# COMPSCI 514: Algorithms for Data Science

Cameron Musco

University of Massachusetts Amherst. Fall 2026.

Lecture 13

# Logistics

- Midterm 1 grades have been posted.
- Median around an 88%, which is quite good.
- If you are worried about your grade, feel free to email me and set up a time to chat.
- Problem Set 2 is nearly done with grading, and we will post grades shortly.

# Summary

Last Few Classes:

- Approximate nearest neighbor search and solution via LSH.
- Guest lecture on solution via greedy search on 'navigable' graphs.

Today:

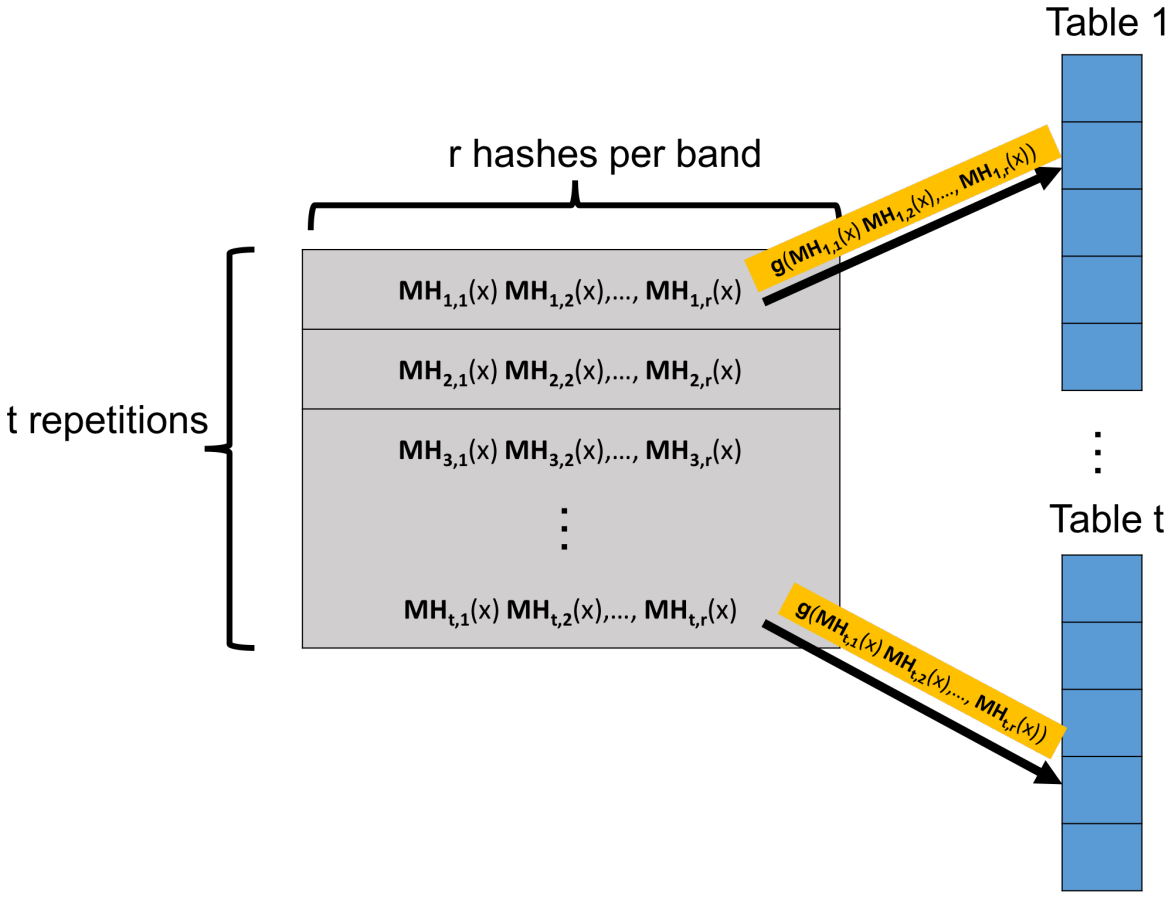
- Finish up analysis of LSH repetitions.
- Start on spectral methods unit: linear algebraic techniques for working with large and high dimensional datasets.
- Today we will cover low-distortion embeddings and the Johnson-Lindenstrauss (JL) Lemma.

If you feel shaky on your linear algebra background, you'll want to start doing some review. See the reading material for resources.

# LSH Wrap Up

# Balancing Hit Rate and Query Time

We want to balance a small probability of false negatives (a high hit rate) with a small probability of false positives (a small query time.)




Create *t* hash tables. Each is indexed into not with a single MinHash value, but with *r* values, appended together. A length *r* signature.

