

Dimension Reduction

and nearest-neighbor search

Preserve distances, distinguish neighbors, reduce search

Random projections compress high-dimensional geometry. A radius-sensitive sketch separates near points from far ones, while locality-sensitive hashing turns a collision gap into a search index. The central questions are what the representation preserves and how that promise becomes an algorithm with explicit space, time, and error guarantees.

Advanced Algorithms teaching team

Nanjing University

Lecture 5 - randomized algorithms and randomization techniques

Learning goals and reading routes

This lecture uses concentration to build short representations of high-dimensional points and to search a large dataset. By the end, you should be able to:

1. prove the Johnson–Lindenstrauss theorem from a one-vector tail bound and a union bound;
2. explain Gaussian, random-subspace, and random-sign constructions, including their scaling;
3. distinguish approximate nearest neighbor from fixed-radius approximate near neighbor;
4. analyze both a Hamming signature table and a locality-sensitive hashing index; and
5. combine radius-specific structures while accounting for approximation, failure probability, and actual query cost.

For a first reading, follow the Gaussian JL proof in Section 2, then its connection to search in Section 4. Sections 5–7 develop the Hamming signature table, locality-sensitive hashing, and the near-to-nearest reduction. Each construction explains its purpose before its proof and cost analysis.

The alternative JL proofs in Section 3 can be read immediately after the Gaussian proof or on a second pass. Section 8 compares the methods and gives research perspectives. The appendices supply Gaussian, geometric, and probability tools. All main algorithms and implementation guarantees remain in the main text.

Throughout, $[n] = \{1, \dots, n\}$ and $\ln$ is the natural logarithm. Probabilities concern the algorithm's random choices for inputs fixed independently of those choices, unless stated otherwise. Arithmetic, bit, and word-operation costs are distinguished where needed. A returned point is represented by its identifier. Constants with a subscript, such as $O_c(\cdot)$, may depend on that parameter.

Contents

Learning goals and reading routes	i
1 Geometry, representations, and search	1
2 Johnson–Lindenstrauss dimension reduction	2
3 Other Euclidean transforms: orthogonality and random signs	4
4 From distance preservation to approximate search	7
5 A Hamming sketch and a polynomial-space lookup table	8
6 Locality-sensitive hashing: search only a few buckets	11
7 From near neighbor to nearest neighbor	15
8 Putting the methods in perspective	17
9 Exercises	19
A Gaussian and Euclidean tools	20
B Probability tools for collision and gap arguments	22
Bibliographic notes	23

1 Geometry, representations, and search

High-dimensional data raise two costs: computing a distance and finding a close point among many candidates. Dimension reduction addresses the first; a search index must also address the second. We develop both ideas and distinguish their guarantees.

A *metric space* is a set X with a function $d_X : X \times X \rightarrow \mathbb{R}_{\geq 0}$ satisfying, for all $x, y, z \in X$,

$$d_X(x, y) = 0 \iff x = y, \quad d_X(x, y) = d_X(y, x), \quad d_X(x, z) \leq d_X(x, y) + d_X(y, z).$$

The Euclidean distance on $\mathbb{R}^d$ is $\|x - y\|_2$, where $\|x\|_2 = (\sum_{j=1}^d x_j^2)^{1/2}$. On $\{0, 1\}^d$, the Hamming distance is

$$\text{Ham}(x, y) = \sum_{j=1}^d \mathbf{1}\{x_j \neq y_j\} = \|x - y\|_1 = \|x - y\|_2^2. \quad (1)$$

The last equality is particular to Boolean vectors. On general real vectors, squared Euclidean distance is not a metric: for the points 0, 1, 2 on a line, $4 > 1 + 1$ violates the triangle inequality.

An embedding $\phi : X \rightarrow Y$ changes the representation of points. We quantify its quality by explicit inequalities. One symmetric convention is

$$\alpha^{-1} d_X(x, y) \leq d_Y(\phi(x), \phi(y)) \leq \alpha d_X(x, y), \quad \alpha \geq 1.$$

Another convention allows a scale $a > 0$ and writes $ad_X \leq d_Y \leq Dad_X$; its distortion is D . The symmetric inequalities imply $D \leq \alpha^2$, not necessarily $D \leq \alpha$.

Euclidean dimension reduction

Given a fixed set $S = \{x_1, \dots, x_n\} \subseteq \mathbb{R}^d$ and $0 < \varepsilon < 1$, construct $\phi : \mathbb{R}^d \rightarrow \mathbb{R}^k$ such that every pair in S satisfies

$$(1 - \varepsilon)\|x - y\|_2^2 \leq \|\phi(x) - \phi(y)\|_2^2 \leq (1 + \varepsilon)\|x - y\|_2^2. \quad (2)$$

The aim is $k \ll d$ with an efficiently computable map.

Taking square roots gives factors $\sqrt{1 - \varepsilon}$ and $\sqrt{1 + \varepsilon}$ for distances. In particular, these imply the weaker factors $1 - \varepsilon$ and $1 + \varepsilon$. Conversely, a $(1 \pm \eta)$ guarantee for distances gives factors $(1 \pm \eta)^2$ for squared distances. We work with squared norms because they are sums, and convert the approximation factor explicitly when applying a map to search.

The Johnson–Lindenstrauss theorem preserves *all pairwise Euclidean distances in a fixed finite set*. A Hamming sketch instead separates $\text{Ham}(x, y) \leq r$ from $\text{Ham}(x, y) > cr$ at a *chosen radius* r . Locality-sensitive hashing makes near pairs collide more often than far pairs, limiting the number of candidates examined.

Three questions, three guarantees

How accurately is a distance represented? How reliably does a sketch distinguish a near pair from a far pair? How much work is required to retrieve and verify a candidate? These are related questions, but none of their answers automatically supplies the other two.

2 Johnson–Lindenstrauss dimension reduction

2.1 The theorem and the algorithm

Theorem 2.1 (Johnson–Lindenstrauss). *Let $S \subseteq \mathbb{R}^d$ contain $n \geq 2$ points, and let $0 < \epsilon, \delta < 1$. Set*

$$k_0 = \left\lceil \frac{8}{\epsilon^2} \ln \frac{n(n-1)}{\delta} \right\rceil. \quad (3)$$

If $A \in \mathbb{R}^{k_0 \times d}$ has independent $\mathcal{N}(0, 1/k_0)$ entries, then with probability at least $1 - \delta$, $x \mapsto Ax$ satisfies (2) simultaneously for every pair in S . Consequently, there exists such a linear embedding into dimension at most $\min\{d, k_0\}$.

For the final assertion, use the identity map when $d \leq k_0$. The theorem is usually summarized as $k = O(\epsilon^{-2} \log n)$ for constant failure probability, or even for failure probability $1/n$. The ambient dimension does not enter this upper bound on the number of output coordinates [1, 4].

Gaussian Johnson–Lindenstrauss transform

Choose k as in (3). Draw independent $G_{ij} \sim \mathcal{N}(0, 1)$ and set $A = G/\sqrt{k}$. Store A and replace each input point x_i by Ax_i . When $k \geq d$, the identity map is an equally valid choice with no distortion.

Each new coordinate is a random linear combination of *all* the old coordinates. Sampling a small set of original coordinates is different: for $S = \{0, e_1\}$, a uniformly selected k -element coordinate subset misses the only nonzero coordinate with probability $1 - k/d$. No rescaling repairs the zero vector produced on that event.

2.2 From one vector to all pairs

We first seek a guarantee for one fixed nonzero vector v . Linearity then gives $Ax - Ay = A(x - y)$. For each distinct pair define

$$u_{xy} = \frac{x - y}{\|x - y\|_2}, \quad \|u_{xy}\|_2 = 1.$$

Thus it suffices to control $\|Au\|_2^2$ for a fixed unit vector u . The desired bound is

$$\mathbb{P}(|\|Au\|_2^2 - 1| > \epsilon) \leq 2e^{-k\epsilon^2/8}. \quad (4)$$

Only after proving this do we take a union bound over the $\binom{n}{2}$ vectors u_{xy} . The bad events for different pairs need not be independent. Repeated points cause no difficulty: their zero differences are preserved exactly.

The order of the quantifiers

The guarantee is: for every fixed u , a fresh random A succeeds with high probability. It is not: with high probability, A succeeds for every $u \in \mathbb{R}^d$. If $k < d$, every linear A has a nonzero vector in its kernel, so simultaneous norm preservation on the entire space is impossible.

2.3 A Gaussian projection gives a chi-square variable

A standard Gaussian $Z \sim \mathcal{N}(0, 1)$ has density $(2\pi)^{-1/2}e^{-z^2/2}$, mean 0, and variance 1. More generally, $\mathcal{N}(\mu, \sigma^2)$ is the distribution of $\mu + \sigma Z$, with $\sigma > 0$. Its second parameter is the variance, not the standard deviation.

