Approximate Nearest Neighbors and the Fast Johnson-Lindenstrauss Transform

Nir Ailon*
Department of Computer Science
Princeton University
nailon@cs.princeton.edu

Bernard Chazelle†
Department of Computer Science
Princeton University
chazelle@cs.princeton.edu

ABSTRACT

We introduce a new low-distortion embedding of ℓ_2^d into $\ell_p^{O(\log n)}$ $(p = 1, 2)$, called the *Fast-Johnson-Lindenstrauss-Transform*. The FJLT is faster than standard random projections and just as easy to implement. It is based upon the preconditioning of a sparse projection matrix with a randomized Fourier transform. Sparse random projections are unsuitable for low-distortion embeddings. We overcome this handicap by exploiting the "Heisenberg principle" of the Fourier transform, ie, its local-global duality. The FJLT can be used to speed up search algorithms based on low-distortion embeddings in ℓ_1 and ℓ_2. We consider the case of approximate nearest neighbors in ℓ_2^d. We provide a faster algorithm using classical projections, which we then further speed up by plugging in the FJLT. We also give a faster algorithm for searching over the hypercube.

Categories and Subject Descriptors

F.2.0 [**Analysis of Algorithms and Problem Complexity**]: General

General Terms

Algorithms

Keywords

Johnson-Lindenstrauss dimension reduction, Approximate nearest neighbor searching, Fourier transform, High-dimensional geometry

*This work was supported in part by a Charlotte Elizabeth Procter Honorific Fellowship from Princeton University.

†This work was supported in part by NSF grants CCR-998817, 0306283, ARO Grant DAAH04-96-1-0181.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
STOC'06, May21–23, 2006, Seattle, Washington, USA.
Copyright 2006 ACM 1-59593-134-1/06/0005 ...$5.00.

1. INTRODUCTION

By the Johnson-Lindenstrauss lemma [25,27,31], n points in Euclidean space can be projected down to $k = O(\varepsilon^{-2} \log n)$ dimensions while incurring a distortion of at most $1 + \varepsilon$ in their pairwise distances. To achieve this requires a dense k-by-d matrix; and so mapping each point takes $O(d \log n)$ time (for fixed ε). Is that optimal? A lower bound of Alon [3] dashes any hope of reducing the number of rows (as a function of n), so the obvious question is whether the matrix can be made sparse.

Achlioptas [1] has shown that the density can be reduced by a constant factor, but one cannot go much further because a sparse matrix will typically distort a sparse vector. To prevent this, we use a central concept from harmonic analysis known as the *Heisenberg principle* (so named because it is the key idea behind the Uncertainty Principle): A signal and its spectrum cannot be both concentrated. With this in mind, we precondition the random projection with a Fourier transform (via an FFT) in order to isometrically enlarge the support of any sparse vector. To prevent the inverse effect, ie, the sparsification of dense vectors, we randomize the Fourier transform.

The result is the *Fast-Johnson-Lindenstrauss-Transform*: a randomized FFT followed by a sparse projection. The FJLT shares the low-distortion characteristics of a random projection but with a lower complexity. It embeds ℓ_2 into ℓ_p, for $p = 1, 2$. The running time of the FJLT is $O(d \log d + \min\{d\varepsilon^{-2} \log n, \varepsilon^{p-4} \log^{p+1} n\})$, which outperforms the $O(d\varepsilon^{-2} \log n)$ complexity of its predecessors. Note that for ℓ_1, the running time has the simpler form of $O(d \log d + \varepsilon^{-3} \log^2 n)$.

The FJLT can be used to speed up search algorithms based on low-distortion random projections. We consider the case of approximate nearest neighbors in ℓ_2^d. Given a set P of n points, an *ε-ANN query* x asks for a point $p \in P$ no more than $1 + \varepsilon$ times away from x than its nearest neighbor. This problem has received considerable attention lately. There are two good reasons for this: (i) ANN boasts more applications than virtually any other geometric problem [24]; (ii) allowing a small error ε makes it possible to break the curse of dimensionality. We present two new, faster ANN algorithms. Note that both of them contain additional improvements, independent of FJLT.

- One works for ℓ_2^d. It stores n points in $\mathbb{R}^d$ and answers any ε-ANN query in $O(d \log d + \varepsilon^{-3} \log^2 n)$ time, while using $n^{O(\varepsilon^{-2})}$ storage. The solution is faster than its predecessors (at least for subexponential n) and con-

siderably simpler.

- The other works for the Hamming cube. It stores n points in the d-dimensional Hamming cube $\{0,1\}^d$ and answers any ε-ANN query in $O((d+\varepsilon^{-2}\log n)\log d)$ time, while using $d^2 n^{O(\varepsilon^{-2})}$ storage. This improves on the best previous query time of $O((d\log d)\varepsilon^{-2}\log n)$ [30].

1.1 Comparison with previous results

Following Johnson and Lindenstrauss's seminal paper [27], researchers suggested variants and simplifications of their design and/or proof. Frankl and Maehara [18] considered projections onto random orthogonal vectors. Independently, Dasgupta and Gupta [14] and Indyk and Motwani [26] greatly simplified the original proofs. Achlioptas [1] proved that, instead of Gaussian matrices, random ± 1-matrices can be used for JL. He also showed how to obtain a constant speedup by making roughly a 2/3 of the matrix null. His motivation was to make random projections easier to use in practice. Bingham and Mannila [7] also considered sparse projections heuristics for dimension-reduction based algorithms, and noticed that in practice they seem to give a considerable speedup with little compromise in quality. The FJLT also relies on matrix sparsity, but to make the scheme provably work, we precondition the projection with a randomized Fourier transform: this is, to our knowledge, a new idea in the context of JL embeddings.

There is an abundant literature on (approximate) nearest neighbor searching [4–6,8,11–13,16,21,23,24,26,29,30,32,34,35]. The early solutions typically suffered from the curse of dimensionality, but the last decade has witnessed a flurry of new algorithms that "break the curse" (see [24] for a recent survey). A few milestones deserve special mention in the context of this paper. Kleinberg [29] gave two algorithms for ANN in ℓ_2^d. The first one runs in $O((d\log^2 d)(d+\log n))$ time but requires storage exponential in d. The second one runs in $O(n+d\log^3 n)$ time, improving on the trivial $O(nd)$ algorithm, and requires only $O(dn \text{ polylog } n)$ space. Both algorithms are based on the key idea of *projection onto random lines*, which was used in subsequent results as well as in this paper.

