Big Data Computing

Master's of Science in Computer Science

Sapienza University of Rome

2026/2027

Gabriele Tolomei

tolomei@di.uniroma1.it

Module 2: High-Dimensional Data Representations

Lecture 2.2: Dimensionality Reduction & Random Projections

Philosophy of Projection

> *"You don't need to measure everything precisely — you just need to measure enough, in the right way, to preserve what matters."*
> *— The spirit of the Johnson-Lindenstrauss Lemma*

Lecture 2.1 proved that high-dimensional distance metrics degrade. This lecture proves — constructively — that the damage can be controlled: a random projection into logarithmically many dimensions preserves pairwise distances up to a small, controllable distortion.

Lecture Roadmap

This lecture provides the constructive answer to the curse of dimensionality established in Lecture 2.1.

Specifically, we will study:

1. **The Dimensionality Reduction Problem** — what we want to preserve, and why exact preservation is impossible
2. **Random Projections** — construction and first properties
3. **The Johnson-Lindenstrauss Lemma** — statement and parameter analysis
4. **Proof of the JL Lemma** — the Dasgupta & Gupta (2003) elementary route
5. **Practical Implications & Limits** — when and how to apply JL in practice
6. **Synthesis** — from curse to cure

> *Every result in this lecture follows constructively from the random projection matrix. Let's build it.*

Why This Matters: Answering Lecture 2.1's Curse of Dimensionality

Lecture 2.1 ended with a precise, derived conclusion:

- Pairwise Euclidean distances in $\mathbb{R}^d$ concentrate as d grows — the relative gap between near and far distances vanishes.
- Distance-based algorithms (nearest-neighbor search, clustering, classification) lose discriminative power in raw high-dimensional space.

The natural question: can we map data from $\mathbb{R}^d$ into a much lower-dimensional space $\mathbb{R}^k$, while keeping pairwise distances approximately intact?

The Johnson-Lindenstrauss Lemma says: **yes, and a random projection achieves this**.

§ 1 — The Dimensionality Reduction Problem

The Curse Revisited: Distances Lose Meaning in High d

Recall the central result from Lecture 2.1, Section 5:

$$\frac{\text{Std. Dev.}\left[\|\mathbf{x}-\mathbf{y}\|_2^2\right]}{\mathbb{E}\left[\|\mathbf{x}-\mathbf{y}\|_2^2\right]} = O\left(\frac{1}{\sqrt{d}}\right) \xrightarrow{d\to\infty} 0$$

- All pairwise distances concentrate around the same value — the signal of "closeness" is swamped by the scale of typical distances.
- *Any algorithm hoping to exploit distance as a discriminative signal needs a way out of this concentration.*

The Core Question: Can We Reduce d While Preserving Structure?

Rather than working in the original $\mathbb{R}^d$ space, suppose we map all n data points into a much lower-dimensional space $\mathbb{R}^k$, $k \ll d$:

$$f : \mathbb{R}^d \longrightarrow \mathbb{R}^k$$

We want this map f to **preserve the geometric structure** of the original dataset — specifically, pairwise distances between points.

What "Preserving Structure" Means

The goal is not to preserve the data's values or coordinates — only to preserve the **pairwise distance relationships** between points:

- If $\mathbf{x}$ and $\mathbf{y}$ are close in $\mathbb{R}^d$, they should remain close in $\mathbb{R}^k$.
- If $\mathbf{x}$ and $\mathbf{z}$ are far in $\mathbb{R}^d$, they should remain far in $\mathbb{R}^k$.
- Downstream distance-based algorithms (nearest-neighbor, clustering) operating on $f(\mathbf{x})$ should produce approximately the same results as operating on $\mathbf{x}$ directly.

The Isometry Concept: Distance-Preserving Maps

A map $f : \mathbb{R}^d \to \mathbb{R}^k$ is an **isometry** if it preserves all pairwise distances exactly:

$$\|f(\mathbf{x}) - f(\mathbf{y})\|_2 = \|\mathbf{x} - \mathbf{y}\|_2 \quad \text{for all } \mathbf{x}, \mathbf{y} \in \mathbb{R}^d$$

- Examples of isometries: rotations, reflections, translations — all of which preserve distances perfectly.
- An isometry from $\mathbb{R}^d$ to $\mathbb{R}^k$ with $k < d$ would compress the data without any geometric distortion.

Exact Isometry vs. Approximate Isometry

Exact dimension reduction via isometry is generally impossible when $k < d$:

- Any linear map from $\mathbb{R}^d$ to $\mathbb{R}^k$ with $k < d$ must have a non-trivial null space — some vectors map to zero, destroying their distances.
- Intuitively: you cannot fold a 3D object flat into 2D without creasing or stretching it.

*However, exact preservation is not what we need — an **approximate** isometry, with controlled distortion, is sufficient for most downstream applications.*

Approximate Isometry as a Tractable Relaxation

An **approximate isometry** (or **embedding with distortion** ϵ) requires only that distances are preserved up to a $(1 \pm \epsilon)$ multiplicative factor:

$$(1-\epsilon)\|\mathbf{x}-\mathbf{y}\|_2 \;\leq\; \|f(\mathbf{x})-f(\mathbf{y})\|_2 \;\leq\; (1+\epsilon)\|\mathbf{x}-\mathbf{y}\|_2$$

- This allows a small, bounded relative distortion of each pairwise distance.
- For $\epsilon = 0.1$ (10% distortion), downstream algorithms that are reasonably robust to small perturbations in distances remain reliable.

The $(1 \pm \epsilon)$-Distortion Criterion

The two-sided inequality deserves careful reading:

- **Lower bound** $(1-\epsilon)\|\mathbf{x}-\mathbf{y}\| \leq \|f(\mathbf{x})-f(\mathbf{y})\|$: the projected distance cannot be *too small* — nearby points remain nearby.
- **Upper bound** $\|f(\mathbf{x})-f(\mathbf{y})\| \leq (1+\epsilon)\|\mathbf{x}-\mathbf{y}\|$: the projected distance cannot be *too large* — far points remain far.

Together, these ensure the relative ordering and scale of distances is approximately preserved throughout the embedding.

Formal Setup: n Points in $\mathbb{R}^d$, Target Dimension k

The setting for the JL Lemma, stated precisely:

- We have a set $S = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^d$ of n points.
- We seek a map $f : \mathbb{R}^d \to \mathbb{R}^k$ for some target dimension $k \ll d$.
- We want f to satisfy the $(1 \pm \epsilon)$-distortion criterion for **all** $\binom{n}{2}$ pairs simultaneously — not just for a typical pair.

The central challenge: satisfying this for all pairs simultaneously makes the problem significantly harder than satisfying it for just one pair.

Two Families of Solutions: Data-Dependent vs. Data-Oblivious

Before building the random-projection solution, it is worth locating it within the broader landscape of dimensionality reduction techniques:

- **Data-dependent methods** (e.g., **PCA**, **SVD**) compute the reduction map *from* the dataset itself — they examine the data's covariance structure and choose directions that best explain its variance.
- **Data-oblivious methods** (e.g., the **random projections** built in this lecture) choose the reduction map *independently* of the data, before ever looking at it.