Independent Gaussian variables are stable under linear combinations: if $G_1, \dots, G_d$ are independent standard Gaussians, then

$$\sum_{j=1}^d v_j G_j \stackrel{d}{=} \|v\|_2 Z. \quad (5)$$

Appendix A proves this using rotational invariance. For a fixed unit u , different rows of G use independent randomness. Hence

$$(Au)_i \stackrel{d}{=} \frac{Z_i}{\sqrt{k}}, \quad \|Au\|_2^2 \stackrel{d}{=} \frac{1}{k} \sum_{i=1}^k Z_i^2, \quad (6)$$

where the Z_i are independent standard Gaussians. Their squared sum is said to have the *chi-square distribution with k degrees of freedom*, denoted χ_k^2 . In particular,

$$\mathbb{E}\|Au\|_2^2 = 1, \quad \text{Var}(\|Au\|_2^2) = 2/k.$$

The variance identity follows from $\mathbb{E}Z^4 = 3$. Chebyshev alone would give $2/(k\varepsilon^2)$, too weak to obtain $k = O(\log n)$ after a union over all pairs. We need an exponential tail. Since Z_i^2 is unbounded, ordinary bounded-variable Hoeffding is not directly applicable.

2.4 Exponential tails for the chi-square distribution

Lemma 2.2 (The exponential moment of a squared Gaussian). For $Z \sim \mathcal{N}(0, 1)$,

$$\mathbb{E}e^{\lambda Z^2} = (1 - 2\lambda)^{-1/2} \quad \text{for } \lambda < 1/2. \quad (7)$$

For $\lambda \geq 1/2$, the expectation is infinite.

Proof. For $\lambda < 1/2$, substitute $y = \sqrt{1 - 2\lambda} z$ in

$$\frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} e^{-(1-2\lambda)z^2/2} dz.$$

The result is $(1 - 2\lambda)^{-1/2}$ times the integral of a standard Gaussian density. If $1 - 2\lambda \leq 0$, the integrand does not decay and its integral diverges. $\square$

Lemma 2.3 (Chi-square concentration). Let $Q = \sum_{i=1}^k Z_i^2 \sim \chi_k^2$. For $0 < \varepsilon < 1$,

$$\mathbb{P}(Q \geq (1 + \varepsilon)k) \leq e^{-\frac{k}{2}(\varepsilon - \ln(1+\varepsilon))} \leq e^{-k\varepsilon^2/8}, \quad (8)$$

$$\mathbb{P}(Q \leq (1 - \varepsilon)k) \leq e^{-\frac{k}{2}(-\varepsilon - \ln(1-\varepsilon))} \leq e^{-k\varepsilon^2/4}. \quad (9)$$

Proof. Independence and Lemma 2.2 give $\mathbb{E}e^{\lambda Q} = (1 - 2\lambda)^{-k/2}$ for $\lambda < 1/2$. For $0 < \lambda < 1/2$, exponential Markov yields

$$\mathbb{P}(Q \geq (1 + \varepsilon)k) \leq \exp\left(-\lambda(1 + \varepsilon)k - \frac{k}{2} \ln(1 - 2\lambda)\right).$$

The minimizing parameter is $\lambda = \varepsilon/[2(1 + \varepsilon)]$, which is in the allowed interval. Substitution gives the first upper-tail expression. Moreover,

$$\varepsilon - \ln(1 + \varepsilon) = \int_0^\varepsilon \frac{t}{1+t} dt \geq \int_0^\varepsilon \frac{t}{2} dt = \varepsilon^2/4.$$

For the lower tail take $\lambda < 0$. On $Q \leq (1 - \varepsilon)k$, the value of $e^{\lambda Q}$ is at least $e^{\lambda(1-\varepsilon)k}$. The same calculation, now minimized at $\lambda = -\varepsilon/[2(1 - \varepsilon)]$, gives the first expression in (9). Finally,

$$-\varepsilon - \ln(1 - \varepsilon) = \int_0^\varepsilon \frac{t}{1-t} dt \geq \varepsilon^2/2.$$

Both tails have now been proved; the lower-tail argument is not an application of symmetry of Q . $\square$

Proof of Theorem 2.1. Equation (6) and Lemma 2.3 imply (4). A union bound over unordered distinct pairs gives total failure probability at most

$$\binom{n}{2} 2e^{-k\varepsilon^2/8} = n(n-1)e^{-k\varepsilon^2/8} \leq \delta$$

for the integer k in (3). Homogeneity converts the unit-vector estimates back to the original distances. The identity map handles $d \leq k_0$ in the dimension-minimizing version. $\square$

2.5 Computational cost and what the theorem does not promise

In a real-arithmetic model with Gaussian sampling as a primitive, constructing the dense matrix costs $O(kd)$, transforming the dataset costs $O(nkd)$, and applying it to a new vector costs $O(kd)$. Storing the map and the transformed points uses $O(kd + nk)$ real numbers. A sparse input vector with s nonzero coordinates can be transformed in $O(ks)$ arithmetic operations by summing the corresponding matrix columns.

These are arithmetic-operation bounds, not finite-precision bit bounds. Exact Gaussian samples are not finite bit strings; rounding analysis requires an input-precision model and an error budget. The sign construction in Section 3.2 avoids Gaussian sampling, but numerical input arithmetic still has a cost.

A scan through all transformed points costs $O(nk)$ after transforming the query. This may improve the dependence on d , but the number of candidates remains n . A dimension-reduction theorem is not by itself a sublinear-time search index. Section 4 makes the connection to approximate search precise.

3 Other Euclidean transforms: orthogonality and random signs

The Gaussian proof separates two ingredients: a one-vector exponential tail and a union bound. We now keep the second ingredient and change the random map. This shows which features of the Gaussian construction are essential, and which are merely convenient.

3.1 Projection onto a uniformly random subspace

Let $1 \leq k < d$. Choose an orthonormal k -frame $q_1, \dots, q_k$ in $\mathbb{R}^d$ with a rotation-invariant distribution, and let Q have these vectors as rows. Thus $QQ^\top = I_k$. The coordinates of the orthogonal projection onto their span are Qx . Define

$$Tx = \sqrt{d/k} Qx. \quad (10)$$

The factor $\sqrt{d/k}$ compensates for the expected fraction k/d of squared length retained by projection. The rows of Q are orthogonal, not independent. One construction takes k independent Gaussian vectors and applies Gram-Schmidt; with probability one they are linearly independent. Rotating the input Gaussian vectors rotates the resulting frame, giving the required invariance.

Fix a unit vector u . By rotational invariance, projecting this fixed vector onto a random subspace has the same squared-length distribution as projecting a uniform point on the unit sphere onto the first k coordinate directions. Appendix A justifies this symmetry. For independent standard Gaussians $G_1, \dots, G_d$, the vector $G/\|G\|_2$ is uniform on that sphere. Consequently,

$$\|Qu\|_2^2 \stackrel{d}{=} \frac{U}{U+V}, \quad U = \sum_{i=1}^k G_i^2, \quad V = \sum_{i=k+1}^d G_i^2. \quad (11)$$

Here U and V are independent, whereas U and $U+V$ are not. The ratio has mean k/d , since the d squared coordinates of a uniform unit vector are exchangeable and sum to one.

Lemma 3.1 (A random-subspace tail). *For $0 < \beta < d/k$, $\beta \neq 1$, the probability of $U/(U+V) \geq \beta k/d$ when $\beta > 1$, or of $U/(U+V) \leq \beta k/d$ when $\beta < 1$, is at most*

$$\beta^{k/2} \left(1 + \frac{(1-\beta)k}{d-k} \right)^{(d-k)/2} \leq \exp \left(\frac{k}{2} (\ln \beta + 1 - \beta) \right). \quad (12)$$

Proof. Put $H = (d - \beta k)U - \beta kV$. The upper event is $H \geq 0$; the lower event is $H \leq 0$. Choose

$$t = \frac{\beta - 1}{2\beta(d - \beta k)}.$$

Its sign matches that of $\beta - 1$, so the relevant bad event implies $e^{tH} \geq 1$ in both cases. The chi-square MGF and independence give

$$\mathbb{P}(\text{bad event}) \leq \mathbb{E}e^{tH} = (1 - 2t(d - \beta k))^{-k/2} (1 + 2t\beta k)^{-(d-k)/2}.$$

Both factors are finite: their bases at this choice of t are $1/\beta > 0$ and $(d - k)/(d - \beta k) > 0$. Substituting these values proves the first expression in (12). The inequality $\ln(1 + z) \leq z$, valid for $z > -1$, gives the second expression. $\square$

Apply the lemma with $\beta = 1 - \varepsilon$ and $\beta = 1 + \varepsilon$. If $(1 + \varepsilon)k \geq d$, the upper event has probability zero (the ratio is strictly below one almost surely). Otherwise the lemma applies. The logarithmic estimates from Lemma 2.3 give

$$\mathbb{P}(|\|Tu\|_2^2 - 1| > \varepsilon) \leq e^{-k\varepsilon^2/4} + e^{-k\varepsilon^2/8} \leq 2e^{-k\varepsilon^2/8}. \quad (13)$$

Thus the same dimension choice and pairwise union bound prove JL for (10). When $k = d$, any orthogonal matrix is an exact isometry and the scaling is one. This rotation-based proof is associated with the original JL construction and its elementary analysis by Dasgupta and Gupta [1, 2].