The first algorithms with query times of $\text{poly}(d,\log n,\varepsilon^{-1})$ and polynomial storage (for fixed ε) were those of Indyk and Motwani [26] and Kushilevitz, Ostrovsky, and Rabani [30]. The first reference describes a reduction from ε-ANN to *approximate point location in equal balls* (ε-PLEB) for ℓ_2^d (as well as other metric spaces). The ε-PLEB problem is that of outputting a point $p \in P$ such that $\|x-p\|_2 \le r$ for some $r > 0$ if such a point exists, and null if all points p satisfy $\|x-p\|_2 > (1+\varepsilon)r$. Based on a new space decomposition (of independent interest), Indyk and Motwani showed how to reduce an ε-NN query to $O(\log^2 n)$ queries to an ε-PLEB oracle, a reduction later improved to $O(\log(n/\varepsilon))$ by Har-Peled [19]. The PLEB reduction can be thought of as a way of performing a binary search on the (unknown) distance to the nearest neighbor, while overcoming the possible unboundedness of the search space (we overcome this problem by using a different, simpler technique). One approach to PLEB is to use LSH (locality-sensitive hashing [26]), which requires $O(n^{1/(1+\varepsilon)})$ query time and near-quadratic (for small ε) storage. Another approach is to use dimension reduction techniques: using the methods of [19, 22, 26], this provides a query time of $O(\varepsilon^{-2}d\log n)$ with $n^{O(\varepsilon^{-2})}$ storage. We mention here that the dimension reduction overwhelms the running time of the algorithm: to remedy this was, in fact, the initial motivation for our work on the FJLT. Kushilevitz et al. [30]. described an ingenious (but intricate) reduction from ℓ_2^d to the Hamming cube, which results in $O(\varepsilon^{-2}d^2\text{polylog}\, n)$ query time and polynomial storage (again, for fixed ε). Our solution is faster than its predecessors (at least for $n = 2^{O(\varepsilon d)}$) and considerably simpler.

ANN searching over the Hamming cube does not suffer from the "unbounded binary search" problem. Kushilevitz et al. [30] gave a random-projection based algorithm with a query time of $O((d\log d)\varepsilon^{-2}\log n)$—an extra $\log\log d$ factor can be shaved off [10]. We improve the running time of their algorithm to $O((d+\varepsilon^{-2}\log n)\log d)$, which is the best to date (using polynomial storage). Again, we show how to optimize the dimension reduction step in their algorithm, but this time over GF (2).

2. THE FAST JOHNSON-LINDENSTRAUSS TRANSFORM

The transform (denoted FJLT) is a random distribution of linear mappings from $\mathbb{R}^d$ to $\mathbb{R}^k$, where the embedding dimension k is set to be $c\varepsilon^{-2}\log n$, for some large enough constant $c = c(p)$ Recall that $p \in \{1,2\}$ refers to the type of embedding we seek: $\ell_2^d \mapsto \ell_p^k$.

We may assume w.l.o.g. that $d = 2^m > k$. We will also assume that $n \ge d$ and $d = \Omega(\varepsilon^{-1/2})$ (otherwise the dimension of the reduced space is linear in the original dimension).

A random embedding $\Phi \sim FJLT(n,d,\varepsilon,p)$ can be obtained as a product of three real-valued matrices: $\Phi = PHD$. The matrices P and D are random and H is deterministic:

- $\boldsymbol{P}$ = k-by-d matrix whose elements are independent mixtures of 0 with an unbiased normal distribution of variance q^{-1}, where
$$q = \min\left\{\Theta\left(\frac{\varepsilon^{p-2}\log^p n}{d}\right), 1\right\}.$$
More precisely, $P_{ij} \sim N(0, q^{-1})$ with probability q, and $P_{ij} = 0$ with probability $1-q$.
- $\boldsymbol{H}$ = d-by-d normalized Hadamard matrix:
$$H_{ij} = d^{-1/2}(-1)^{\langle i-1, j-1\rangle},$$
where $\langle i,j\rangle$ is the dot-product of the m-bit vectors i,j expressed in binary.
- $\boldsymbol{D}$ = d-by-d diagonal matrix, where each D_{ii} is drawn independently from $\{-1,1\}$ with probability 1/2.

The Hadamard matrix encodes the discrete Fourier transform over the additive group $(\mathbb{Z}/2\mathbb{Z})^d$: its FFT is particularly simple and requires $O(d\log d)$ time. It follows that, with high probability (which we make precise in Lemma 2.1), the mapping Φx of *any* vector $x \in \mathbb{R}^d$ can be computed in time $O(d\log d + qd\varepsilon^{-2}\log n)$ (all running times are expected over the random bits of the algorithm). We now make our statement on the FJLT precise.

LEMMA 2.1. **[The FJLT Lemma]** *Fix any set X of n vectors in $\mathbb{R}^d$, $\varepsilon < 1$, and $p \in \{1,2\}$. Let $\Phi \sim FJLT$. With probability at least 2/3, the following two events occur:*

1. *For all $x \in X$,*
$$(1-\varepsilon)\alpha_p\|x\|_2 \le \|\Phi x\|_p \le (1+\varepsilon)\alpha_p\|x\|_2 \ ,$$
where $\alpha_1 = k\sqrt{2\pi^{-1}}$ and $\alpha_2 = k$.
2. *The mapping $\Phi : \mathbb{R}^d \to \mathbb{R}^k$ requires*
$$O\left(d\log d + \min\{d\varepsilon^{-2}\log n, \varepsilon^{p-4}\log^{p+1} n\}\right)$$
operations.

Note that by repeating the construction $O(\log(1/\delta))$ times we can amplify the success probability to $1-\delta$ for any $\delta > 0$. If we know X, it is possible to test for success. In the ANN example presented in the following sections, however, it is not possible to check if the first part of the FJLT guarantee succeeds, because the vectors of X are not known during construction (only the size $|X|$ is known). However, one can amplify the probability of success by repeating the construction $O(\log(1/\delta))$ times, running the nearest neighbor algorithm w.r.t. each construction and taking the nearest among the outputs.

PROOF. Without loss of generality, we can assume that $\varepsilon < \varepsilon_0$ for some suitably small ε_0.