Both are valid answers to "how do we reduce d?" — but they make very different trade-offs, especially at scale.

PCA/SVD: The Classical, Data-Dependent Baseline

Principal Component Analysis (PCA), computed via the **Singular Value Decomposition (SVD)**, is the classical approach to dimensionality reduction:

- Given data $\mathbf{X} \in \mathbb{R}^{n \times d}$, PCA finds the k orthogonal directions of maximum variance.
- Equivalently, the SVD $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ gives the optimal rank-k approximation $\mathbf{X}_k = \mathbf{U}_k\mathbf{\Sigma}_k\mathbf{V}_k^\top$ (Eckart–Young theorem).
- This is **optimal for reconstruction error** — no other rank-k linear map approximates $\mathbf{X}$ more accurately in Frobenius norm.

PCA/SVD is the right tool when the goal is faithful reconstruction or interpretable, variance-maximizing axes.

Why PCA/SVD Is Harder to Scale Out

The optimality of PCA/SVD comes at a computational cost that matters directly for this course's setting:

- Computing the exact SVD of $\mathbf{X} \in \mathbb{R}^{n \times d}$ costs roughly $O(nd \cdot \min(n, d))$ — a **global operation** over the entire dataset.
- The projection directions (singular vectors) depend on **relationships across all data points**, so they cannot be fixed in advance.
- Distributed SVD requires **communication and coordination across workers** (e.g., aggregating $\mathbf{X}^\top \mathbf{X}$ or performing distributed QR/SVD), unlike a fixed random projection, which can be applied independently on each worker.

This is precisely the gap that motivates JL's data-oblivious approach in a scale-out setting.

Random Projections (JL): The Data-Oblivious Alternative

By contrast, the random projection matrix $\mathbf{A}$ built in this lecture:

- Is drawn **without ever looking at the data** — the same $\mathbf{A}$ works for *any* dataset, with high probability.
- Requires **no global pass, no covariance computation, no eigendecomposition** — just $O(kd)$ multiplications per point.
- Is **embarrassingly parallel**: each point is projected independently, making it a natural fit for MapReduce/Spark-style pipelines (Module 1).

The trade-off: JL gives distortion guarantees on pairwise distances, not optimal reconstruction — it solves a different (and, at scale, more tractable) problem than PCA/SVD.

Choosing Between Them in Practice

	PCA / SVD	Random Projection (JL)
Depends on data?	Yes (data-dependent)	No (data-oblivious)
Optimizes for	Reconstruction error	Pairwise distance preservation
Cost	$O(nd \cdot \min(n, d))$, global	$O(kd)$ per point, embarrassingly parallel
Target dimension	Chosen by variance explained	$k = O(\log n / \epsilon^2)$, independent of d
Best suited for	Interpretability, compression, denoising	Fast preprocessing before distance-based algorithms at scale

Note (pointer, not derivation): randomized and distributed SVD algorithms exist and narrow this gap — but they are a distinct topic from the elementary random-projection construction developed in this lecture.

✅ Checkpoint 1 — The Dimensionality Reduction Problem

Let's test our basic understanding:

1. Why is exact dimensionality reduction via isometry generally impossible when $k < d$?
2. State the $(1 \pm \epsilon)$-distortion criterion precisely, and explain what each inequality guarantees.
3. Why must the distortion criterion hold for **all** $\binom{n}{2}$ pairs, not just a single pair?
4. What property of downstream algorithms makes approximate distance preservation sufficient?
5. Contrast PCA/SVD and random projections along two axes: what each optimizes for, and why one scales out more naturally than the other.

Write down your thoughts before moving to Section 2.

§ 2 — Random Projections

The Idea: Project Onto a Random Low-Dimensional Subspace

The construction at the heart of JL is elegantly simple: **pick a random linear map** from $\mathbb{R}^d$ to $\mathbb{R}^k$.

- Rather than carefully designing a projection tailored to a specific dataset, simply draw the projection matrix at random.
- The remarkable fact — which the JL Lemma makes precise — is that a random projection works well for **any** dataset, with high probability.

Why Random? Intuition from Equatorial Concentration

The intuition behind why random projections work connects directly to Lecture 2.1, Section 4:

- Recall: most of a high-dimensional sphere's surface area concentrates near any fixed equator.
- Equivalently: a random direction in $\mathbb{R}^d$ is nearly orthogonal to any fixed direction.
- This means a random projection "sees" all directions in $\mathbb{R}^d$ roughly equally — it does not systematically distort any particular direction more than any other.

Isotropy of the random projection is the geometric reason it preserves distances without knowing the data in advance.

Constructing a Random Projection Matrix

Define the random projection matrix $\mathbf{A} \in \mathbb{R}^{k \times d}$ as follows:

$$A_{ij} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1), \quad i = 1, \ldots, k, \quad j = 1, \ldots, d$$

Each of the $k \times d$ entries is drawn independently from the standard normal distribution. The projected image of a point $\mathbf{x} \in \mathbb{R}^d$ is then:

$$f(\mathbf{x}) = \frac{1}{\sqrt{k}} \mathbf{A}\mathbf{x} \in \mathbb{R}^k$$

The scaling factor $1/\sqrt{k}$ is essential — its role is clarified shortly.

Gaussian Random Matrices: Entry Distribution

The choice of $\mathcal{N}(0,1)$ entries is not arbitrary:

- **Rotational invariance:** any orthogonal transformation of a Gaussian random vector produces another Gaussian random vector with the same distribution.
- This means: the distribution of $\mathbf{Ax}$ depends only on $||\mathbf{x}||_2$, not on the direction of $\mathbf{x}$.
- Consequently, all directions in $\mathbb{R}^d$ are treated identically by the projection — the isotropy property mentioned on the previous slide.

The Normalized Projection: Scaling to Preserve Expected Distances

For a fixed vector $\mathbf{x} \in \mathbb{R}^d$ with $\|\mathbf{x}\|_2 = 1$, each component $(\mathbf{Ax})_i = \sum_{j=1}^{d} A_{ij}x_j$ is a linear combination of d i.i.d. $\mathcal{N}(0,1)$ terms, hence itself Gaussian with mean 0 and variance $\|\mathbf{x}\|_2^2$ — i.e. $(\mathbf{Ax})_i \sim \mathcal{N}(0, \|\mathbf{x}\|_2^2)$ (derivation on the next slide). For a unit vector this is exactly $\mathcal{N}(0,1)$. Therefore:

$$\mathbb{E}\left[\|\mathbf{Ax}\|_2^2\right] = \mathbb{E}\left[\sum_{i=1}^{k}\left(\sum_{j=1}^{d} A_{ij}x_j\right)^2\right] = k \cdot \|\mathbf{x}\|_2^2$$

Dividing by k (i.e., scaling by $1/\sqrt{k}$) gives:

$$\mathbb{E}\left[\left\|\frac{1}{\sqrt{k}}\mathbf{Ax}\right\|_2^2\right] = \|\mathbf{x}\|_2^2$$

The scaling factor $1/\sqrt{k}$ ensures the projection is unbiased: expected projected length equals original length.