Generating the frame by Gram-Schmidt takes $O(dk^2)$ arithmetic operations, and applying the stored frame to one vector takes $O(dk)$. Gaussian rows avoid this orthogonalization step. Neither construction needs the input set to be in a random orientation.

3.2 Random signs also suffice

Spherical symmetry is not necessary for JL. Let $R \in \{-1, 1\}^{k \times d}$ have independent fair signs and set $A = R/\sqrt{k}$. This construction uses exactly kd random bits. The matrix-vector product uses additions and subtractions, followed by one common scaling. It is a basic form of the binary-coin construction [3].

Theorem 3.2 (JL with independent signs). *For the sign matrix above, every fixed unit $u \in \mathbb{R}^d$ and $0 < \varepsilon < 1$ satisfy*

$$\mathbb{P}(|\|Au\|_2^2 - 1| > \varepsilon) \leq 2e^{-k\varepsilon^2/8}. \quad (14)$$

Consequently Theorem 2.1 holds with the same sufficient dimension k_0 for this construction.

Proof. For one unscaled row, write $Y = \sum_j u_j \varepsilon_j$. Independence, centering, and expansion of the fourth power give

$$\mathbb{E}Y^2 = 1, \quad \mathbb{E}Y^4 = \sum_j u_j^4 + 6 \sum_{j < \ell} u_j^2 u_\ell^2 = 3 - 2 \sum_j u_j^4 \leq 3. \quad (15)$$

Also, $\cosh s \leq e^{s^2/2}$ implies

$$\mathbb{E}e^{aY} = \prod_j \cosh(au_j) \leq e^{a^2/2}.$$

To bound the square's positive exponential moment, introduce an independent standard Gaussian Z . The identity $\mathbb{E}_Z e^{bZ} = e^{b^2/2}$ gives, for $0 \leq \lambda < 1/2$,

$$\mathbb{E}_Y e^{\lambda Y^2} = \mathbb{E}_Z \mathbb{E}_Y e^{\sqrt{2\lambda}ZY} \leq \mathbb{E}_Z e^{\lambda Z^2} = (1 - 2\lambda)^{-1/2}. \quad (16)$$

The interchange is legitimate because the integrand is nonnegative. Independent rows now give the same upper-tail bound as (8).

For the lower tail use $e^{-z} \leq 1 - z + z^2/2$ for $z \geq 0$. Equation (15) implies

$$\mathbb{E}e^{-\lambda Y^2} \leq 1 - \lambda + \frac{3}{2}\lambda^2 \leq e^{-\lambda + 3\lambda^2/2} \quad (\lambda \geq 0).$$

For the independent row values $Y_1, \dots, Y_k$, exponential Markov therefore yields

$$\mathbb{P}\left(\sum_i Y_i^2 \leq (1 - \varepsilon)k\right) \leq e^{k(-\lambda\varepsilon + 3\lambda^2/2)} \leq e^{-k\varepsilon^2/6},$$

where $\lambda = \varepsilon/3$. Combine this with the upper tail, then use the same pairwise union bound as before. In general $\sum_i Y_i^2$ is *not* chi-square; its exponential moments, rather than its distribution, are what we compare. $\square$

3.3 What the three constructions have in common

Map	One-vector analysis	Randomness and computation
$G/\sqrt{k}$	A normalized chi-square sum.	Independent Gaussian entries; dense product.
$\sqrt{d/k}Q$	Rotation and a ratio of two independent chi-square sums.	Orthonormal frame; generation includes orthogonalization.
$R/\sqrt{k}$	Exponential-moment comparison for squared sign sums.	Independent random bits; additions and subtractions.

All three give a distribution over linear maps independent of the dataset, and all use the same finite-set union-bound argument. A small target dimension does not mean the matrix is sparse or fast to apply. Sparse and structured transforms address that separate implementation question; Section 8 briefly discusses these directions.

4 From distance preservation to approximate search

4.1 Nearest neighbor and the fixed-radius problem

Let (X, dist) be a metric space and $S = \{y_1, \dots, y_n\} \subseteq X$ a nonempty dataset. For a query $x \in X$, write

$$d_*(x) = \min_{y \in S} \text{dist}(x, y).$$

Exact nearest-neighbor search returns a minimizer. A c -approximate nearest neighbor, for $c > 1$, is any $y \in S$ satisfying $\text{dist}(x, y) \leq cd_*(x)$. In particular, if $d_*(x) = 0$, a multiplicative guarantee requires an exact match.

The output is a point identifier; copying the entire reported vector, if required, adds its output length.

A direct scan evaluates n distances. Its cost is $O(nT_{\text{dist}})$, where T_{dist} is the cost of one distance computation; for dense Euclidean or unpacked Hamming vectors, $T_{\text{dist}} = O(d)$. In low dimension, geometric structures such as k -d trees (recursive coordinate partitions) and Voronoi diagrams (regions with the same nearest point) offer alternative organizations of the search. High-dimensional search motivates a different tradeoff: allow approximation and randomization instead of insisting on exact answers. The resulting tradeoffs must specify the computational model and the roles of both n and d .

Definition 4.1 (Approximate near neighbor). Fix $c > 1$ and $r > 0$ before preprocessing. A (c, r) -approximate near-neighbor query has the following requirements:

$$\begin{aligned} d_*(x) \leq r & & : \text{ return some } y \in S \text{ with } \text{dist}(x, y) \leq cr; \\ d_*(x) > cr & & : \text{ return } \perp; \\ r < d_*(x) \leq cr & & : \text{ either outcome is allowed.} \end{aligned}$$

Here $\perp$ means that no point is reported. Our implementations verify reported points, so a non- $\perp$ output always has distance at most cr , even in the gap between the two radii.

The distinction between *near* and *nearest* is important: a near-neighbor structure is built for one prescribed radius, whereas the nearest distance is unknown. Section 7 combines several radius-specific structures using an explicit search invariant.

For a randomized structure, success probability is measured over preprocessing and any query randomness, for every fixed dataset and fixed query independent of those random choices. We will obtain one-sided error: verification prevents an invalid returned point, but a query with a true r -neighbor may incorrectly return $\perp$. The gap region remains unconstrained except for verification.

4.2 What JL gives for a fixed Euclidean query

Suppose a linear map A preserves squared distances from a fixed query x to all data points within factors $1 \pm \varepsilon$. This follows by applying the one-vector bound to the n vectors $x - y_i$, with sufficient dimension

$$k \geq \frac{8}{\varepsilon^2} \ln \frac{2n}{\delta}. \quad (17)$$

The query need not be supplied at preprocessing time; it must be fixed independently of A for this probabilistic assertion.

Proposition 4.2 (Transfer of an approximate nearest neighbor). *If $y \in S$ is a b -approximate nearest neighbor of Ax among the points Ay_i in Euclidean distance, then, on the distance-preservation event,*

$$\|x - y\|_2 \leq b \sqrt{\frac{1+\varepsilon}{1-\varepsilon}} d_*(x). \quad (18)$$

Proof. Let y_* be a true nearest neighbor in the original space. The lower distortion inequality for y , the projected approximation guarantee, and the upper distortion inequality for y_* give

$$\sqrt{1-\varepsilon} \|x - y\|_2 \leq \|Ax - Ay\|_2 \leq b \|Ax - Ay_*\|_2 \leq b \sqrt{1+\varepsilon} d_*(x).$$

Divide by $\sqrt{1-\varepsilon} > 0$. □

For exact search after projection, $b = 1$. To obtain target approximation $c > 1$, it suffices to choose $\varepsilon \leq (c^2 - 1)/(c^2 + 1)$. Preserving distances only *within the dataset* is insufficient: the query-to-data vectors must also satisfy the estimate. For any predetermined finite collection of Q queries, a union bound over nQ differences supplies the corresponding simultaneous guarantee. This reasoning does not cover arbitrary queries chosen after observing the map.

4.3 Two routes beyond a projected scan

Even with Proposition 4.2, a scan in the projected space still compares n candidates. We now develop two search structures in Hamming space. The first compresses to $k = O(\log n)$ bits and precomputes an answer for *every possible compressed query*, paying 2^k space. The second stores only buckets reached by the data and searches several randomized tables. It pays for repeated hashing and candidate verification, rather than enumerating the entire compressed universe.