Fix some $x \in X$, and define the random variable $u = HDx = (u_1, \ldots, u_d)^T$. Assume w.l.o.g. that $\|x\|_2 = 1$. Note that u_1 is of the form $\sum_{i=1}^d a_i x_i$, where each $a_i = \pm d^{-1/2}$ is chosen independently and uniformly. A Chernoff-type argument shows that, with probability at least $1-1/20$,
$$\max_{x\in X}\|HDx\|_\infty = O(d^{-1/2}\sqrt{\log n}) \, . \tag{1}$$
Indeed,
$$\mathbf{E}\,[e^{tdu_1}] = \prod_i \mathbf{E}\,[e^{tda_i x_i}] = \prod_i \cosh(t\sqrt{d}\,x_i) \le e^{t^2 d\|x\|_2^2/2}$$
and hence, by Markov's inequality,
$$\begin{aligned}\mathrm{Prob}[|u_1| \ge s] &\le 2\mathbf{E}\,[e^{sdu_1}]/e^{s^2 d} \\ &\le 2e^{s^2 d\|x\|_2^2/2 - s^2 d} \\ &= 2e^{-s^2 d/2} \le 1/(20nd)\end{aligned}$$
for $s = \Theta(d^{-1/2}\sqrt{\log n})$, from which (1) follows by a union bound over all $nd \le n^2$ coordinates of the vectors $\{HDx : x \in X\}$.

Assume from now on that (1) holds, i.e., $\|u\|_\infty \le s$ (where $u = HDx$ for any $x \in X$, which we keep fixed). It is convenient (and harmless) to assume that $m \stackrel{\text{def}}{=} s^{-2}$ is integral.

Note that $\|u\|_2 = \|x\|_2$ because both H and D are isometries. Let $y = (y_1, \ldots, y_k)^T = Pu = \Phi x$. By the definition of the FJLT, y_1 is obtained as follows: pick random i.i.d. indicator variables $b_1, \ldots, b_d$, where each b_j equals 1 with probability q, and random i.i.d. variables $r_1, \ldots, r_d$ distributed $N(0, q^{-1})$. Then set $y_1 = \sum_{j=1}^d r_j b_j u_j$. Let $Z = \sum_{j=1}^d b_j u_j^2$. By the 2-stability of the normal distribution, $(y_1 | Z = z) \sim N(0, q^{-1}z)$. Note that all of $y_1, \ldots, y_k$ are i.i.d (given u), therefore we have corresponding random i.i.d. variables $Z = Z_1, \ldots, Z_k$. Also note that $E[Z] = q$.

The ℓ_1 case: We choose
$$q = \min\{1/(\varepsilon m), 1\} = \min\left\{\Theta\left(\frac{\varepsilon^{-1}\log n}{d}\right), 1\right\} \ .$$
We now bound the moments of Z (over the random b_i's).

LEMMA 2.2. *For any $t > 1$, $\mathbf{E}\,[Z^t] = O(qt)^t$, and*
$$(1-\varepsilon)\sqrt{q} \le \mathbf{E}\,[\sqrt{Z}\,] \le \sqrt{q} \ .$$

PROOF. For the case $q = 1$ the claim is trivial because then Z is constant 1. So we assume $q = 1/(\varepsilon m) < 1$. By Jensen's inequality, $\mathbf{E}\,[\sqrt{Z}\,] \le \sqrt{\mathbf{E}\,[Z]} = \sqrt{q}$. By convexity (resp. concavity), $\max_u \mathbf{E}\,[Z^t]$ (resp. $\min_u \mathbf{E}\,[\sqrt{Z}\,]$) is achieved at a vertex of the polytope
$$\mathcal{P} = \{\, (u_1^2, \ldots, u_d^2) \,|\, u_j^2 \le 1/m \ \forall j \text{ and } \|u\|_2^2 = 1 \,\} \ ,$$
i.e., for $u_j = m^{-1/2}$ ($j \le m$) and $u_j = 0$ (else). It follows that to upper-bound $\mathbf{E}\,[Z^t]$, we may focus on $Z \sim m^{-1}B(m, q)$ (in words, the binomial distribution with parameters m, q multiplied by the constant m^{-1}). Therefore, by standard bounds on the binomial moments,
$$\mathbf{E}\,[Z^t] = O(m^{-t}(mqt)^t) = O(qt)^t,$$
as required by the lemma. We now lower-bound $\mathbf{E}\,[\sqrt{Z}]$: Since $\sqrt{x} \ge 1 + \frac{1}{2}(x-1) - (x-1)^2$ for all $x \ge 0$,
$$\begin{aligned}\mathbf{E}\,[\sqrt{Z}\,] &= \sqrt{q}\,\mathbf{E}\,[\sqrt{Z/q}\,] \\ &\ge \sqrt{q}\left(1 + \frac{1}{2}\mathbf{E}\,[Z/q - 1\,] - \mathbf{E}\,[(Z/q-1)^2\,]\right) \ .\end{aligned} \tag{2}$$
Now note that $\mathbf{E}\,[Z/q - 1\,] = 0$ and
$$\begin{aligned}\mathbf{E}\,[(Z/q-1)^2\,] &= \mathbf{var}(Z/q) = (1-q)/(qm) \\ &\le 1/(qm) = \varepsilon \ .\end{aligned}$$
Plugging this into (2) gives $\mathbf{E}\,[\sqrt{Z}\,] \ge \sqrt{q}\,(1-\varepsilon)$, as required. □