Deriving the Expectation, Step by Step

Where does $\mathbb{E}[\|\mathbf{Ax}\|_2^2] = k \cdot \|\mathbf{x}\|_2^2$ actually come from? Three short steps:

1. Each row is Gaussian. Fix row i. Since $A_{i1}, \ldots, A_{id}$ are independent $\mathcal{N}(0,1)$ and $x_1, \ldots, x_d$ are fixed constants, $(\mathbf{Ax})_i = \sum_j A_{ij} x_j$ is Gaussian with:

$$\mathbb{E}[(\mathbf{Ax})_i] = \sum_j x_j \, \mathbb{E}[A_{ij}] = 0, \qquad \mathrm{Var}[(\mathbf{Ax})_i] = \sum_j x_j^2 \, \mathrm{Var}[A_{ij}] = \|\mathbf{x}\|_2^2$$

2. Square one row. Since the mean is 0, $\mathbb{E}[Z^2] = \mathrm{Var}(Z)$ applies directly:

$$\mathbb{E}\big[(\mathbf{Ax})_i^2\big] = \mathrm{Var}[(\mathbf{Ax})_i] = \|\mathbf{x}\|_2^2 \quad \text{for every } i = 1, \ldots, k$$

3. Sum the k rows. By linearity of expectation (no independence needed here):

$$\mathbb{E}\big[\|\mathbf{Ax}\|_2^2\big] = \sum_{i=1}^{k} \mathbb{E}\big[(\mathbf{Ax})_i^2\big] = \sum_{i=1}^{k} \|\mathbf{x}\|_2^2 = k \cdot \|\mathbf{x}\|_2^2$$

Intuition: each of the k rows is an independent, unbiased 1-D Gaussian "measurement" of $\|\mathbf{x}\|_2$; summing k such squared measurements scales the expected total by k. The $1/\sqrt{k}$ normalization on the previous slide simply undoes this scaling.

Why Gaussian Entries Are the Right Choice: Rotational Invariance

The key algebraic property of Gaussian random matrices used in the JL proof:

> *If* $\mathbf{g} = (g_1, \ldots, g_d)$ *with* $g_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1)$*, then for any fixed unit vector* $\mathbf{u} \in \mathbb{R}^d$*:*
>
> $$\mathbf{g} \cdot \mathbf{u} = \sum_{j=1}^{d} g_j u_j \sim \mathcal{N}(0,1)$$

This means the projection of $\mathbf{u}$ onto any single row of $\mathbf{A}$ is always a standard normal — regardless of which direction $\mathbf{u}$ points. This isotropy is what makes the analysis tractable.

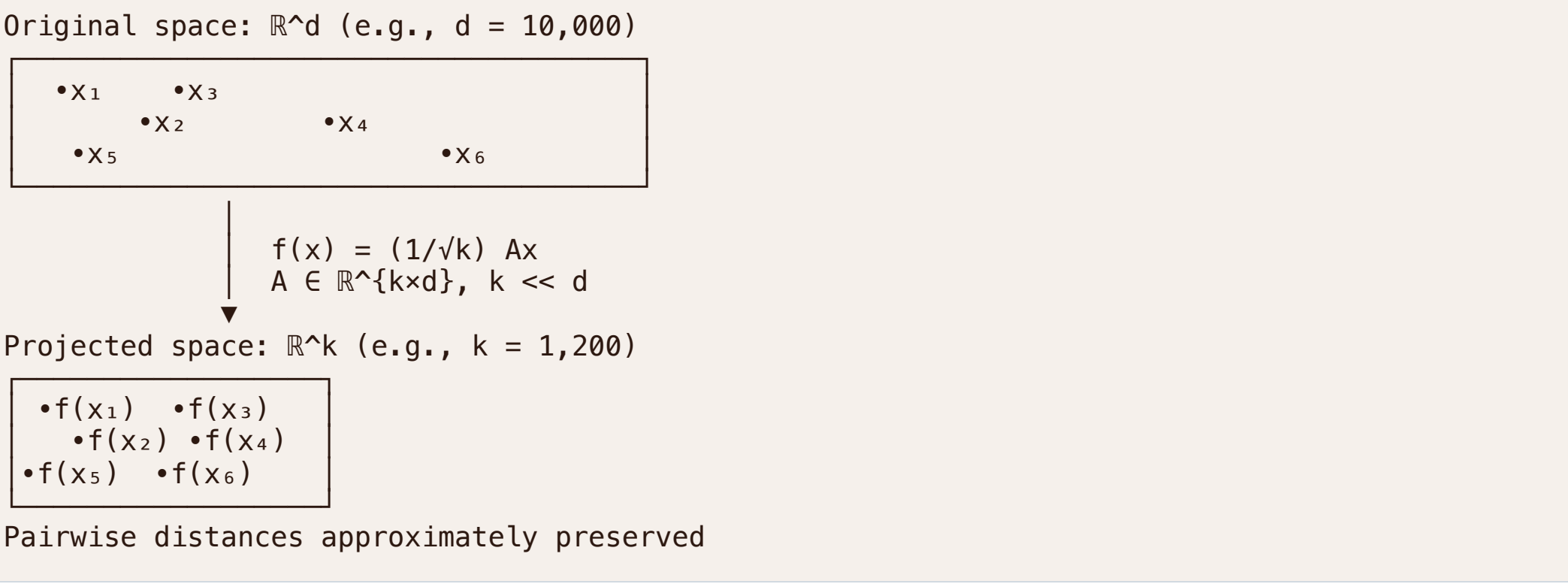
The Projection Operation, Defined Formally

Given the normalized random matrix $f(\mathbf{x}) = \frac{1}{\sqrt{k}}\mathbf{A}\mathbf{x}$, the squared projected distance between two points $\mathbf{x}, \mathbf{y}$ is:

$$\|f(\mathbf{x}) - f(\mathbf{y})\|_2^2 = \left\| \frac{1}{\sqrt{k}}\mathbf{A}(\mathbf{x} - \mathbf{y}) \right\|_2^2 = \frac{1}{k}\|\mathbf{A}\mathbf{u}\|_2^2$$

where $\mathbf{u} = \mathbf{x} - \mathbf{y}$. *The entire analysis reduces to understanding the behavior of $\|\mathbf{A}\mathbf{u}\|_2^2/k$ for a fixed vector $\mathbf{u}$ — a single random variable.*

Diagram: Projection From $\mathbb{R}^d$ to $\mathbb{R}^k$



The Role of the Scaling Factor $\sqrt{d/k}$

An alternative (equivalent) way to state the normalization:

$$f(\mathbf{x}) = \sqrt{\frac{d}{k}} \cdot \frac{1}{\sqrt{d}} \mathbf{A}\mathbf{x}$$

Here $\frac{1}{\sqrt{d}}\mathbf{A}$ is a matrix of i.i.d. $\mathcal{N}(0, 1/d)$ entries. The factor $\sqrt{d/k}$ compensates for the dimensionality reduction: projecting from d dimensions to $k < d$ shrinks lengths by a factor of roughly $\sqrt{k/d}$, and this scaling undoes that shrinkage in expectation.