These constructions deliberately solve a weaker geometric task than preserving every distance. They need only separate distances at most r from distances greater than cr . The annulus between these thresholds provides room for concentration or collision amplification to work.

5 A Hamming sketch and a polynomial-space lookup table

Fix a nonempty dataset $S \subseteq \{0,1\}^d$, a radius $r \geq 1$, and an approximation factor $c > 1$. If $cr \geq d$, any data point is an acceptable answer. We concentrate on the nontrivial case $cr < d$. For $0 \leq r < 1$, a true r -neighbor is an exact match, which can be tested separately.

The goal is to compress a query to a short Boolean signature that distinguishes near and far data points. The compressed Hamming distance will not be a scaled copy of the original distance. Instead, each output bit supplies an independent test whose expectation changes by a definite amount across the gap from r to cr .

5.1 The parity of a random subset

Choose a $k \times d$ matrix A whose entries are independent $\text{Bern}(p)$ variables, where $0 < p < 1/2$. All matrix-vector products in this section are over $\mathbb{F}_2$:

$$(Ax)_i = \left(\sum_{j=1}^d A_{ij}x_j \right) \bmod 2. \quad (19)$$

For two fixed points x, y , put $t = \text{Ham}(x, y)$. Only their differing coordinates contribute to $A(x - y)$ over $\mathbb{F}_2$.

Lemma 5.1 (One-row disagreement). *For every row i ,*

$$\mathbb{P}((Ax)_i \neq (Ay)_i) = q_p(t) := \frac{1 - (1 - 2p)^t}{2}. \quad (20)$$

The disagreement indicators from different rows are independent.

Proof. Let $B_1, \dots, B_t$ be the independent matrix entries at the differing coordinates. Their sum is odd precisely when the two output bits disagree. If q is that probability, then

$$1 - 2q = \mathbb{E}(-1)^{B_1 + \dots + B_t} = \prod_{j=1}^t \mathbb{E}(-1)^{B_j} = (1 - 2p)^t.$$

Solve for q . Different rows use disjoint sets of independent entries. □

There is also a useful sampling interpretation. Include each coordinate in a set D independently with probability $2p$, and on D choose independent fair bits as the row entries; set the remaining entries to zero. This produces exactly independent $\text{Bern}(p)$ entries. If D misses every differing coordinate, the outputs agree. Otherwise their parity difference is a fair bit. The probability of a miss is $(1 - 2p)^t$, which gives (20) again.

5.2 Tune the sketch to one radius

Choose

$$p = \frac{1 - 2^{-1/r}}{2}, \quad \gamma_c = \frac{1}{8} - 2^{-(c+2)}, \quad \tau_c = \frac{3}{8} - 2^{-(c+2)} = \frac{1}{4} + \gamma_c. \quad (21)$$

Because $c > 1$, $\gamma_c > 0$. Equation (20) now reads $q_p(t) = (1 - 2^{-t/r})/2$, a monotone function of t . Therefore

$$t \leq r \implies q_p(t) \leq 1/4, \quad (22)$$

$$t > cr \implies q_p(t) > 1/2 - 2^{-(c+1)} = 1/4 + 2\gamma_c. \quad (23)$$

The threshold τ_c is the midpoint between these two expectation bounds. For example, $c = 2$ gives $\gamma_c = 1/16$ and $\tau_c = 5/16$: the near mean is at most $1/4$, while the far mean is greater than $3/8$.

Let $Z = \text{Ham}(Ax, Ay)$. It is a sum of k independent Bernoulli variables. Hoeffding's inequality, recalled and proved in Appendix B, gives

$$\text{Ham}(x, y) \leq r \implies \mathbb{P}(Z > \lfloor k\tau_c \rfloor) \leq e^{-2\gamma_c^2 k}, \quad (24)$$

$$\text{Ham}(x, y) > cr \implies \mathbb{P}(Z \leq \lfloor k\tau_c \rfloor) \leq e^{-2\gamma_c^2 k}. \quad (25)$$

Indeed, in the first event the integer Z is greater than $k\tau_c \geq \mathbb{E}Z + \gamma_c k$. In the second case $Z \leq k\tau_c < \mathbb{E}Z - \gamma_c k$. Both deviations have size at least $\gamma_c k$.

5.3 Precompute every possible compressed query

Set $s = \lfloor k\tau_c \rfloor$ and $z_i = Ay_i$. For each $u \in \{0, 1\}^k$, define the list

$$B_s(u) = \{i \in [n] : \text{Ham}(u, z_i) \leq s\}. \quad (26)$$

There are 2^k possible signatures, regardless of the original dimension. The list stores identifiers of original points, not copies of their d -bit vectors.

Hamming near neighbor by a complete signature table

Preprocessing. Choose A and compute every $z_i = Ay_i$. Enumerate all $u \in \{0, 1\}^k$ and form $B_s(u)$. Store the dataset and the table.

Query x . Compute $u = Ax$. If $B_s(u)$ is empty, return $\perp$. Otherwise take any identifier $i \in B_s(u)$, compute $\text{Ham}(x, y_i)$, and return y_i if this distance is at most cr ; return $\perp$ otherwise.

Theorem 5.2 (Complete-table search). *For every fixed S of size $n \geq 1$, every fixed query x , and $0 < \delta < 1$, choose*

$$k = \left\lceil \frac{\ln(2n/\delta)}{2\gamma_c^2} \right\rceil. \quad (27)$$

The algorithm solves the (c, r) -near-neighbor query with probability at least $1 - \delta$. Every reported point is valid, even outside the success event.

Proof. For each data point at distance at most r , regard exclusion from the queried list as a bad event. For each data point at distance greater than cr , regard inclusion as a bad event. There is at most one such event per point. Equations (24)–(25) and a union bound give probability at most $ne^{-2\gamma_c^2 k} \leq \delta/2$ that any bad event occurs.

On the complementary event, every true r -neighbor lies in $B_s(Ax)$, and every identifier in that list has original distance at most cr . Thus, when a true neighbor exists, the list is nonempty and any chosen identifier passes verification. If all points are farther than cr , the list is empty. Verification prevents invalid outputs under every outcome of A . $\square$

The union bound involves the n distances from *one fixed query*, not all 2^d queries. The table has an address for every compressed query, but that fact alone does not strengthen the probability guarantee.

5.4 Space, preprocessing, and a smaller table

Computing all signatures costs $O(nkd)$ bit operations. A direct enumeration of all lists takes $O(nk2^k)$ additional bit comparisons. The lists contain at most $n2^k$ identifiers in total. Besides their identifier and pointer storage, retain the nd input bits and the kd matrix bits. A query takes $O(kd + d)$ bit operations plus a constant number of table accesses in a RAM whose words can address the 2^k cells and the dataset. Computing the k -bit address is included in the transform cost.

In fact, it suffices to store *one representative identifier* from each nonempty $B_s(u)$, or $\perp$ if the list is empty. The proof of Theorem 5.2 does not need the other members: on its success event, every representative in the queried cell is valid. This reduces the stored table to 2^k entries. Its bit storage, apart from address conventions, is

$$O(nd + kd + 2^k \lceil \log_2(n+1) \rceil). \quad (28)$$

Verification is particularly useful with this smaller table: an unlucky far representative can cause a false negative, never a false positive.

For fixed $c > 1$ and $\delta = n^{-O(1)}$, we have $k = O_c(\log n)$ and $2^k = n^{O_c(1)}$. Thus polynomial space buys $O_c(d \log n)$ query time. The exponent in the space bound can be large. As $c \downarrow 1$, $\gamma_c = \Theta(c - 1)$ and $k = O((c - 1)^{-2} \log(n/\delta))$. This is an existence-oriented data structure, not automatically a practical small index. Randomized Hamming search of this kind is closely related to the work of Kushilevitz, Ostrovsky, and Rabani [6].

Sampling probabilities and finite descriptions. The exact formulation permits independent $\text{Bern}(p)$ sampling with the specified parameter. A finite-bit variant can round p to a dyadic p' with $|p' - p| \leq \gamma_c/(4d)$, and round τ_c to within $\gamma_c/4$. Coupling the d row entries bounds the change in any row-disagreement probability by $d|p' - p| \leq \gamma_c/4$. The separation margin remains at least $\gamma_c/2$. Increasing k by a factor of four therefore preserves the preceding failure bound. For fixed computable c and an integer radius, $O(\log(d/\gamma_c))$ bits of bias precision suffice; independent fair-bit blocks implement the rounded Bernoulli trials. No continuous random matrix is necessary for this construction.

