced in [1].

References

1 I. Abraham, Y. Bartal, and O. Neiman. Local embeddings of metric spaces. In *Proc. 39th ACM Symposium on Theory of Computing*, pages 631–640. ACM Press, 2007.

2 D. Achlioptas. Database-friendly random projections: Johnson-Lindenstrauss with binary coins. *J. Comput. Syst. Sci.*, 66(4):671–687, 2003.

3 A. Andoni and P. Indyk. E^{2}LSH 0.1 User Manual, Implementation of LSH: E2LSH, `http://www.mit.edu/~andoni/LSH`, 2005.

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

5 A. Andoni and I. Razenshteyn. Optimal data-dependent hashing for approximate near neighbors. In *arXiv:1501.01062, to appear in the Proc. 47th ACM Symp. Theory of Computing*, STOC'15, 2015.

6 S. Arya, G. D. da Fonseca, and D. M. Mount. Approximate polytope membership queries. In *Proc. 43rd Annual ACM Symp. Theory of Computing*, STOC'11, pages 579–586, 2011.

7 S. Arya, T. Malamatos, and D. M. Mount. Space-time tradeoffs for approximate nearest neighbor searching. *J. ACM*, 57(1):1:1–1:54, 2009.

8 S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Y. Wu. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *J. ACM*, 45(6):891–923, 1998.

9 A. Beygelzimer, S. Kakade, and J. Langford. Cover trees for nearest neighbor. In *Proc. 23rd Intern. Conf. Machine Learning*, ICML'06, pages 97–104, 2006.

10 S. Dasgupta and Y. Freund. Random projection trees and low dimensional manifolds. In *Proc. 40th Annual ACM Symp. Theory of Computing*, STOC'08, pages 537–546, 2008.

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

12 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.

13 A. Gupta, R. Krauthgamer, and J. R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *Proc. 44th Annual IEEE Symp. Foundations of Computer Science*, FOCS'03, pages 534–, 2003.

14 S. Har-Peled and M. 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.

15 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.

16 P. Indyk and A. Naor. Nearest-neighbor-preserving embeddings. *ACM Trans. Algorithms*, 3(3), 2007.

17 H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 33(1):117–128, 2011.

18 W. B. Johnson and J. Lindenstrauss. Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

19 D. R. Karger and M. Ruhl. Finding nearest neighbors in growth-restricted metrics. In *Proc. 34th Annual ACM Symp. Theory of Computing*, STOC'02, pages 741–750, 2002.

20 R. Krauthgamer and J. R. Lee. Navigating nets: Simple algorithms for proximity search. In *Proc. 15th Annual ACM-SIAM Symp. Discrete Algorithms*, SODA'04, pages 798–807, 2004.

21 S. Meiser. Point location in arrangements of hyperplanes. *Inf. Comput.*, 106(2):286–303, 1993.

22 D. M. Mount. ANN programming manual: `http://www.cs.umd.edu/~mount/ANN/`, 2010.

23 R. O'Donnell, Yi 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, 2014.

24 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.

25 I. Psarros. Low quality embeddings and approximate nearest neighbors, MSc Thesis, Dept. of Informatics & Telecommunications, University of Athens, 2014.

26 S. Vempala. Randomly-oriented k-d trees adapt to intrinsic dimension. In *Proc. Foundations of Software Technology & Theor. Computer Science*, pages 48–57, 2012.