✅ Checkpoint 2 — Random Projections

Let's test our understanding:

1. Why are Gaussian entries the natural choice for a random projection matrix?
2. Show that $\mathbb{E}[\|f(\mathbf{x})\|_2^2] = \|\mathbf{x}\|_2^2$ for the normalized random projection $f(\mathbf{x}) = \frac{1}{\sqrt{k}}\mathbf{A}\mathbf{x}$.
3. Why does the entire distance-preservation analysis reduce to studying $\|\mathbf{A}\mathbf{u}\|_2^2/k$ for a single vector $\mathbf{u}$?
4. How does the rotational invariance of Gaussian distributions support the isotropy of the projection?

Discuss with your colleague or write down your answers.

§ 3 — The Johnson-Lindenstrauss Lemma

Statement of the JL Lemma: Informal Version

> *For any set of n points in high-dimensional Euclidean space, there exists a map into $O(\log n/\epsilon^2)$ dimensions that approximately preserves all pairwise distances up to a $(1 \pm \epsilon)$ factor — and a random projection finds such a map with high probability.*

Before stating this formally, let's appreciate what it claims: the target dimension k depends only on n (the number of points) and ϵ (the desired accuracy) — not at all on the original dimension d.

Formal Statement of the JL Lemma

Lemma (Johnson & Lindenstrauss, 1984; Dasgupta & Gupta, 2003). *Let $\epsilon \in (0, 1/2)$ and let $S = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subset \mathbb{R}^d$. Set*

$$k = \left\lceil \frac{24 \ln n}{\epsilon^2} \right\rceil$$

Then for a random matrix $\mathbf{A}$ with i.i.d. $\mathcal{N}(0,1)$ entries, the map $f(\mathbf{x}) = \frac{1}{\sqrt{k}} \mathbf{A}\mathbf{x}$ satisfies, with probability at least $1 - 1/n$:

$$(1-\epsilon)\|\mathbf{x}_i - \mathbf{x}_j\|_2 \le \|f(\mathbf{x}_i) - f(\mathbf{x}_j)\|_2 \le (1+\epsilon)\|\mathbf{x}_i - \mathbf{x}_j\|_2$$

for all $\binom{n}{2}$ *pairs (i, j) simultaneously.*

Unpacking the Formal Statement: What Each Quantifier Means

The formal statement contains several layers worth reading carefully:

- **"for all pairs simultaneously":** the guarantee applies jointly to every pair — not just each pair individually with high probability.
- **"with probability at least $1 - 1/n$":** over the randomness of $\mathbf{A}$, the event that all pairs are preserved happens with very high probability.
- **"$\epsilon \in (0, 1/2)$":** the distortion parameter must be less than 1/2 for the proof to work — very large distortion makes the lemma vacuous.

The Key Parameters: n Points, ϵ Distortion, Target Dimension k

Three parameters govern the JL Lemma:

Parameter	Role	Direction
n	Number of points	More points $\rightarrow$ larger required k
ϵ	Allowed distortion	Tighter accuracy $\rightarrow$ larger required k
k	Target dimension	Output dimension we are solving for

And crucially:

- d (original dimension) **does not appear** in the bound for k.

The Dimension Bound: $k = O(\log n / \epsilon^2)$

The required target dimension is:

$$k = O\left(\frac{\log n}{\epsilon^2}\right)$$

This is a **doubly logarithmic** compression: even if $d = 10^6$ (one million dimensions), $n = 10^6$ points require only:

$$k \approx \frac{24 \ln(10^6)}{0.01} = \frac{24 \times 13.8}{0.01} \approx 33{,}000 \text{ dimensions}$$

for $\epsilon = 0.1$ — a compression by a factor of 30, regardless of d.

Why k Depends on $\log n$, Not n Itself

The $\log n$ dependence comes directly from the **union bound** in the proof (Section 4):

- There are $\binom{n}{2} \approx n^2/2$ pairs to satisfy simultaneously.
- Each pair fails with exponentially small probability $e^{-\Omega(k)}$.
- To make the total failure probability small (say $< 1/n$) across all $n^2/2$ pairs requires:

$$e^{-\Omega(k)} \cdot n^2 \lesssim \frac{1}{n} \implies k = \Omega(\log n)$$

The $\log n$ factor is precisely the price of ensuring all pairs succeed simultaneously.

Why k Depends on ϵ^{-2}

The ϵ^{-2} factor comes from the **concentration rate** of the projected distance around its expectation:

- The probability that a single pair's projected distance falls outside $(1 \pm \epsilon)$ of its true value is roughly $e^{-k\epsilon^2/C}$ (for a constant C).
- Achieving a small failure probability for one pair requires $k \gtrsim C/\epsilon^2$.
- Combining with the $\log n$ from the union bound: $k = \Omega(\log n/\epsilon^2)$.

Both the $\log n$ and ϵ^{-2} factors will emerge naturally from the proof in Section 4.

Worked Numeric Example: $n = 10^6$ Points, $\epsilon = 0.1$

Computing the required k explicitly with the JL dimension bound $k = \lceil 24 \ln n / \epsilon^2 \rceil$:

$$k = \left\lceil \frac{24 \ln(10^6)}{(0.1)^2} \right\rceil = \left\lceil \frac{24 \times 13.816}{0.01} \right\rceil = \lceil 33{,}158 \rceil = 33{,}158$$

So one million points in any number of dimensions d can be embedded into $\mathbb{R}^{33158}$ with at most 10% pairwise distance distortion.

The Remarkable Conclusion: $k \approx 33{,}000$ Regardless of Original d

This calculation deserves emphasis: d plays no role.

Original Dimension d	n	ϵ	Required k
1,000	10^6	0.1	33,158
100,000	10^6	0.1	33,158
10,000,000	10^6	0.1	33,158

Whether the original space has a thousand or ten million dimensions, the same random projection into the same k-dimensional space works with high probability.

✅ Checkpoint 3 — The JL Lemma

Let's test our understanding:

1. State the JL Lemma formally, including the dimension bound and the success probability.
2. Explain, in words, why the target dimension k depends on $\log n$ rather than n itself.
3. Why does k scale as ϵ^{-2} rather than ϵ^{-1}?
4. Compute k for $n = 10^4$ points and $\epsilon = 0.2$ using the bound $k = \lceil 24 \ln n / \epsilon^2 \rceil$.

Review your answers before moving on.