Since the expectation of the absolute value of $N(0,1)$ is $\sqrt{2\pi^{-1}}$, by taking conditional expectations
$$\mathbf{E}\,[|y_1|] = \sqrt{2/q\pi}\,\mathbf{E}\,[\sqrt{Z}\,] \ ,$$
and by Lemma 2.2,
$$(1-\varepsilon)\sqrt{2\pi^{-1}} \le \mathbf{E}\,|y_1| \le \sqrt{2\pi^{-1}} \, . \tag{3}$$
We now show that $\|y\|_1$ is sharply concentrated around its mean $\mathbf{E}\,[\|y\|_1] = k\mathbf{E}\,[|y_1|]$. To do this, first we bound the moments of $|y_1| = |\sum_j b_j r_j u_j|$. For any integer $t \ge 0$, by taking conditional expectation,
$$\mathbf{E}\,[|y_1|^t] = \mathbf{E}\,[(q^{-1}Z)^{t/2}]\,\mathbf{E}\,[|U|^t] \ ,$$
where $U \sim N(0,1)$. It is well known that $\mathbf{E}\,[|U|^t] = O(t)^{t/2}$; therefore, by Lemma 2.2,
$$\mathbf{E}\,[|y_1|^t] = O(t)^t \ .$$
It follows that the moment generating function
$$\begin{aligned}\mathbf{E}\,[e^{\lambda|y_1|}] &= 1 + \lambda\mathbf{E}\,[|y_1|] + \sum_{t>1}\mathbf{E}\,[|y_1|^t]\lambda^t/t! \\ &\le 1 + \lambda\mathbf{E}\,[|y_1|] + \sum_{t>1} O(t)^t\lambda^t/t!\end{aligned}$$
converges for any $0 \le \lambda < \lambda_0$, where λ_0 is an absolute constant, and
$$\mathbf{E}\,[e^{\lambda|y_1|}] = 1 + \lambda\mathbf{E}\,|y_1| + O(\lambda^2) = e^{\lambda\mathbf{E}\,|y_1| + O(\lambda^2)} \ .$$
By independence,
$$E[e^{\lambda\|y\|_1}] = (E[e^{\lambda|y_1|}])^k = e^{\lambda\mathbf{E}\,\|y\|_1 + O(\lambda^2 k)} \ .$$

By Markov's inequality and (3),

$$\begin{aligned}\text{Prob}[\|y\|_1 \geq (1+\varepsilon)\mathbf{E}[\|y\|_1] &\leq E[e^{\lambda\|y\|_1}]/e^{\lambda(1+\varepsilon)\mathbf{E}[\|y\|_1]} \\ &\leq e^{-\lambda\varepsilon\mathbf{E}[\|y\|_1]+O(\lambda^2 k)} \\ &\leq e^{-\Omega(\varepsilon^2 k)} ,\end{aligned}$$

for some $\lambda = \Theta(\varepsilon)$. The constraint $\lambda < \lambda_0$ entails that ε be smaller than same absolute constant. A similar argument leads to a similar lower tail estimate. Our choice of k ensures that, for any $x \in X$, $\|\Phi x\|_1 = \|y\|_1$ deviates from its mean by at most ε with probability at least $1 - 1/20$. By (3), this mean, $k\mathbf{E}[|y_1|]$, is itself concentrated (up to a relative error of ε) around $\alpha_1 = k\sqrt{2\pi^{-1}}$; rescaling ε by a constant factor and ensuring (1) proves the ℓ_1 claim of the first part of the FJLT lemma.

The ℓ_2 case: We now choose

$$q = \min\left\{\frac{c_1 \log^2 n}{d}, 1\right\} ,$$

for a large enough constant c_1.

LEMMA 2.3. *With probability at least $1 - 1/20n$,*

1. *$q/2 \leq Z_i \leq 2q$ for all $i = 1, \ldots, k$, and*
2. *$kq(1-\varepsilon) \leq \sum_{i=1}^k Z_i \leq kq(1+\varepsilon)$.*

PROOF. If $q = 1$ then Z is the constant q and the claim is trivial. Otherwise, $q = c_1 d^{-1}\log^2 n < 1$. For any real λ, the function

$$f_\lambda(u_1^2, \ldots, u_d^2) = \mathbf{E}[e^{\lambda Z}]$$

is convex. Therefore, it achieves its maximum on the vertices of the polytope $\mathcal{P}$ (as in the proof of Lemma 2.2). Hence, as argued before, $\mathbf{E}[e^{\lambda Z}] \leq \mathbf{E}[e^{\lambda Z'}]$, where $Z' \sim m^{-1}B(m,q)$. We conclude the proof of the first part by using standard tail estimates on the scaled binomial Z' derived from bounds on its moment generating function $\mathbf{E}[e^{\lambda Z'}]$ (e.g. [2]), and union bounding.

For the second part, let $S = \sum_{i=1}^k Z_i$. Again, the moment generating function of S is bounded above by that of $S' \sim m^{-1}B(mk, q)$, for which it is immediate to check the required concentration bound. □

Henceforth we will assume that the premise of Lemma 2.3 holds with respect to all choices of $x \in X$. Using the union bound, this happens with probability at least $1 - 1/20$. For each $i = 1, \ldots, k$ the random variable $y_i^2 q/Z_i$ is distributed χ^2 with 1 degree of freedom. It follows that, conditioned on Z_i, $\mathbf{E}[y_i^2] = Z_i/q$ and the moment generating function of y_i^2 is

$$\mathbf{E}[e^{\lambda y_i^2}] = (1 - 2\lambda Z_i/q)^{-1/2} .$$

Given any $\lambda > 0$ less than some fixed λ_0, and for large enough ξ, the moment generating function converges and equals:

$$\mathbf{E}[e^{\lambda y_i^2}] \leq e^{\lambda Z_i/q + \xi\lambda^2 (Z_i/q)^2}$$

(we used the fact that $Z_i/q = O(1)$ from the first part of Lemma 2.3). Therefore, by independence,

$$\mathbf{E}[e^{\lambda\sum_{i=1}^k y_i^2}] \leq e^{\lambda\sum_{i=1}^k Z_i/q + \xi\lambda^2\sum_{i=1}^k (Z_i/q)^2} ,$$

and hence

$$\begin{aligned}\text{Prob}[\sum_{i=1}^k y_i^2 > (1+\varepsilon)\sum_{i=1}^k Z_i/q] &= \text{Prob}[e^{\lambda\sum_{i=1}^k y_i^2} > e^{(1+\varepsilon)\lambda\sum_{i=1}^k Z_i/q}] \\ &\leq \mathbf{E}[e^{\lambda\sum_{i=1}^k y_i^2}]/e^{(1+\varepsilon)\lambda\sum_{i=1}^k Z_i/q} \\ &\leq e^{-\varepsilon\lambda\sum_{i=1}^k Z_i/q + \xi\lambda^2\sum_{i=1}^k (Z_i/q)^2} .\end{aligned} \quad (4)$$

If we plug

$$\lambda = \frac{\varepsilon\sum_{i=1}^k (Z_i/q)}{2\xi\sum_{i=1}^k (Z_i/q)^2}$$

into (4) and assume that ε is sufficiently small, we avoid convergence problems (using Lemma 2.3). Using Lemma 2.3 again, we conclude that

$$\text{Prob}[\sum_{i=1}^k y_i^2 > (1+\varepsilon)k] \leq e^{-\Omega(\varepsilon^2 k)} .$$

A similar technique can be used to bound the left tail estimate. By choosing $k = c\varepsilon^{-2}\log n$ for some large enough c, using the union bound, and possibly rescaling ε, we conclude the ℓ_2 case of the first part of the FJLT lemma.