A radius-sensitive sketch is not a Hamming JL embedding

As t/r grows, $q_p(t)$ approaches $1/2$. Widely different large Hamming distances can therefore have nearly identical compressed expectations. This saturation is harmless for a fixed-radius near-neighbor query: the only required distinction is between $t \leq r$ and $t > cr$.

6 Locality-sensitive hashing: search only a few buckets

The complete table pays for all 2^k compressed signatures. Locality-sensitive hashing instead stores only buckets containing data and uses several independent tables. Its analysis must balance two competing events: a true near point must collide somewhere, while far points must not generate too much work.

6.1 A collision gap and its amplification

Definition 6.1 (Locality-sensitive hashing). A distribution over maps $h : X \rightarrow U$ is an (r, cr, p, q) locality-sensitive hash family if $c > 1$, $r > 0$, $0 < q < p < 1$, and, for every fixed $x, y \in X$,

$$\text{dist}(x, y) \leq r \implies \mathbb{P}(h(x) = h(y)) \geq p, \quad (29)$$

$$\text{dist}(x, y) > cr \implies \mathbb{P}(h(x) = h(y)) \leq q. \quad (30)$$

Unlike ordinary hashing, collisions between similar objects are desirable. The family is a distribution: the probability is over a sampled function, not over a distributional assumption on the input points. LSH was introduced as an algorithmic approach to high-dimensional approximate search by Indyk and Motwani [4].

Choose K independent hashes and concatenate their values:

$$g(x) = (h_1(x), \dots, h_K(x)) \in U^K. \quad (31)$$

For a fixed pair, equality of the full keys requires equality in every component. Independence implies that g is (r, cr, p^K, q^K) -sensitive. This *tensorization*, or AND amplification, lowers the far-collision probability, but it also lowers the near-collision probability.

For $n \geq 2$, set

$$K = \left\lceil \frac{\ln n}{\ln(1/q)} \right\rceil, \quad p_* = p^K, \quad L = \lceil 1/p_* \rceil. \quad (32)$$

Then $q^K \leq 1/n$. A single table has at most one expected far-point occurrence in the query bucket. A near point, however, appears with probability only p_* . Using L independent tables supplies an OR amplification: it is enough that a near point collides in one of them.

6.2 The data structure and the candidate budget

In table j , store a dictionary from each attained key $g_j(y_i)$ to the list of all point identifiers having that key. Equal keys share one list; they must not overwrite one another. The original points are stored once and accessed for verification. The dictionaries are exact; Section 6.4 gives an implementation and accounts for long keys.

LSH near-neighbor search with a cutoff

Preprocessing. Independently generate L concatenated hashes $g_1, \dots, g_L$, each using K independent base hashes. Build the L dictionaries of buckets.

Query x . Compute $g_j(x)$ and retrieve its bucket in each table. Traverse these lists in a fixed order, examining at most $10L$ candidate occurrences in total. For each occurrence of y_i , compute $\text{dist}(x, y_i)$. Return the first point whose distance is at most cr . If the lists end or the budget is exhausted without such a point, return $\perp$.

A point stored in several retrieved buckets may be examined several times. Counting occurrences makes both the work bound and the probability analysis explicit; no duplicate-removal data structure is necessary.

Theorem 6.2 (The basic LSH guarantee). *For every fixed dataset of $n \geq 2$ points and fixed query independent of the random hashes, the above structure solves (c, r) -near neighbor with failure probability at most*

$$e^{-1} + \frac{1}{10} < \frac{1}{2}. \quad (33)$$

It never returns a point farther than cr . The number of examined candidate occurrences is deterministically at most $10L$.

Proof. Suppose a point $y_* \in S$ satisfies $\text{dist}(x, y_*) \leq r$, and fix one such point independently of the hashes. Its probability of missing all L tables is at most

$$(1 - p_*)^L \leq e^{-p_* L} \leq e^{-1}.$$

Let F count *far occurrences* in all retrieved buckets, namely pairs (j, i) such that $\text{dist}(x, y_i) > cr$ and $g_j(x) = g_j(y_i)$. By linearity of expectation,

$$\mathbb{E}F = \sum_{j=1}^L \sum_{i: \text{dist}(x, y_i) > cr} \mathbb{P}(g_j(x) = g_j(y_i)) \leq Lnq^K \leq L. \quad (34)$$

No independence between different data points is used. Markov gives $\mathbb{P}(F \geq 10L) \leq 1/10$.

If y_* collides somewhere and $F < 10L$, then the traversal reaches an acceptable point before exhausting the budget: every occurrence preceding the first acceptable one is far, and there are fewer than $10L$ such occurrences even in the full collection. Thus failure is contained in the union of the two events just bounded. They need not be independent. If all data points are farther than cr , verification forces output $\perp$. In the gap region either outcome is allowed. $\square$

A useful feature of this proof is that the candidate count is controlled without bounding the entire bucket size. A bucket may contain many acceptable points; the query stops at the first one. Only the work caused by far points can delay success.

6.3 The exponent and the actual resource bounds

Define

$$\rho = \frac{\ln(1/p)}{\ln(1/q)} = \frac{\ln p}{\ln q} \in (0, 1). \quad (35)$$

Rounding K in (32) gives the useful precise bounds

$$p n^{-\rho} \leq p_* \leq n^{-\rho}, \quad n^\rho \leq L \leq p^{-1} n^\rho + 1. \quad (36)$$

Thus $L = O(n^\rho)$ when p is bounded below by a positive constant. The tradeoff is L bucket lookups and $O(L)$ distance verifications, with nL point occurrences stored across the tables.

For implementation, suppose each base hash value is encoded in b bits, a base hash takes time T_h , and a distance takes time T_{dist} . Exact compressed-trie dictionaries, described in Section 6.4, give the query bound

$$O(L(KT_h + Kb + T_{\text{dist}})). \quad (37)$$

They use $O(nL)$ machine-word records, *in addition to* the original dataset and the descriptions of the LK sampled base hashes. Long full keys need not be stored permanently: a leaf stores a representative data-point identifier from which its key can be recomputed for an equality check. A simpler implementation that retains every full key uses an additional $O(nLKb)$ bits. Word sizes must suffice for point identifiers, addresses, and positions in a Kb -bit key.

For example, one deterministic preprocessing implementation computes all keys, sorts each table's keys, groups equal keys, and builds its compressed trie. Besides hash generation, it takes $O(nLK(T_h + b \log n))$ time; temporary key storage can be reused from table to table. The resource accounting distinguishes storage of candidate references from storage of vectors and hash descriptions.

Under unit-cost hashes, constant-length hash values, unit-cost distances, and fixed p, q , equation (37) becomes the familiar $O(n^\rho \log n)$ query time, with $O(n^{1+\rho})$ index records. Those assumptions are not automatic for high-dimensional data. The Hamming example below makes the dimensional costs explicit.

6.4 Exact bucket lookup with long signatures

The key $g_j(y)$ has $B = Kb$ bits. A binary trie stores keys by branching on their successive bits. An ordinary trie for m distinct keys uses at most $1 + mB$ nodes; following the B query bits finds the matching bucket or detects absence. This also gives the exact-membership structure for Hamming vectors used in Section 7.

To reduce permanent index space, suppress every nonbranching internal path. Each remaining internal node stores the next bit position at which the keys below it differ, together with two child pointers. A leaf stores a representative point identifier and the bucket list. Along any root-to-leaf path, tested positions strictly increase, so there are at most B bit tests. A binary tree with m leaves and two children at every internal node has $2m - 1$ nodes. Thus each table uses $O(m)$ tree records and n list records.

Skipping common-prefix positions means that reaching a leaf does not alone establish equality. Recompute $g_j(y)$ for the leaf's representative y , compare the entire key with $g_j(x)$, and return the bucket only if they agree. A matching stored key always follows its own path; a nonmatching key is rejected by this final comparison. Lookup therefore costs $O(B + KT_h)$ after the query key has been computed, while the full stored keys can be discarded. This construction is an exact dictionary, independent of the probabilistic assumptions on LSH collisions.

When an entire key fits in a constant number of machine words, FKS perfect hashing [5] is another exact dictionary option, with constant-time bucket access. Hash evaluation and distance verification remain separate costs. The compressed-trie construction above avoids assuming constant-time access to an arbitrarily long key.

Higher confidence. Build $a = \lceil \log_2(1/\delta) \rceil$ independent copies of the entire structure. Query them until a verified point is returned, or all copies fail. For a fixed query with a true near point, all copies fail with probability at most $2^{-a} \leq \delta$. Storage, preprocessing, and the worst-case query work grow by a factor a . Since each output is verified, taking the first valid answer preserves one-sided error.