§ 4 — Proof of the JL Lemma

Proof Strategy Overview

The Dasgupta & Gupta (2003) proof proceeds in four steps:

1. **Reduce to a single vector:** fix one pair $(\mathbf{x}_i, \mathbf{x}_j)$; it suffices to analyze $\mathbf{u} = \mathbf{x}_i - \mathbf{x}_j$ (WLOG $\|\mathbf{u}\|_2 = 1$).
2. **Identify the key random variable:** $\|\mathbf{A}\mathbf{u}\|_2^2 / k$ and connect it to a chi-squared distribution.
3. **Bound its concentration:** use moment generating functions to bound tail probabilities.
4. **Apply the union bound:** combine one-pair bounds over all $\binom{n}{2}$ pairs.

Each step is self-contained and elementary in the sense of requiring no tools beyond standard probability.

Step 1: Fixing One Pair — Reducing to a Single Vector

Fix any pair $(\mathbf{x}_i, \mathbf{x}_j)$ and let $\mathbf{u} = (\mathbf{x}_i - \mathbf{x}_j)/\|\mathbf{x}_i - \mathbf{x}_j\|_2$ — a unit vector in the difference direction.

The distortion condition for this pair is:

$$(1-\epsilon) \leq \frac{\|f(\mathbf{x}_i) - f(\mathbf{x}_j)\|_2}{\|\mathbf{x}_i - \mathbf{x}_j\|_2} \leq (1+\epsilon)$$

which, squaring and substituting $f(\mathbf{x}) = \frac{1}{\sqrt{k}}\mathbf{A}\mathbf{x}$, is equivalent to:

$$(1-\epsilon)^2 \leq \frac{\|\mathbf{A}\mathbf{u}\|_2^2}{k} \leq (1+\epsilon)^2$$

So the entire analysis for one pair reduces to bounding the concentration of $\|\mathbf{A}\mathbf{u}\|_2^2/k$ around 1.

The Key Probabilistic Claim (One Pair)

We need to bound:

$$\Pr\left[\left|\frac{\|\mathbf{A}\mathbf{u}\|_2^2}{k} - 1\right| > \epsilon\right]$$

for a fixed unit vector $\mathbf{u}$ and a $k \times d$ matrix $\mathbf{A}$ with i.i.d. $\mathcal{N}(0,1)$ entries.

Using the $(1 \pm \epsilon)^2 \approx (1 \pm 2\epsilon)$ approximation (valid for small ϵ), this simplifies to bounding the probability that a chi-squared random variable deviates from its mean.

Step 2: The Chi-Squared Distribution

By rotational invariance of Gaussian random matrices (Section 2), $\mathbf{Au}$ has the same distribution as $\mathbf{g} = (g_1, \ldots, g_k)$ with $g_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1)$, since $\|\mathbf{u}\|_2 = 1$.

Therefore:

$$\|\mathbf{Au}\|_2^2 = \sum_{i=1}^{k} g_i^2 \;\sim\; \chi_k^2$$

This is a **chi-squared random variable with k degrees of freedom** — a classical, well-studied distribution.

The next two slides unpack exactly how this reduction happens, row by row, and then state the definition being invoked.

Unpacking the Reduction: Row by Row

$\mathbf{A}$ has k rows, each an independent vector of d i.i.d. $\mathcal{N}(0,1)$ entries. Write row i as $\mathbf{a}_i = (A_{i1}, \ldots, A_{id})$, so that:

$$(\mathbf{Au})_i = \mathbf{a}_i \cdot \mathbf{u} = \sum_{j=1}^{d} A_{ij} u_j$$

Within one row: this is exactly the dot product from the "Gaussian Random Matrices" slide (Section 2) — a fixed unit vector $\mathbf{u}$ against a row of i.i.d. standard normals. Rotational invariance of the Gaussian gives $\mathbf{a}_i \cdot \mathbf{u} \sim \mathcal{N}(0,1)$, **regardless of which direction $\mathbf{u}$ points.**

Across rows: the k rows $\mathbf{a}_1, \ldots, \mathbf{a}_k$ are independent of each other (different, non-overlapping entries of $\mathbf{A}$), so the k resulting scalars $(\mathbf{Au})_1, \ldots, (\mathbf{Au})_k$ are independent too.

Conclusion: $\mathbf{Au} = ((\mathbf{Au})_1, \ldots, (\mathbf{Au})_k)$ *behaves exactly like a vector* $\mathbf{g} = (g_1, \ldots, g_k)$ *of* k *i.i.d.* $\mathcal{N}(0,1)$ *variables — one standard normal per row, all mutually independent.*

Definition: The Chi-Squared Distribution

This is the point where a *new named distribution* enters the proof — it is simply a definition, not a new theorem:

> ***Definition.*** *If* $g_1, \ldots, g_k \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1)$*, then the random variable*
>
> $$Z = \sum_{i=1}^{k} g_i^2$$
>
> *is said to follow the* ***chi-squared distribution with*** k ***degrees of freedom****, written* $Z \sim \chi^2_k$*.*

- "Degrees of freedom" here just means: *the number of independent standard normals being squared and summed.*
- Since the previous slide showed $(\mathbf{Au})_1, \ldots, (\mathbf{Au})_k$ are themselves k i.i.d. $\mathcal{N}(0,1)$ variables, plugging them into this definition gives $\|\mathbf{Au}\|_2^2 = \sum_i (\mathbf{Au})_i^2 \sim \chi^2_k$ immediately — **by definition**, not by a separate derivation.

The Chi-Squared Distribution: Moments

The chi-squared distribution with k degrees of freedom, $\chi^2_k = \sum_{i=1}^{k} g_i^2$, has:

$$\mathbb{E}[\chi^2_k] = k, \qquad \mathrm{Var}[\chi^2_k] = 2k$$

Where these come from, briefly:

For a single $g_i \sim \mathcal{N}(0,1)$, $\mathbb{E}[g_i^2] = \mathrm{Var}(g_i) = 1$, so by linearity of expectation, $\mathbb{E}[\sum_i g_i^2] = \sum_i \mathbb{E}[g_i^2] = k$.

For the variance, $\mathrm{Var}(g_i^2) = \mathbb{E}[g_i^4] - (\mathbb{E}[g_i^2])^2 = 3 - 1 = 2$ (using the standard normal's fourth moment $\mathbb{E}[g_i^4] = 3$); since the g_i are independent, variances of the sum add: $\mathrm{Var}[\sum_i g_i^2] = \sum_i \mathrm{Var}(g_i^2) = 2k$.

The Chi-Squared Distribution: Moments

Let $Z = \|\mathbf{Au}\|_2^2 \sim \chi_k^2$.

We know that $\mathbb{E}[Z] = k \text{ and } \operatorname{Var}(Z) = 2k$.

Now define the normalized quantity $Y = \frac{Z}{k} = \frac{\|\mathbf{Au}\|_2^2}{k}$.

Therefore, $\mathbb{E}[Y] = \mathbb{E}\left[\frac{Z}{k}\right] = \frac{1}{k}\mathbb{E}[Z] = \frac{1}{k}\, k = 1$.

And, since $\operatorname{Var}(cZ) = c^2 \operatorname{Var}(Z)$, $\operatorname{Var}(Y) = \operatorname{Var}\left(\frac{Z}{k}\right) = \frac{1}{k^2}\operatorname{Var}(Z) = \frac{1}{k^2}\, 2k = \frac{2}{k}$.