Running time: Computing Dx takes $O(d)$ time, because D is a diagonal matrix. Computing $H(Dx)$ takes $O(d\log d)$ time using the Walsh-Hadamard transform. Finally, computing $P(HDx)$ takes $O(|P|)$ time, where $|P|$ is the number of nonzeros in P. This number is distributed $B(nk, q)$. It is now immediate to verify that

$$\mathbf{E}[|P|] = O(\varepsilon^{p-4}\log^{p+1} n) .$$

Using a Markov bound, we conclude the proof for the running time guarantee of the FJLT lemma.

This concludes the proof of the FJLT lemma, up to possible rescaling of ε.

□

3. ANN SEARCHING IN EUCLIDEAN SPACE

We give a new ANN algorithm for ℓ_2^d that differs (and improves on) its predecessors in its use of *handles*: a bootstrapping device that allows for fast binary searching. Plugging in the FJLT automatically provides a further improvement. Let P be a set of n points in $\mathbb{R}^d$. Given $x \in \mathbb{R}^d$, let $x_{\min}$ be its nearest neighbor in P. The answer to an *ε-ANN query* for x is any point $p \in P$ such that $\|x-p\|_2 \leq (1+\varepsilon)\|x - x_{\min}\|_2$.

The algorithm has two stages. First (part I), we compute an answer q to an $O(n)$-ANN query for x. This opens the way for a second stage, where we answer the ε-ANN query for x within the (smaller) set $P_x = P \cap \mathcal{B}_2(q, 2\|x-q\|_2)$, where $\mathcal{B}_2(q,r)$ denotes the ℓ_2-ball of radius r centered at q. The key property of P_x is that it contains $x_{\min}$ and the distance from x to its furthest neighbor in P_x is only $O(n\|x - x_{\min}\|_2)$. This sets the stage for a binary search over a bounded domain (part II). Instead of reducing the problem to an ANN query over Hamming cube (as in [30]), we embed P directly into a low-dimensional ℓ_1-space, which we then discretize. We get a two-fold benefit from our use of handles and of the FJLT.

```
O(n)-APPROXIMATION ANN (P, x)
  Choose a random unit vector v ∈ R^d.
  For all p ∈ P
      Compute projection π(p) = v^T (p − x).
  Return point x^v_min = argmin_{p∈P} |π(p)|.
```

3.1 Part I: Linear-factor approximation

The projections $\{ v^T p \mid p \in P \}$ are precomputed and kept in sorted order. Given a query x, finding $x^v_{\min}$ takes $O(d + \log n)$ time. By repeating the procedure $O(\log \delta^{-1})$ times (with independently drawn vectors v) and keeping the point $x^v_{\min}$ nearest to x, we increase the probability of success to the $O(n)$-ANN query to $1 - \delta$.

LEMMA 3.1.

$$\mathbf{E}\ \|x - x^v_{\min}\|_2 = O(n\|x - x_{\min}\|_2)\ .$$

PROOF. Let $\chi(p)$ be the indicator variable of the event $|\pi(p)| \le |\pi(x_{\min})|$. Elementary trigonometry shows that

$$\mathbf{E}\ \chi(p) = O(\|x - x_{\min}\|_2 / \|x - p\|_2)\ .$$

Clearly, $\|x - x^v_{\min}\|_2 \le \sum_{p\in P} \chi(p)\|x - p\|_2$; therefore, by linearity of expectation,

$$\mathbf{E}\ \|x - x^v_{\min}\|_2 \le \sum_{p\in P} \|xp\|_2\, \mathbf{E}\ \chi(p) = O(n\|x - x_{\min}\|_2).$$

□

3.2 Part II: Binary search with handles

Assume that the previous step succeeds in returning an $O(n)$-ANN q for x. From now on, we can confine our search within P_x. Note that there are only $O(n^2)$ distinct sets P_x and, given x, P_x can be found by binary search in $O(d + \log n)$ time. If the diameter of P_x is small enough, say $\Delta(P_x) \le \frac{1}{2}\varepsilon\|x - q\|_2$, then q is a satisfactory answer to the ε-ANN query for x. By precomputing all diameters, we can test this in constant time and be done if the outcome is positive. So, assume now that $\Delta(P_x) > \frac{1}{2}\varepsilon\|x - q\|_2$. The distance from x to any point of P_x lies in the interval $I = [\Omega(\Delta(P_x)/n), 6\varepsilon^{-1}\Delta(P_x)]$, so a binary search will require $\log(n/\varepsilon) + O(1)$ steps. Each step t is associated with two items, l_t and p_t:

1. A *search radius* $l_t \in I$: with high probability, the t-th step in the binary search finds out whether there is a point in P_x at distance at most $(1 + \varepsilon)l_t$ to x (*success*) or whether all points are at distance at least l_t to x (*failure*). For initialization, we set l_1 to the high endpoint of I. For $t > 1$, $l_t = l_{t-1} \pm 2^{-t} l_1$.

2. A *handle* $p_t \in P_x$ such that $\|x - p_t\|_2 \le 2l_t$. We set $p_1 = q$.

Let Φ be the (normalized) FJLT: with high probability, for any pair of points $p, q \in P$, $(1-\varepsilon)\|p - q\|_2 \le \|\Phi p - \Phi q\|_1 \le (1 + \varepsilon)\|p - q\|_2$. As usual, $k = O(\varepsilon^{-2} \log n)$ denotes the embedding dimension.