6.5 Coordinate sampling in Hamming space

Assume $1 \leq r < d/c$. Choose J uniformly from $[d]$ and let $h(x) = x_J$. If $t = \text{Ham}(x, y)$, then

$$\mathbb{P}(h(x) = h(y)) = 1 - t/d. \quad (38)$$

Thus this is an (r, cr, p, q) family with $p = 1 - r/d$ and $q = 1 - cr/d$. Its exponent satisfies

$$\rho = \frac{\ln(1 - r/d)}{\ln(1 - cr/d)} \leq \frac{1}{c}. \quad (39)$$

To prove this, put $u = r/d \in (0, 1/c)$. Then

$$-\ln(1 - cu) = c \int_0^u \frac{dt}{1 - ct} \geq c \int_0^u \frac{dt}{1 - t} = -c \ln(1 - u).$$

Dividing by the positive denominator gives (39).

Each base hash reads one bit, so $T_h = O(1)$ and $b = 1$. A full Hamming-distance verification costs $O(d)$ bit operations. Crucially, the concatenation length is

$$K = \left\lceil \frac{\ln n}{-\ln(1 - cr/d)} \right\rceil \leq 1 + \frac{d}{cr} \ln n. \quad (40)$$

It need not be $O(\log n)$ uniformly in r/d . Since $p > 1 - 1/c$, $L = O_c(n^{1/c})$, and the resulting query bound is

$$O_c \left(n^{1/c} \left(d + \frac{d}{r} \ln n + 1 \right) \right). \quad (41)$$

The index has $O(nL)$ records; the hash descriptions consist of LK coordinate indices; the dataset consists of nd bits. When $cr \geq d$, return any data point instead. Radii below one can be handled by exact membership. Packed-bit distance computation can improve T_{dist} in a specified word model, but it does not justify omitting that cost without stating the model.

AND reduces unwanted collisions; OR restores the desired one

Concatenation makes $q^K \leq 1/n$, controlling expected far work per table. Independent tables make a fixed near point appear somewhere with constant probability. The cutoff converts an expectation bound into a bounded-work Monte Carlo algorithm, and exact distance checks prevent false positives.

7 From near neighbor to nearest neighbor

A fixed-radius structure answers a threshold question. To approximate the nearest distance, build structures at geometric scales. Two details require care: the range of radii must cover the relevant situation, and approximate answers in the gap region need not be monotone.

7.1 A reduction with a known positive distance range

Suppose a query is known to satisfy $0 < a \leq d_*(x) \leq b$. Choose a near-neighbor factor $c_0 > 1$ and a scale ratio $\eta > 1$. Build (c_0, r_j) structures at

$$r_j = a\eta^j, \quad 0 \leq j < J, \quad J = 1 + \lceil \log_\eta(b/a) \rceil. \quad (42)$$

A YES answer means that a verified point at distance at most $c_0 r_j$ was returned. A NO answer means $\perp$. For now assume every radius-specific query satisfies its promise.

Search for a YES–NO boundary

Query r_0 . If the answer is YES, return its point. Otherwise query r_{J-1} , which must answer YES because $r_{J-1} \geq b \geq d_*(x)$.

Maintain indices $\ell < h$ such that r_ℓ answered NO and r_h answered YES, retaining the point returned at h . While $h - \ell > 1$, query $j = \lfloor (\ell + h)/2 \rfloor$. On NO set $\ell = j$; on YES set $h = j$ and retain the new returned point. At adjacency, return the point retained at h .

When $J = 1$, the first radius already brackets the entire distance range, so only the first query is needed.

Proposition 7.1 (Approximation from a bracket). *On the event that all the radius queries are correct, this procedure returns a $c_0\eta$ -approximate nearest neighbor using $O(1 + \log J)$ radius queries.*

Proof. If the first answer is YES, its distance is at most $c_0 a \leq c_0 d_*(x)$. Otherwise the procedure keeps an actual NO at ℓ and an actual YES at h , regardless of answers at other radii. A correct NO implies $d_*(x) > r_\ell$, since a near point would force a YES. A verified YES supplies a point at distance at most $c_0 r_h$. At termination, $r_h = \eta r_\ell$, so its distance is at most

$$c_0 r_h = c_0 \eta r_\ell < c_0 \eta d_*(x). \quad (43)$$

Each step at least halves the number of intervening indices. □

A YES at a smaller radius need not imply a YES at every larger radius: both calls may lie in their permitted gap regions. The proof does not assume such monotonicity. It maintains two facts about the *answers actually received*.

For a target factor $C > 1$, choose $c_0 = \eta = \sqrt{C}$. The radius-specific approximation and the spacing of radii both contribute to the final factor. Using factor C for both would in general give C^2 , not C .

7.2 Covering arbitrary metric queries with endpoint guards

Pairwise distances inside the dataset do not by themselves bound the query's nearest distance. For instance, the dataset $\{0, 1\} \subset \mathbb{R}$ has minimum and maximum interpoint distance one, but a query can be arbitrarily close to zero or arbitrarily far from both points. A correct data-dependent reduction must handle these extremes.

Assume the dataset has $n \geq 2$ distinct points and define

$$D_{\min} = \min_{i < j} \text{dist}(y_i, y_j) > 0, \quad D_{\max} = \max_{i < j} \text{dist}(y_i, y_j), \quad R = D_{\max} / D_{\min}.$$

For target $C > 1$, put $c_0 = \eta = \sqrt{C}$ and choose

$$a = \frac{D_{\min}}{2(c_0 + 1)}, \quad b = \max \left\{ a, \frac{D_{\max}}{C - 1} \right\}. \quad (44)$$

Build the grid (42). Query the smallest radius first. If it returns a point, return that point. If it answers NO, query the largest radius. A NO there now means *return any data point*; a YES initializes the same boundary search as before.

Theorem 7.2 (Guarded near-to-nearest reduction). *On correctness of the radius-specific calls, the guarded procedure returns a C -approximate nearest neighbor for every query location, including queries with nearest distance zero. The number of stored scales is*

$$J = 1 + \left\lceil \log_{\sqrt{C}} \max \left\{ 1, \frac{2R}{\sqrt{C} - 1} \right\} \right\rceil. \quad (45)$$

For fixed $C > 1$, this is $O_C(1 + \log R)$.

Proof. If $d_*(x) \leq a$, fix a nearest point y_* . Any different data point y satisfies

$$\text{dist}(x, y) \geq \text{dist}(y, y_*) - \text{dist}(x, y_*) \geq D_{\min} - a > c_0 a.$$

The smallest-radius call must therefore return y_* itself. This also handles $d_*(x) = 0$. If $d_*(x) > a$ but the smallest-radius call returns a point, that point has distance at most $c_0 a < c_0 d_*(x)$.

After a NO at the smallest radius, $d_*(x) > a$. If the largest-radius call answers NO, then $d_*(x) > r_{J-1} \geq D_{\max} / (C - 1)$. For any data point y , the triangle inequality gives

$$\text{dist}(x, y) \leq d_*(x) + D_{\max} < C d_*(x).$$

If the largest-radius answer is YES, Proposition 7.1's bracket argument applies; it only needs the NO and YES endpoints, not an a priori upper bound on d_* . Finally, $b/a = \max\{1, 2R/(\sqrt{C} - 1)\}$, giving (45). $\square$

A singleton dataset is answered by its only point. Duplicate points can be merged before computing $D_{\min}$. Computing exact dataset extrema by a direct method costs $O(n^2 T_{\text{dist}})$ preprocessing time; that cost must be included if these parameters are not already available.

7.3 Failure probabilities and the Hamming specialization

For one fixed query, let each of the J radius structures fail with probability at most δ/J . A union bound over *all stored radii* ensures that every answer needed by the adaptive boundary search satisfies its promise with probability at least $1 - \delta$. This argument does not condition a per-query guarantee on an adaptively chosen scale. It is sufficient even without independence between scales.

If each radius structure uses space s and query time t at that confidence level, the total is $O(Js)$ space and $O(t(1 + \log J))$ query time, plus shared dataset storage and any endpoint setup. When confidence is obtained by independent LSH copies, s and t include the factor $O(\log(J/\delta))$; it is not free.

In Hamming space there is a simpler alternative that avoids computing $D_{\min}$ and $D_{\max}$. First test exact membership in S . A binary trie of the stored d -bit vectors uses $O(nd)$ nodes and answers this test in $O(d)$ time; Section 6.4 explains the basic trie operation. If the query is not in S , then $1 \leq d_*(x) \leq d$. Use (42) with $a = 1$, $b = d$, and $c_0 = \eta = \sqrt{C}$. The result is C -approximate nearest-neighbor search with $J = 1 + \lceil \log_{\sqrt{C}} d \rceil$ scales. All zero-distance and radius-endpoint cases are then explicit.

8 Putting the methods in perspective

8.1 Three different meanings of a short representation

A random representation should preserve the information needed by the query, not necessarily every feature of the input. The three constructions in this lecture make different promises.