> *The normalized quantity $\|\mathbf{Au}\|_2^2/k$ is unbiased (mean = 1) and has variance shrinking as $1/k$ — consistent with the concentration we need.*

Splitting the Two-Sided Bound into Two One-Sided Tails

Before invoking any inequality, notice that the event we care about is really **two** events combined:

$$\left|\frac{Z}{k} - 1\right| > \epsilon \quad \Longleftrightarrow \quad \left(\frac{Z}{k} > 1+\epsilon\right) \text{ or } \left(\frac{Z}{k} < 1-\epsilon\right)$$

where $Z = \|\mathbf{Au}\|_2^2 \sim \chi_k^2$. Multiplying through by k:

$$\Pr\left[\left|\frac{Z}{k} - 1\right| > \epsilon\right] = \Pr\left[Z > (1+\epsilon)k\right] + \Pr\left[Z < (1-\epsilon)k\right]$$

(the two events are disjoint — Z can't be simultaneously above and below its mean — so their probabilities simply add; no union bound "slack" is lost here). *This is why the proof handles the **upper tail** $\Pr[Z > (1+\epsilon)k]$ and the **lower tail** $\Pr[Z < (1-\epsilon)k]$ as two separate, symmetric-looking derivations, then adds the two resulting bounds at the end.*

Step 3: Concentration via Moment Generating Functions

To get a sharper (exponential) tail bound rather than the polynomial bound from Chebyshev, Dasgupta & Gupta use the MGF of the chi-squared distribution.

For $Z \sim \chi^2_k$ and $t < 1/2$:

$$\mathbb{E}[e^{tZ}] = (1 - 2t)^{-k/2}$$

This is the standard MGF of the chi-squared — its exact closed form is what enables the "elementary" nature of the proof. *(Derived from scratch on the next slide.)*

Deriving the Chi-Squared MGF From Scratch

Recall $Z = \sum_{i=1}^{k} g_i^2$ with $g_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1)$. Since the g_i are independent, the MGF of a sum of independent variables is the **product** of their individual MGFs:

$$\mathbb{E}[e^{tZ}] = \mathbb{E}\left[e^{t\sum_i g_i^2}\right] = \prod_{i=1}^{k} \mathbb{E}\left[e^{tg_i^2}\right]$$

So it suffices to compute $\mathbb{E}[e^{tg^2}]$ for a single $g \sim \mathcal{N}(0,1)$, then raise it to the k-th power. Writing out the expectation as a Gaussian integral:

$$\mathbb{E}[e^{tg^2}] = \int_{-\infty}^{\infty} e^{tx^2} \cdot \frac{1}{\sqrt{2\pi}} e^{-x^2/2}\, dx = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-x^2(1-2t)/2}\, dx$$

This is again a Gaussian integral (just with a rescaled variance), valid whenever $1 - 2t > 0$, i.e. $t < 1/2$ — **this is exactly why the MGF is only defined for $t < 1/2$**. Using $\int e^{-ax^2/2} dx = \sqrt{2\pi/a}$ with $a = 1 - 2t$:

$$\mathbb{E}[e^{tg^2}] = \frac{1}{\sqrt{2\pi}} \cdot \sqrt{\frac{2\pi}{1-2t}} = (1-2t)^{-1/2}$$

Raising to the k-th power (for the k independent, identically-distributed g_i^2 terms) gives exactly $\mathbb{E}[e^{tZ}] = (1-2t)^{-k/2}$, as claimed.

The Chernoff Method: Why Exponentiate, Then Apply Markov

We want $\Pr[Z > (1+\epsilon)k]$, but Z itself has no simple tail formula. The **Chernoff trick** works around this in two moves:

1. Exponentiate (a reversible, monotone step). For any $t > 0$, $z \mapsto e^{tz}$ is strictly increasing, so the event $\{Z > (1+\epsilon)k\}$ is *exactly* the same event as $\{e^{tZ} > e^{t(1+\epsilon)k}\}$ — nothing is lost or approximated here, we've simply relabeled the same event.

2. Apply Markov's inequality to the new random variable e^{tZ}. Markov's inequality says: for any non-negative random variable W and any $a > 0$, $\Pr[W \geq a] \leq \mathbb{E}[W]/a$. Setting $W = e^{tZ}$ (which is indeed non-negative) and $a = e^{t(1+\epsilon)k}$:

$$\Pr[Z > (1+\epsilon)k] = \Pr\left[e^{tZ} > e^{t(1+\epsilon)k}\right] \leq \frac{\mathbb{E}[e^{tZ}]}{e^{t(1+\epsilon)k}}$$

This is precisely the inequality on the next slide. The payoff of exponentiating first: Markov's inequality applied directly to Z would only give a weak bound, but applied to e^{tZ} it captures information from **all** the moments of Z at once (since e^{tZ}'s expectation is the MGF) — which is what produces an exponentially decaying bound instead of a polynomial one like Chebyshev's.

Why we're free to pick any $t > 0$: the inequality above holds for *every* valid $t \in (0, 1/2)$, so it holds in particular for whichever t makes the right-hand side smallest — this is exactly the "optimizing over t" step on the next slide.

Applying the MGF: Upper Tail Bound

Using the Chernoff method for the upper tail (bounding $\Pr[Z > (1+\epsilon)k]$):

For $t > 0$ and applying Markov's inequality to e^{tZ}:

$$\Pr[Z > (1+\epsilon)k] \leq \frac{\mathbb{E}[e^{tZ}]}{e^{t(1+\epsilon)k}} = \frac{(1-2t)^{-k/2}}{e^{t(1+\epsilon)k}}$$

Optimizing over t by setting $t = \epsilon/(2(1+\epsilon))$ and simplifying yields (Dasgupta & Gupta, 2003):

$$\Pr[Z > (1+\epsilon)k] \leq \left[(1+\epsilon)e^{-\epsilon}\right]^{k/2}$$

Full derivation of where $t = \epsilon/(2(1+\epsilon))$ comes from, and how it simplifies, on the next slide.

Optimizing Over t: Where $t = \epsilon/(2(1+\epsilon))$ Comes From

The bound $\Pr[Z > (1+\epsilon)k] \le (1-2t)^{-k/2} e^{-t(1+\epsilon)k}$ holds for **every** $t \in (0, 1/2)$ — so it holds, in particular, for whichever t makes the right-hand side as small as possible. We minimize it.