3.3 Poisson discretization

At step t, for $i = 1, \ldots, k$, consider a random two-sided infinite Poisson process Ψ_i with rate k/l_t. This implies that the number of random points in every interval $[a, a+\tau)$ obeys a Poisson distribution with expectation $\tau k/l_t$. Given $u \in \mathbb{R}^k$, define the quantization T^u of u to be the k-dimensional integer vector, whose i-th coordinate is the signed count of Poisson events between 0 and u_i, ie,

$$T^u_i = \begin{cases} |\Psi_i \cap [0, u_i)| & \text{if } u_i \ge 0 \\ -|\Psi_i \cap [u_i, 0)| & \text{else.} \end{cases}$$

We use T to define a pseudometric D over $\mathbb{R}^d$: $D(x, y) = \|T^{\Phi x} - T^{\Phi y}\|_1$. For fixed $x, y \in \mathbb{R}^d$, $D(x, y)$ is a Poisson variable with rate $k\|\Phi(x-y)\|_1/l_t$. The point process might fail its purpose, so we say that $p \in P_x$ is *reliable at step t* if either (i) $\|x - p\|_2 \le l_t$ and $D(x, p) \le (1 + 2\varepsilon)k$ or (ii) $\|x - p\|_2 \ge (1 + 2\varepsilon)l_t$ and $(1 - 2\varepsilon)k\|x - p\|_2/l_t \le D(x, p) \le (1 + 2\varepsilon)k\|x - p\|_2/l_t$. By our choice of k, concentration bounds [2] for the Poisson distribution show that, at step t, all points of P_x are reliable with high probability. When this happens, we say that step t itself is reliable. (Note that a randomly shifted grid also works but the proof is less elegant.)

3.4 A pruned data structure

Step reliability ensures that all the required information for the binary search is contained in the vector $T^{\Phi x}$ (up to possible rescaling of ε); and so we can use that discrete vector to index a lookup table S^{p_t, l_t}. The entry $S^{p_t,l_t}[v]$ stores the nearest ℓ_1-neighbor of $v \in \{ T^{\Phi p} \mid p \in P_x \}$. Assuming that step t is reliable, then

$$D(x, p_t) \le 2k(1 + 2\varepsilon)\ ; \tag{5}$$

therefore, only the points x in the D-metric ball specified by (5) need to be stored (say, in a pruned k-way tree). For fixed p_t, the number of vectors $T^{\Phi x}$ that satisfy (5) is bounded by the number of integral points $(n_1, \ldots, n_k) \in \mathbb{Z}^k$ such that $n_1 + \cdots + n_k \le 2k(1 + 2\varepsilon)$. This puts the storage requirement at $\binom{3k}{k-1} 2^{O(k)} = 2^{O(k)} = n^{O(\varepsilon^{-2})}$. If we implement the table as a weight-balanced radix tree, the lookup time is only $O(k)$. The complete binary search takes $O(k \log(n/\varepsilon)) = O(\varepsilon^{-2}(\log n/\varepsilon)^2)$ time. Obviously, we may assume $\varepsilon > n^{-O(1)}$ (otherwise the naive algorithm is faster), so the binary search time is $O(\varepsilon^{-2} \log^2 n)$. In preprocessing, we build all possible tables S^{p_t,l_t} for all steps t, all points p_t, and all P_x, which requires a total of $n^{O(\varepsilon^{-2})}$ storage.

THEOREM 3.2. *Given a set P of n points in ℓ_2^d, for any $\varepsilon > 0$, there a randomized data structure of size $n^{O(\varepsilon^{-2})}$ that can answer any ε-ANN query in time $O(d \log d + \varepsilon^{-3} \log^2 n)$ with high probability (over the preprocessing).*

4. ANN SEARCHING OVER THE HAMMING CUBE

Kushilevitz et al. [30] gave an algorithm for ANN queries over $\{0, 1\}^d$. Its bottleneck is the repeated multiplication of the query point by various random matrices. Our improvement is based on the observation that, although most of these matrices are dense, by using some algebra over GF (2),

one can decompose them into a sparse part together with a dense part that is of low complexity.

The ANN data structure of [30] consists of d separate substructures, $\mathcal{S}_l$ $(1 \le l \le d)$, each one meant to handle queries whose targeted nearest neighbors are at the (unknown) distance l. To supply enough randomness so that *every* query succeeds with high probability, each $\mathcal{S}_l$ itself is a collection of σ similarly built data structures $\mathcal{S}_{l,j}$. For any $j = 1, \ldots, \sigma$, $\mathcal{S}_{l,j}$ consists of: (i) a random k-by-d matrix $R^{l,j}$ whose elements are chosen independently in $\{0,1\}$, with the probability of a 1 being $1/2l$; (ii) a table $T_{l,j}$ of pointers to P, indexed by $z \in \{0,1\}^k$, and initialized as follows: Set all entries to ∞; then, for each $p \in P$ in turn, set $T_{l,j}[\mathcal{B}_1(z, k(\mu(l)+\frac{1}{3}\varepsilon'))]$ to p, where $z = R^{l,j}p \pmod 2$, $\mathcal{B}_1(z,r)$ is the Hamming ball of radius r around z, $\mu(l) = \frac{1}{2}(1-(1-\frac{1}{2l})^l)$ and $\varepsilon' = \Theta(1-e^{-\varepsilon/2})$.

LEMMA 4.1. **[30]** *Assume that $n \ge \log d$, and set $\sigma = cd\log d$ and $k = c\varepsilon^{-2}\log n$, for a large enough constant c. With high probability, the following holds true for any $x \in \{0,1\}^d$ and any $1 \le l \le d$: Given a random $j \in \{1, \ldots, \sigma\}$, with high probability, the point $T_{l,j}[R^{l,j}x]$, if finite, is at distance at most $(1+\varepsilon)l$ from x. Furthermore, if x's nearest neighbor is at distance at most l, then the point in question, indeed, is finite.*

For a random j, we say that a query point x *passes the l-test* if the point $T_{l,j}[R^{l,j}x]$ is finite. (Note that passing is not an intrinsic property of x but a random variable.) The test is called *reliable* if both of the high-probability events in the lemma hold. Assuming reliability, failure of the test means that x's nearest neighbor lies at distance greater than l, while success yields a neighbor of x at distance at most $(1+\varepsilon)l$.

This immediately suggests an ANN algorithm. Beginning with $l = \lceil d/2 \rceil$, run an l-test on x and repeat for $l/2$ if it passes and $3l/2$ if it fails; then, proceed in standard binary search fashion. Suppose for a moment that all the l-tests are reliable. Then, the binary search terminates with the discovery of an index l and a point $p \in P$ that is at most $(1+\varepsilon)l$ away from x, together with the certainty that the distance from x to its nearest neighbor exceeds l. Obviously, the point p is an acceptable answer to the ε-ANN query.