Construction	What is preserved	How it helps search
Euclidean JL	All pairwise distances in a fixed finite set, approximately.	Reduces the cost of a distance computation; an index is still needed to avoid a scan.
Hamming parity map	A gap between near and far distances at one prescribed scale.	A small Boolean universe permits a complete answer table.
Locality-sensitive hashing	A gap between near- and far-collision probabilities.	Concatenation, repetition, and a cutoff limit candidate verification.

The parity map is not an isometry of the Hamming cube, and an LSH collision guarantee is not a JL distance guarantee. A map can be useful for search without preserving all distances. Conversely, a distance-preserving map need not supply a sublinear-time search algorithm by itself.

8.2 How much can Euclidean dimension be reduced?

For a fixed distortion bounded away from one-sided collapse, logarithmic dimension is unavoidable in the worst case. The following elementary argument explains the dependence on the number of points.

Proposition 8.1 (A packing lower bound). *Let $0 < \varepsilon < 1$. If the n standard basis vectors in $\mathbb{R}^n$ are mapped into $\mathbb{R}^k$ while satisfying (2), then*

$$k \geq \frac{\ln n}{\ln \left(1 + 2\sqrt{(1+\varepsilon)/(1-\varepsilon)} \right)}. \quad (46)$$

Proof. Distinct basis vectors have distance $\sqrt{2}$. Their images have mutual distances between $a = \sqrt{2(1-\varepsilon)}$ and $b = \sqrt{2(1+\varepsilon)}$. The open radius- $a/2$ balls about the images are pairwise disjoint. All lie in the radius- $(b+a/2)$ ball about the first image. Comparing k -dimensional volumes gives $n(a/2)^k \leq (b+a/2)^k$. Take logarithms. If $k = 0$ and $n \geq 2$, the lower distance bound is already impossible. $\square$

For any fixed $\varepsilon < 1$, this is $\Omega_\varepsilon(\log n)$, even for nonlinear embeddings. It does not establish the sharp dependence on ε . Stronger lower bounds tracking that dependence are developed by Larsen and Nelson [7]; their parameter regimes matter, since retaining the original d coordinates is always possible.

Sparse and fast transforms. The dense Gaussian and sign maps cost $O(kd)$ per vector. Two further goals are to reduce the number of nonzero entries per column, or to give the matrix a structure supporting fast multiplication. Sparse JL transforms [8] and fast JL transforms [9] address these goals. One useful idea for the latter is to apply random signs followed by a normalized Hadamard transform, spreading a vector's energy among coordinates before sampling them. For a power-of-two dimension, the Hadamard transform can be evaluated in $O(d \log d)$ additions and subtractions by its recursive sum-and-difference construction. The sampling stage still needs its own concentration argument; arbitrary coordinate deletion is not justified merely by the existence of JL. These refinements concern evaluation cost, not a different meaning of distance preservation.

8.3 A research connection: sparse tags in an olfactory circuit

Dasgupta, Stevens, and Navlakha [10] investigated similarity search through the fruit fly's olfactory circuit. Their computational description has three stages: normalize the input response, expand it using sparse binary connections, and retain only the strongest responses. In the circuit studied, roughly 50 input projection neurons connect to about 2000 Kenyon cells, each combining about six inputs; approximately the highest-responding 5% of cells form the sparse tag.

The important contrast is with dimensional *reduction*: the intermediate representation expands from about 50 to 2000 coordinates, then sparsifies. The paper relates the similarity of these tags to similarity search and evaluates algorithms inspired by this organization. That connection does not by itself prove the particular (r, cr, p, q) collision inequalities of Definition 6.1, nor does it make the circuit a Euclidean JL map. It illustrates a broader design principle: a representation may use more intermediate coordinates while leaving only a small number active or stored.

What to check in a randomized search guarantee

State the metric and query problem, the randomness, the covered pairs or queries, and the cost of evaluating the representation. Separate candidate retrieval from verification. Low dimension, a small collision exponent, and low total query cost are distinct achievements.

9 Exercises

The exercises follow the progression from preserving distances to designing a complete search structure. All probability statements should specify which inputs are fixed before sampling the random map.

Euclidean dimension reduction

1. **Squared distances and approximation factors.** Starting from (2), derive the corresponding bounds for distances. An algorithm returns a b -approximate nearest neighbor after projection. Determine the largest allowed ε that guarantees an original-space factor $C > b$. What changes when the nearest distance is zero?
2. **Data points versus queries.** Construct a dataset and a linear map that preserve all dataset distances exactly, but fail to preserve some query-to-data distance. Then explain why a map from $\mathbb{R}^d$ to $\mathbb{R}^k$ with $k < d$ cannot preserve every distance in the whole space with positive lower distortion.
Hint. A projection can preserve a line containing the dataset while erasing an orthogonal direction.
3. **Choose the failure budget.** For n fixed data points and Q fixed queries, give a sufficient Gaussian target dimension that preserves all query-to-data squared distances within $1 \pm \varepsilon$ with total failure probability at most δ . Compare this with preserving all pairs in the union of the data and queries.
4. **The random-subspace calculation.** Verify both MGF factors at the optimizing t in Lemma 3.1. Substitute $\beta = 1 - \varepsilon$ and $1 + \varepsilon$, handling separately the case $(1 + \varepsilon)k \geq d$. Why is treating the numerator and denominator in (11) as independent invalid?
5. **Signs are not Gaussian.** Prove (15) directly. For $u = e_1$, compute $\|Au\|_2^2$ exactly under the sign transform. Explain why the positive-MGF comparison alone does not prove the lower-tail estimate.

Radius-sensitive compression

6. **Parity and a concrete gap.** Derive (20) by conditioning on the random support, rather than by using $(-1)^Z$. For $c = 2$, compute γ_c , τ_c , and a sufficient integer k for failure probability $1/n$. Explain the role of the floor in $s = \lfloor k\tau_c \rfloor$.
7. **One representative per signature.** Prove that storing a single identifier in each nonempty table cell is enough on the event used in Theorem 5.2. Give a reason that this does not imply every query in the Boolean cube succeeds simultaneously. What original-space verification guarantees even off the good event?

Hashing and candidate verification

8. **Vary the candidate budget.** Replace $L = \lceil 1/p_* \rceil$ by $L = \lceil a/p_* \rceil$ and the cutoff $10L$ by BL , where $a > 0$ and $B > 1$. Prove a failure bound $e^{-a} + 1/B$, using an integer cutoff if necessary. Distinguish the random collision event from the deterministic distance verification.

9. **Occurrences versus distinct points.** A far point can appear in many tables. Explain why a bound on the number of distinct far points is not by itself a bound on list-traversal work. Give an implementation that counts occurrences, and an alternative that avoids repeated distance evaluations while still accounting for every traversed list record.
10. **The dependence on the radius.** Prove (39) and derive K when $r/d \rightarrow 0$. Compare $r = 1$ with $r = \Theta(d)$ for fixed c . State the resulting query costs when one full Hamming-distance computation costs d bit operations.

Combining radius-specific structures

11. **Searching without monotone answers.** Describe a permissible sequence of YES/NO answers in the gap region that is not monotone. Prove that the boundary search still achieves factor $c_0\eta$. Identify the two invariants it actually uses.
12. **The smallest and largest scales.** For the dataset $\{0, 1\} \subset \mathbb{R}$ and target $C = 4$, compute the guarded radii. Check separately a query equal to a data point, one very close to a data point, and one far outside the dataset. Then explain why allocating failure probability δ/J to each stored scale protects the adaptively chosen sequence of radius queries.

A Gaussian and Euclidean tools

A.1 Normalization, moments, and linear combinations

The Gaussian integral is

$$\int_{-\infty}^{\infty} e^{-x^2/2} dx = \sqrt{2\pi}.$$

One verification squares the integral, changes to polar coordinates in $\mathbb{R}^2$, and obtains $\int_0^{2\pi} \int_0^{\infty} e^{-r^2/2} r dr d\theta = 2\pi$. All integrands are nonnegative, so the iterated integrations are justified. For a standard normal Z , symmetry gives $\mathbb{E}Z = 0$. Since the density $\phi(z) = (2\pi)^{-1/2}e^{-z^2/2}$ satisfies $\phi'(z) = -z\phi(z)$, integration by parts gives

$$\mathbb{E}Z^{2m} = (2m-1)\mathbb{E}Z^{2m-2} \quad (m \geq 1).$$

The boundary term vanishes because a polynomial times $e^{-z^2/2}$ tends to zero. Thus $\mathbb{E}Z^2 = 1$, $\mathbb{E}Z^4 = 3$, and $\text{Var}(Z^2) = 2$.