1. Take logs to turn a product into a sum. Since $\ln(\cdot)$ is increasing, minimizing the bound is the same as minimizing its log:

$$g(t) = \ln\left[(1-2t)^{-k/2} e^{-t(1+\epsilon)k}\right] = -\frac{k}{2}\ln(1-2t) - t(1+\epsilon)k$$

2. Differentiate with respect to t and set to zero. Using $\frac{d}{dt}\ln(1-2t) = \frac{-2}{1-2t}$:

$$g'(t) = \frac{k}{1-2t} - (1+\epsilon)k = 0 \quad\Longrightarrow\quad \frac{1}{1-2t} = 1+\epsilon$$

3. Solve for t. Cross-multiplying: $1 = (1+\epsilon)(1-2t) = (1+\epsilon) - 2t(1+\epsilon)$, so

$$2t(1+\epsilon) = \epsilon \quad\Longrightarrow\quad t = \frac{\epsilon}{2(1+\epsilon)}$$

which is exactly the value used. *(This is a minimum, not a maximum: the bound equals the trivial value 1 as $t \to 0^+$ and blows up to $+\infty$ as $t \to \frac{1}{2}^-$, and there is only one critical point in between.)*

Optimizing Over t: Plugging Back In

Substitute $t^* = \dfrac{\epsilon}{2(1+\epsilon)}$ back into the two factors of the bound separately.

The $(1-2t)^{-k/2}$ factor:

$$1 - 2t^* = 1 - \frac{\epsilon}{1+\epsilon} = \frac{(1+\epsilon)-\epsilon}{1+\epsilon} = \frac{1}{1+\epsilon} \quad \Longrightarrow \quad (1-2t^*)^{-k/2} = (1+\epsilon)^{k/2}$$

The $e^{-t(1+\epsilon)k}$ factor:

$$t^*(1+\epsilon)k = \frac{\epsilon}{2(1+\epsilon)} \cdot (1+\epsilon) \cdot k = \frac{\epsilon k}{2} \quad \Longrightarrow \quad e^{-t^*(1+\epsilon)k} = e^{-\epsilon k/2}$$

Combining the two factors:

$$\Pr[Z > (1+\epsilon)k] \leq (1+\epsilon)^{k/2} \cdot e^{-\epsilon k/2} = \left[(1+\epsilon)e^{-\epsilon}\right]^{k/2}$$

exactly matching the stated result.

Applying the MGF: Lower Tail Bound

By a symmetric argument for the lower tail (bounding $\Pr[Z < (1-\epsilon)k]$) with $t < 0$:

$$\Pr[Z < (1-\epsilon)k] \leq \left[(1-\epsilon)e^{\epsilon}\right]^{k/2}$$

Both bounds can be simplified using $\ln(1 \pm \epsilon) \leq \pm\epsilon - \epsilon^2/2 + O(\epsilon^3)$, giving:

$$\Pr\left[\left|\frac{Z}{k} - 1\right| > \epsilon\right] \leq 2e^{-k\epsilon^2/4}$$

This is the one-pair concentration bound: the probability of distortion exceeding ϵ decays exponentially in $k\epsilon^2$. (The factor of 2 out front is exactly the sum of the upper-tail and lower-tail bounds, per the splitting step above.)

Step 4: Union Bound Over All $\binom{n}{2}$ Pairs

Apply the one-pair bound $2e^{-k\epsilon^2/4}$ to all $\binom{n}{2} \leq n^2/2$ pairs simultaneously, using the union bound:

$$\Pr[\text{any pair fails}] \leq \frac{n^2}{2} \cdot 2e^{-k\epsilon^2/4} = n^2 e^{-k\epsilon^2/4}$$

Choosing k to Make the Union Bound Succeed

We want the failure probability to be at most $1/n$:

$$n^2 e^{-k\epsilon^2/4} \leq \frac{1}{n}$$

Taking logarithms:

$$-k\epsilon^2/4 \leq -3\ln n \implies k \geq \frac{12\ln n}{\epsilon^2}$$

Setting $k = \lceil 24\ln n/\epsilon^2 \rceil$ (with a factor-of-2 slack to absorb the $(1\pm\epsilon)^2 \approx (1\pm 2\epsilon)$ approximation) gives the stated dimension bound.

Unpacking the Log Algebra, Line by Line

Starting from $n^2 e^{-k\epsilon^2/4} \leq \frac{1}{n}$, take the natural log of **both sides** (log is increasing, so the inequality direction is preserved):

$$\ln\left(n^2 e^{-k\epsilon^2/4}\right) \leq \ln\left(\frac{1}{n}\right)$$

Left side: use $\ln(ab) = \ln a + \ln b$ and $\ln(n^2) = 2\ln n$:

$$\ln\left(n^2 e^{-k\epsilon^2/4}\right) = \ln(n^2) + \ln\left(e^{-k\epsilon^2/4}\right) = 2\ln n - \frac{k\epsilon^2}{4}$$

Right side: use $\ln(1/n) = -\ln n$.

So the inequality becomes:

$$2\ln n - \frac{k\epsilon^2}{4} \leq -\ln n$$

Unpacking the Log Algebra, Line by Line

Move $2 \ln n$ to the right-hand side (subtract $2 \ln n$ from both sides):

$$-\frac{k\epsilon^2}{4} \leq -\ln n - 2\ln n = -3\ln n$$

This is exactly the intermediate line shown above. **Now isolate k:** multiply both sides by $-4/\epsilon^2$. Since $-4/\epsilon^2$ is **negative**, the inequality direction flips:

$$k\epsilon^2 \geq 4 \cdot 3\ln n = 12\ln n \quad \Longrightarrow \quad k \geq \frac{12\ln n}{\epsilon^2}$$

Completing the Proof

Combining Steps 1-4:

- For any fixed pair, the distortion exceeds ϵ with probability at most $2e^{-k\epsilon^2/4}$.
- Over all $\binom{n}{2}$ pairs, the union bound gives total failure probability at most $n^2 e^{-k\epsilon^2/4}$.
- Choosing $k = \lceil 24 \ln n / \epsilon^2 \rceil$ makes this failure probability at most $1/n$.
- Therefore, with probability at least $1 - 1/n$, **all pairs simultaneously** satisfy the $(1 \pm \epsilon)$-distortion criterion.

■

The Proof Gives a Constructive Algorithm, Not Just Existence

A key feature of this proof: it is entirely constructive.

- The proof does not just assert "such a map exists" — it says "sample $\mathbf{A}$ with i.i.d. $\mathcal{N}(0,1)$ entries, and it works with probability $\geq 1 - 1/n$."
- In practice: draw $\mathbf{A}$ once, apply $f(\mathbf{x}) = \frac{1}{\sqrt{k}}\mathbf{A}\mathbf{x}$ to all n points, and the result is a valid JL embedding with overwhelming probability.
- **Time complexity of the embedding:** $O(kd)$ per point, $O(nkd)$ total — feasible when $k \ll d$.

✅ Checkpoint 4 — Proof of the JL Lemma

Let's test our understanding:

1. What distribution does $||\mathbf{Au}||_2^2$ follow for a unit vector $\mathbf{u}$ and Gaussian $\mathbf{A}$, and what are its mean and variance?
2. Why is the Chernoff/MGF method used for the tail bound, rather than Chebyshev's inequality?
3. Walk through the union bound step: why does the failure probability across all pairs depend on $\log n$?
4. What does it mean that the proof is "constructive"?

Consolidate your answers before proceeding.