We cannot count on the reliability of every test used in the binary search. But, as in [30], we can overcome this problem by using the fault-tolerant techniques of [17] for computing with unreliable information. Note also that we may assume from now on that $n \ge \log d$: Indeed, having fewer than $\log d$ points gives us a naive (exact) algorithm with $O(d\log d)$ query time. The storage is $d^2 n^{O(\varepsilon^{-2})}$ and the query time is $O(d(\log d)\varepsilon^{-2}\log n)$. To improve this time bound, we seek to exploit the sparsity of the random matrices. That alone cuts down the query time to $O(d\varepsilon^{-2}\log n)$ in a trivial manner, the worst case being a query x that itself belongs to P. We call this the *easy-sparse method*.

Linear algebra gives room for further improvement. For expository purposes, it is convenient to start the binary search with a d-test, so that the first test in the search is always successful. In general, consider the case where an l-step is to be performed and let l' be the last previous successful test in the search. Note that $l \ge l'/2$. The algorithm is now in possession of a *handle*, ie, a point $p \in P$ at distance at most $(1+\varepsilon)l'$ from x. The cost of the current l-test is that of computing $y = R^{l,j}x$ for a random j.

The main idea is to evaluate y as $R^{l,j}x = R^{l,j}(x+2p) = R^{l,j}(x+p) + R^{l,j}p$ over GF (2). Here is the benefit of this decomposition: The point $x+p$ has at most $(1+\varepsilon)l'$ ones, and obviously only the corresponding columns of $R^{l,j}$ are relevant in computing $R^{l,j}(x+p)$. Assuming that the 1's within each column are linked together in a list, the time for computing $R^{l,j}(x+p)$ is proportional to $d+k$ plus the number of ones within the relevant columns. This last number is at most $k(1+\varepsilon)l'(1/2l) \le 2k$ in expected value (over the randomness of the matrix). By precomputing all the points $\{\, R^{l,j}q \mid q \in P \,\}$ in preprocessing (which adds only a factor of n to the storage), we can retrieve $R^{l,j}p$ in $O(k)$ time. In short, we can complete this binary search step in $O(d+k)$ expected time, instead of the previous $O(dk)$ bound.

There is only one problem: The expectation of the query time is defined over the randomness of *both* the query algorithm and the preprocessing. To remove this dependency on the preprocessing, we must ensure that, for *any* query x, the expected running time of any binary search step is $O(d+k)$ over the random choices of the index j during query answering: We call this the *NQLB policy* (for "no query left behind").

It suffices to show that, for any l and any subset $V \subseteq 2^{\{1,\ldots,d\}}$ of column indices, the total number of ones within all the columns (indexed by V) of all the matrices $R^{l,j}$ $(1 \le j \le \sigma)$ is $O(\sigma k|V|/l)$. This number is a random variable $Y = \sum_{1 \le i \le \sigma k|V|} y_i$, where each y_i is chosen independently in $\{0,1\}$ with a probability $1/2l$ of being 1. A Chernoff bound shows that $Y = O(\sigma k|V|/l)$ with probability at least $1 - 2^{-\Omega(\sigma k|V|/l)}$. Summing over all l, V, we find that the probability of violating the NQLB policy is at most

$$\sum_{l=1}^{d}\sum_{v=1}^{d}\binom{d}{v}2^{-\Omega(\sigma k v/l)},$$

which is arbitrary small.

THEOREM 4.2. *Given a set P of n points in the d-dimensional Hamming cube and any $0 < \varepsilon < 1$, there exists a data structure of size $d^2 n^{O(\varepsilon^{-2})}$ that can answer any ε-ANN query in time $O((d+\varepsilon^{-2}\log n)\log d)$.*

The preprocessing is randomized but succeeds with high probability. Success implies that every query is answered correctly with high probability. Similarly, the expectation of the query time applies uniformly to every query.

5. FUTURE WORK

The FJLT can potentially improve other proximity-related problems such as closest pair, furthest neighbor and clustering. The ANN application presented here suffers from the $n^{O(1/\varepsilon^2)}$-space requirement, an almost insurmountable implementation bottleneck for small ε. It is natural to ask if the space and time could be traded off so that an algorithm with running time $O(\varepsilon^{-2}d\log n)$ (comparable to [19,26] and [30]) uses significantly less space.

The Kac random walk. We propose an alternative FJLT transform which we conjecture to be at least as good as the one described in this paper, yet much more elegant. This transform is based on the following random walk on the orthogonal group on $\mathbb{R}^{d\times d}$, defined by Kac [28]. At time $t = 0$, the random walk is at the identity matrix:

$U_0 = Id$. At time $t > 0$, we choose two random coordinates $1 \leq i_t < j_t \leq d$ and a random angle $\theta_t \in [0, 2\pi)$, and set $U_{t+1} = R_{i_t,j_t,\theta_t} U_t$, where $R_{i,j,\theta}$ is a rotation of the (i,j)-plane by angle θ. Clearly U_t is an orthogonal matrix for all $t \geq 0$. The walk has the Haar measure on the group of orthogonal matrices as its unique stationary distribution. For any fixed $x \in \mathbb{R}^d$, computation of $U_T x$ is extremely efficient: for $t = 1, \ldots, T$ replace x_{i_t} (resp. x_{j_t}) with $x_{i_t} \cos\theta_t + x_{j_t} \sin\theta_t$ (resp. $-x_{i_t} \sin\theta_t + x_{j_t} \cos\theta_t$). The Kac version of FJLT is defined as follows: compute $U_T x$ for all vectors $x \in X \subseteq \mathbb{R}^d$, and return the projection onto the first $O(\varepsilon^{-2} \log |X|)$ coordinates of the resulting vectors as the reduced-dimension space. How small can T be in order to ensure the same guarantee as the original JL dimension-reduction technique?