Completing the square proves, for every real a ,

$$\mathbb{E}e^{aZ} = \frac{1}{\sqrt{2\pi}} \int_{\mathbb{R}} e^{az - z^2/2} dz = e^{a^2/2}. \quad (47)$$

If $G_1, \dots, G_d$ are independent standard normals, their joint density is

$$f_G(g) = (2\pi)^{-d/2} e^{-\|g\|_2^2/2}. \quad (48)$$

For any orthogonal matrix O , $\|Og\|_2 = \|g\|_2$ and $|\det O| = 1$. The change-of-variables formula therefore shows $OG \stackrel{d}{=} G$. In particular, for a nonzero $v \in \mathbb{R}^d$, choose an orthogonal matrix with first row $v/\|v\|_2$. Its first coordinate gives

$$\langle v, G \rangle \stackrel{d}{=} \|v\|_2 G_1.$$

Adding means and scaling independent standard normals proves the linear-combination rule (5). Different rows of a Gaussian projection give independent coordinates because they use disjoint independent entries.

A.2 Uniform directions and random frames

The event $G = 0$ has probability zero. In polar coordinates the density (48) depends only on the radius, so $G/\|G\|_2$ is uniform with respect to surface area on the unit sphere; the direction is also independent of the radius. In dimension one the two directions have equal probability.

Here is a direct justification for the symmetry used in Section 3.1. Let Q be a random orthonormal frame whose distribution is invariant under rotations of $\mathbb{R}^d$. The distribution of $\|Qu\|_2$ is then the same for every fixed unit vector u . It equals the distribution obtained by first taking an independent uniform unit vector U and then evaluating $\|QU\|_2$. Conditional on any frame Q , rotating coordinates identifies its span with the first k coordinate directions, and U remains uniform. Therefore the distribution is that of $\sqrt{U_1^2 + \dots + U_k^2}$. Substituting $U = G/\|G\|_2$ proves (11) without claiming independence between the sphere's coordinates.

Gram–Schmidt applied to independent Gaussian row vectors gives such a frame: the vectors are linearly independent almost surely, and the procedure commutes with rotations. The failure event has probability zero because each new Gaussian vector has a density and a proper linear subspace has Lebesgue measure zero.

A.3 Two exponential estimates for random signs

The inequality used for sign sums follows term by term from $(2m)! = \prod_{j=1}^m (2j)(2j-1) \geq 2^m m!$:

$$\cosh s = \sum_{m \geq 0} \frac{s^{2m}}{(2m)!} \leq \sum_{m \geq 0} \frac{(s^2/2)^m}{m!} = e^{s^2/2}.$$

For the lower-tail argument, $e^{-z} \leq 1 - z + z^2/2$ for $z \geq 0$. Indeed, the difference $f(z) = 1 - z + z^2/2 - e^{-z}$ satisfies $f(0) = f'(0) = 0$ and $f''(z) = 1 - e^{-z} \geq 0$. Integrating twice gives $f(z) \geq 0$.

B Probability tools for collision and gap arguments

B.1 First moments, union bounds, and independence

For a nonnegative random variable Z and $a > 0$, the inequality $Z \geq a \mathbf{1}\{Z \geq a\}$ gives Markov's bound $\mathbb{P}(Z \geq a) \leq \mathbb{E}Z/a$. For events $E_1, \dots, E_m$, the pointwise inequality $\mathbf{1}\{\bigcup_i E_i\} \leq \sum_i \mathbf{1}\{E_i\}$ gives

$$\mathbb{P}\left(\bigcup_i E_i\right) \leq \sum_i \mathbb{P}(E_i).$$

Neither bound requires independence. Independence is used elsewhere: independent coordinates or rows factor exponential moments, and independently generated hash tables multiply probabilities that a fixed near point misses every table. Linearity of expectation still holds when collision events for different points are dependent.

B.2 The Bernoulli Hoeffding bound

Lemma B.1. Let $k \geq 1$ and let $X_1, \dots, X_k \in \{0, 1\}$ be independent, with $X = \sum_i X_i$. For every $t > 0$,

$$\mathbb{P}(X - \mathbb{E}X \geq t) \leq e^{-2t^2/k}, \quad \mathbb{P}(X - \mathbb{E}X \leq -t) \leq e^{-2t^2/k}. \quad (49)$$

Proof. For a Bernoulli variable B of mean p , let $h(s) = \ln(1 - p + pe^s) - ps$. If $p = 0$ or 1 , $B - p = 0$. Otherwise $h(0) = h'(0) = 0$ and

$$h''(s) = q_s(1 - q_s) \leq \frac{1}{4}, \quad q_s = \frac{pe^s}{1 - p + pe^s}.$$

Taylor's formula with an integral remainder gives $h(s) \leq s^2/8$ for either sign of s . Hence $\mathbb{E}e^{s(B-p)} \leq e^{s^2/8}$. Multiplying these bounds using independence gives $\mathbb{E}e^{s(X-\mathbb{E}X)} \leq e^{ks^2/8}$. Exponential Markov bounds the upper tail by $e^{-st+ks^2/8}$ for $s > 0$; choose $s = 4t/k$. For the lower tail choose $s = -4t/k$, or apply the same argument to $\sum_i (1 - X_i)$. $\square$

Thus a mean gap of size γk costs at most $e^{-2\gamma^2 k}$ per tail. This is the estimate used to distinguish near and far Hamming distances. A two-sided deviation combines two tails and has a leading factor 2; the one-sided separation events each use a single tail.

B.3 Rounding, confidence, and fixed inputs

When a statistic Z is integer-valued, $Z > \lfloor s \rfloor$ implies $Z > s$, and $Z \leq \lfloor s \rfloor$ implies $Z \leq s$. These implications justify the rounded Hamming threshold. A ceiling on a repetition count only improves a decreasing failure bound, but can change a resource bound by a constant factor that depends on the base collision probability.

For a fixed query, correctness at all J stored scales follows by a union bound on their failure events. It is not necessary that the scales be independent, or that the algorithm examine all of them. In contrast, multiplying the failure probabilities of repeated structures requires independent randomness. For many predetermined queries, include all of their bad events in the union bound; a new query chosen as a function of the stored randomness is not automatically covered by a guarantee for one fixed query.

Bibliographic notes

The Euclidean embedding theorem is due to Johnson and Lindenstrauss [1]. The Gaussian, random-subspace, and sign-matrix arguments offer complementary proofs [4, 2, 3]. Hamming search via randomized representations and the LSH framework are developed in [6, 4]. The remaining references concern the refinements and research connections discussed in Section 8.

References

- [1] William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz Mappings into a Hilbert Space. *Contemporary Mathematics*, 26:189–206, 1984.
- [2] Sanjoy Dasgupta and Anupam Gupta. An Elementary Proof of a Theorem of Johnson and Lindenstrauss. *Random Structures & Algorithms*, 22(1):60–65, 2003. [doi:10.1002/rsa.10073](https://doi.org/10.1002/rsa.10073).
- [3] Dimitris Achlioptas. Database-Friendly Random Projections: Johnson–Lindenstrauss with Binary Coins. *Journal of Computer and System Sciences*, 66(4):671–687, 2003. [doi:10.1016/S0022-0000\(03\)00025-4](https://doi.org/10.1016/S0022-0000(03)00025-4).
- [4] Piotr Indyk and Rajeev Motwani. Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality. In *Proceedings of STOC*, pages 604–613, 1998. [Author manuscript](#).
- [5] Michael L. Fredman, János Komlós, and Endre Szemerédi. Storing a Sparse Table with $O(1)$ Worst Case Access Time. *Journal of the ACM*, 31(3):538–544, 1984.
- [6] Eyal Kushilevitz, Rafail Ostrovsky, and Yuval Rabani. Efficient Search for Approximate Nearest Neighbor in High Dimensional Spaces. *SIAM Journal on Computing*, 30(2):457–474, 2000. [doi:10.1137/S0097539798347177](https://doi.org/10.1137/S0097539798347177).
- [7] Kasper Green Larsen and Jelani Nelson. Optimality of the Johnson–Lindenstrauss Lemma. In *Proceedings of FOCS*, 2017. [arXiv:1609.02094](https://arxiv.org/abs/1609.02094).
- [8] Daniel M. Kane and Jelani Nelson. Sparser Johnson–Lindenstrauss Transforms. *Journal of the ACM*, 61(1), Article 4, 2014. [arXiv:1012.1577](https://arxiv.org/abs/1012.1577).
- [9] Nir Ailon and Bernard Chazelle. The Fast Johnson–Lindenstrauss Transform and Approximate Nearest Neighbors. *SIAM Journal on Computing*, 39(1):302–322, 2009. [doi:10.1137/060673096](https://doi.org/10.1137/060673096).
- [10] Sanjoy Dasgupta, Charles F. Stevens, and Saket Navlakha. A Neural Algorithm for a Fundamental Computing Problem. *Science*, 358(6364):793–796, 2017. [doi:10.1126/science.aam9868](https://doi.org/10.1126/science.aam9868).