§ 5 — Practical Implications & Limits

Algorithmic Consequence: JL as a Preprocessing Step

The JL Lemma is most naturally used as a **preprocessing step** before applying a downstream algorithm:

```
High-dimensional data (ℝ^d)
          |
          |  Draw A with i.i.d. N(0,1) entries
          |  Project: f(x) = (1/√k) Ax
          ▼
Low-dimensional data (ℝ^k, k = O(log n / ε²))
          |
          ▼
Apply any distance-based algorithm
(nearest-neighbor, clustering, etc.)
```

The downstream algorithm sees a dataset where d has been replaced by $k = O(\log n/\epsilon^2)$ — potentially orders-of-magnitude smaller.

Applying JL Before Distance-Based Algorithms: Callback to Lecture 2.1

Recall the practical consequences of the curse of dimensionality from Lecture 2.1, Section 6:

- Nearest-neighbor search loses discriminative power in high d (max/min distance ratio $\to 1$).
- Clustering algorithms (e.g., k-means) produce less meaningful clusters.

JL directly addresses both: after projection into $\mathbb{R}^k$ with $k = O(\log n)$, pairwise distances are preserved up to $(1 \pm \epsilon)$, making the low-dimensional space a valid substitute for distance-based computation at a fraction of the cost.

Sparse Random Projections as a Practical Alternative

Gaussian random matrices require $O(kd)$ random numbers and $O(kd)$ multiplications per point — expensive when d is very large (e.g., $d = 10^6$ for sparse text vectors).

Sparse random projections (Achlioptas, 2003) replace Gaussian entries with sparse alternatives:

$$A_{ij} = \begin{cases} +1 & \text{with prob. } 1/6 \\ 0 & \text{with prob. } 2/3 \\ -1 & \text{with prob. } 1/6 \end{cases}$$

- Same JL guarantee holds (with appropriate scaling).
- Sparsity means most entries are zero — matrix-vector multiplication becomes much faster when $\mathbf{x}$ is also sparse.

The Fast JL Transform (FJLT): Structure for Computational Efficiency

The **Fast JL Transform** replaces a dense random Gaussian matrix with a structured product $\mathbf{A} = \mathbf{PHD}$:

- $\mathbf{D}$: random diagonal sign matrix (random ± 1 on diagonal).
- $\mathbf{H}$: Hadamard matrix (fast $O(d \log d)$ transform).
- $\mathbf{P}$: random subsampling matrix.

This reduces projection time from $O(kd)$ to $O(d \log d)$ per point — crucial when d is very large. The JL guarantee still holds, with a slightly modified constant.

Limits of JL: What It Does Not Preserve

The JL Lemma guarantees **multiplicative** approximate preservation of pairwise Euclidean distances. It does not, in general, preserve:

- **Exact inner products:** $\langle f(\mathbf{x}), f(\mathbf{y}) \rangle \approx \langle \mathbf{x}, \mathbf{y} \rangle$ holds approximately (via polarization identity) but not exactly.
- **Ordering of distances:** individual nearest-neighbor relationships may change for near-ties.
- **Angles between vectors:** angular relationships are approximately preserved (since angles are related to distances on the sphere), but with some distortion.

When JL Fails or Is Insufficient

JL is not a universal solution — several settings limit its applicability:

- **Very small ϵ:** if very tight distance preservation is needed (e.g., $\epsilon = 0.001$), the required $k = O(\log n / 0.001^2) = O(10^6 \log n)$ may not be smaller than d.
- **Very large n:** the $\log n$ factor grows slowly, but for $n = 10^{100}$ (astronomically large datasets), k may still be impractically large.
- **Non-Euclidean metrics:** JL applies to Euclidean space; other distance functions (e.g., Earth Mover's Distance, edit distance) require different embedding results.

The Trade-Off: ϵ vs. k — Choosing the Right Operating Point

In practice, choosing ϵ involves a genuine trade-off:

ϵ	$k \approx 24 \ln n / \epsilon^2$ (for $n = 10^6$)	Downstream algorithm impact
0.5	~1,327	Very compact; some algorithms sensitive to 50% distortion
0.1	~33,158	Good balance for most distance-based algorithms
0.01	~3,315,800	Tight but k may exceed d — JL becomes unhelpful

For most practical applications in big data, $\epsilon \in [0.1, 0.3]$ offers a useful operating point.

✅ Checkpoint 5 — Practical Implications & Limits

Let's test our understanding:

1. Describe the JL embedding pipeline as a preprocessing step for a distance-based algorithm.
2. Why are sparse random projections preferable to Gaussian projections in high-dimensional sparse settings?
3. What does the JL Lemma *not* guarantee, besides approximate distance preservation?
4. For $n = 10^4$ points and $\epsilon = 0.1$, compute the required k and assess whether the embedding is practically useful.

Make sure you understand these trade-offs before wrapping up.

§ 6 — Synthesis & Wrap-up

From Curse (Lecture 2.1) to Cure (This Lecture)

The two lectures in this module so far form a complete argument:

Lecture 2.1	Lecture 2.2
Proved: volume concentrates near boundaries	Built: projection that spreads volume uniformly
Proved: random unit vectors are nearly orthogonal	Used: this isotropy to make the projection direction-agnostic
Proved: pairwise distances concentrate — discrimination lost	Proved: random projection approximately preserves all pairwise distances
Named: the curse of dimensionality	Provided: a constructive, probabilistic cure

Summary of Key Results and the Proof Structure

This lecture established, in sequence:

1. **Setup:** approximate isometry as the tractable relaxation of exact distance preservation.
2. **Construction:** Gaussian random projection matrices, unbiased and isotropic.
3. **The Lemma:** $k = O(\log n/\epsilon^2)$ dimensions suffice, independent of d.
4. **The Proof:** chi-squared concentration + Chernoff MGF bound + union bound over all pairs.
5. **Practice:** JL as a preprocessing step; sparse and fast variants; limits of applicability.

Key Terms Glossary

Term	Meaning
Isometry	A map that exactly preserves all pairwise distances
$(1 \pm \epsilon)$**-distortion**	Approximate distance preservation within a multiplicative factor
Random projection matrix	A matrix with i.i.d. $\mathcal{N}(0,1)$ entries, scaled by $1/\sqrt{k}$
Chi-squared distribution	Distribution of $\sum_{i=1}^{k} g_i^2$ for i.i.d. $\mathcal{N}(0,1)$ variables g_i
JL Lemma	n points can be embedded in $O(\log n/\epsilon^2)$ dimensions with $(1 \pm \epsilon)$ distortion
Union bound	$\Pr[\bigcup_i E_i] \leq \sum_i \Pr[E_i]$ — used to control all pairs simultaneously

Lecture 2.2 Complete

Recommended Reading

- Dasgupta, S., & Gupta, A. (2003). *An elementary proof of a theorem of Johnson and Lindenstrauss*. Random Structures & Algorithms, 22(1).
- Course reference materials on random projections and dimensionality reduction

Contact & Q&A

- **Course:** Big Data Computing
- **Office Hours:** Drop me a message, and we'll arrange a time to meet, either in person or online! 🙂

Thank you for your attention!