Kac defined this walk in the context of statistical physics in an attempt to simplify and understand Boltzmann's equation. Since then, much attention has been given to it from the viewpoints of pure and applied mathematics. For example, it can be used to efficiently estimate high-dimensional spherical integrals [20]. Its spectral properties are by now well understood [9, 15, 33], though spectral techniques may not be suitable for the weak notion of mixing rate we are seeking. Based on experiments and preliminary results, we conjecture that $T = O(d \log d + poly(\log n, \varepsilon^{-1}))$ steps suffice, and propose this as an interesting open problem

Improvements to FJLT and lower bounds. It is natural to ask what is the fastest randomized linear mapping with the Johnson-Lindenstrauss guarantee. More precisely, we pose the following question:

QUESTION 5.1. *What is the lower bound on the expected depth of a randomized linear circuit $C_{n,d} : \mathbb{R}^d \mapsto \mathbb{R}^{O(\varepsilon^{-2} \log n)}$ such that given any set $X \subseteq \mathbb{R}^d$ of n vectors, with probability at least $2/3$, $\alpha\|x\|_2(1-\varepsilon) \leq \|C_{n,d}(x)\|_p \leq \alpha\|x\|_2(1+\varepsilon)$ for all $x \in X$, for some $\varepsilon > 0$, $p \in \{1, 2\}$ and α ?*

Acknowledgments

For their kind willingness to answer our numerous questions, we wish to thank Nina Gantert, Anupam Gupta, Sariel Har-Peled, Piotr Indyk, Elchanan Mossel, Yuval Peres, and Yuval Rabani.

6. REFERENCES

[1] Achlioptas, D. *Database-friendly random projections: Johnson-Lindenstrauss with binary coins*, Journal of Comp. & Sys. Sci. 66 (2003), 671–687.

[2] Alon, N., Spencer, J. *The probabilistic method*, John Wiley, 2nd edition, 2000.

[3] Alon, N. *Problems and results in extremal combinatorics*, I, Discrete Math. 273 (2003), 31–53.

[4] Arya, S., Mount, D.M. *Approximate nearest neighbor searching*, Proc. 4th Annu. ACM-SIAM Symp. Disc. Alg. (1993), 271–280.

[5] Arya, S., Mount, D.M., Netanyahu, N.S., Silverman, R., Wu, A. *An optimal algorithm for approximate nearest neighbor searching*, J. ACM 45 (1998), 891–923.

[6] Bern, M. *Approximate closest-point queries in high dimensions*, Inform. Process. Lett. 45 (1993), 95–99.

[7] Bingham, E., Mannila, H. *Random projection in dimensionality reduction: applications to image and text data*, Knowledge Discovery and Data Mining (2001), 245–250.

[8] Borodin, A., Ostrovsky, R., Rabani, Y. *Lower bounds for high dimensional nearest neighbor search and related problems*, Proc. 31st STOC (1999), 312–321.

[9] Carlen, E. A., Carvalho, M. C., Loss, M. *Determination of the Spectral Gap for Kac's Master Equation and Related Stochastic Evolutions*, Preprint arXiv:math-ph/0109003, (2000)

[10] Chakrabarti, A., Regev, O. *An optimal randomised cell probe lower bound for approximate nearest neighbor searching*, Proc. 44th FOCS (2004).

[11] Chan, T. *Approximate nearest neighbor queries revisited*, Proc. 13th Annu. ACM Symp. Comput. Geom. (1997), 352–358.

[12] Clarkson, K.L. *An algorithm for approximate closest-point queries*, Proc. 10th Annu. ACM Symp. Comput. Geom. 10 (1994), 160–164.

[13] Clarkson, K.L. *Nearest neighbor queries in metric spaces*, Proc. 29th Annu. ACM Sympos. Theory Comput., 1997.

[14] Dasgupta, S., Gupta, A. *An elementary proof of the Johnson-Lindenstrauss lemma*, Technical Report 99-006, UC Berkeley, March 1999.

[15] Diaconis, P., Saloff-Coste, L. *Bounds for Kac's Master Equation*, Communications in Mathematical Physics 209(3), (2000), 729–755

[16] Farach-Colton, M., Indyk, P. *Approximate nearest neighbor algorithms for Hausdorff metrics via embeddings*, Proc. 40th FOCS (1999).

[17] Feige, U., Peleg, D., Raghavan, P., Upfal, E. *Computing with unreliable information*, Proc. 20nd STOC (1990), 128–137.

[18] Frankl, P., Maehara, H. *The Johnson-Lindenstrauss lemma and the sphericity of some graphs*, Journal of Combinatorial Theory Series A, 44 (1987), 355–362.

[19] Har-Peled, S. *A replacement for Voronoi diagrams of near linear size*, Proc. FOCS (2001), 94–103.

[20] Hastings, W. *Monte Carlo sampling methods using Markov chains and their applications*, Biometrica 57, 97–109

[21] Indyk, P. *On approximate nearest neighbors in non-Euclidean spaces*, Proc. 39th FOCS (1999).

[22] Indyk, P. *High-dimensional computational geometry*, Thesis (2000), Stanford University.

[23] Indyk, P. *Dimensionality reduction techniques for proximity problems*, Proc. SODA (2000), 371–378.

[24] Indyk, P. *Nearest neighbors in high-dimensional spaces*, Handbook of Discrete and Computational Geometry, eds., J.E. Goodman and J. O'Rourke, CRC Press (2004).

[25] Indyk, P., Matousek, J. *Low-distortion embeddings of finite metric spaces*, Handbook of Discrete and Computational Geometry, eds., J.E. Goodman and J. O'Rourke, CRC Press (2004).

[26] Indyk, P., Motwani, R. *Approximate nearest neighbors:*

towards removing the curse of dimensionality, Proc. 30th STOC (1998), 604–613.

[27] Johnson, W.B., Lindenstrauss, J. *Extensions of Lipschitz mappings into a Hilbert space*, Contemp. Math. 26 (1984), 189–206.

[28] Kac, M. *Probability and related topics in physical science*, Wiley Interscience, N.Y.

[29] Kleinberg, J. *Two algorithms for nearest neighbor search in high dimensions*, Proc. 29th STOC (1997), 599–608.

[30] Kushilevitz, E., Ostrovsky, R., Rabani, Y. *Efficient search for approximate nearest neighbor in high-dimensional spaces*, SIAM J. Comput. 30 (2000), 457–474.

[31] Matousek, J. *Lectures on Discrete Geometry*, Springer, May 2002.

[32] Muthukrishnan, S., Sahinalp, S. C., *Simple and practical sequence nearest neighbors with block operations*, Proc. 13th Annual Symposium on Combinatorial Pattern Matching (2002)

[33] Pak, I. *Using Stopping Times to Bound Mixing Times*, Proc. SODA (1998)

[34] Yianilos, P.N. *Data structures and algorithms for nearest neighbor search in general metric spaces*, Proc. 2nd Annual ACM-SIAM Symp. Disc. Alg. (1993), 311–321.

[35] Yianilos, P.N. *Locally lifting the curse of dimensionality for nearest neighbor search*, Proc. SODA (2000), 361–370.