# Balancing Hit Rate and Query Time

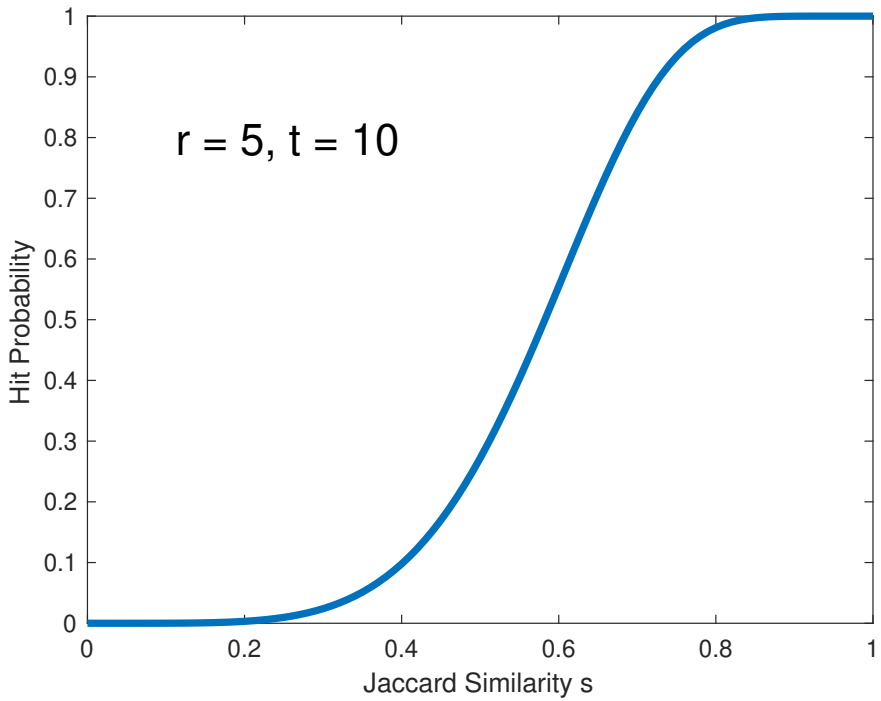
Consider searching for matches in $t$ hash tables, using MinHash signatures of length $r$. For $x$ and $y$ with Jaccard similarity $J(x, y) = s$:

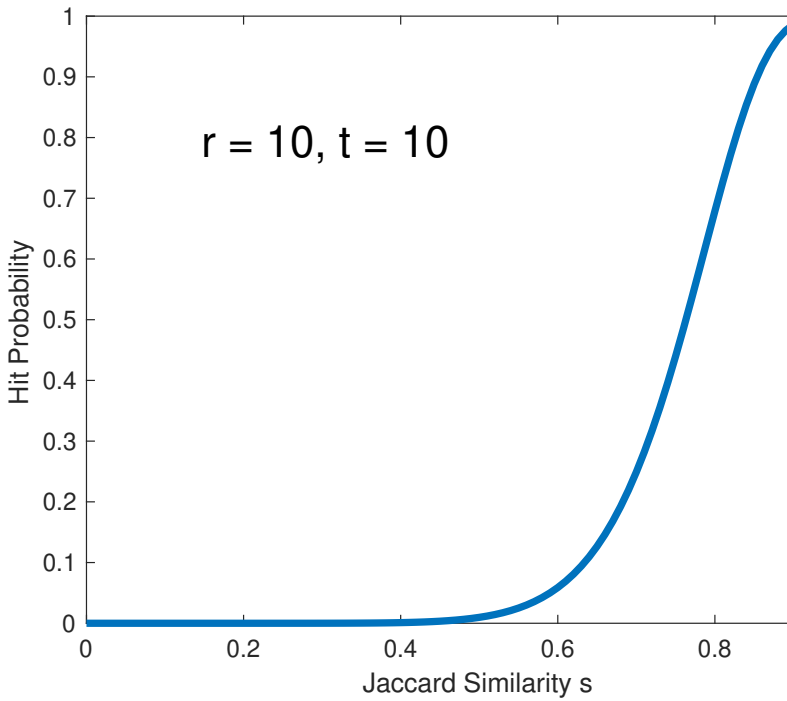
- Probability that a single hash matches.
  $\Pr\left[MH_{i,j}(x) = MH_{i,j}(y)\right] = J(x, y) = s$.
- Probability that $x$ and $y$ having matching signatures in repetition $i$. $\Pr\left[MH_{i,1}(x), \ldots, MH_{i,r}(x) = MH_{i,1}(y), \ldots, MH_{i,r}(y)\right] = s^r$.
- Probability that $x$ and $y$ don't match in repetition $i$: $1 - s^r$.
- Probability that $x$ and $y$ don't match in *all repetitions*: $(1 - s^r)^t$.
- Probability that $x$ and $y$ match in at least one repetition:

Hit Probability: $1 - (1 - s^r)^t$.

# The $s$-curve

Using $t$ repetitions each with a signature of $r$ MinHash values, the probability that $x$ and $y$ with Jaccard similarity $J(x, y) = s$ match in at least one repetition is: $1 - (1 - s^r)^t$.




$r$ and $t$ are tuned depending on application. 'Threshold' when hit probability is 1/2 is $\approx (1/t)^{1/r}$. E.g., $\approx (1/30)^{1/5} = .51$ in this case.

# s-curve Example

**For example:** Consider a database with $10,000,000$ audio clips. You are given a clip $x$ and want to find any $y$ in the database with $J(x,y) \geq .9$.

- There are 10 true matches in the database with $J(x,y) \geq .9$.
- There are $10,000$ near matches with $J(x,y) \in [.7,.9]$.

With signature length $r = 25$ and repetitions $t = 50$, hit probability for $J(x,y) = s$ is $1-(1-s^{25})^{50}$.

- Hit probability for $J(x,y) \geq .9$ is $\geq 1-(1-.9^{25})^{50} \approx .98$
- Hit probability for $J(x,y) \in [.7,.9]$ is $\leq 1-(1-.9^{25})^{50} \approx .98$
- Hit probability for $J(x,y) \leq .7$ is $\leq 1-(1-.7^{25})^{50} \approx .007$

**Expected Number of Items Scanned:** (proportional to query time)

$$\leq 10 + .98 * 10,000 + .007 * 9,989,990 \approx 80,000 \ll 10,000,000.$$

# Compressing High-Dimensional Data and the Johnson-Lindenstrauss Lemma

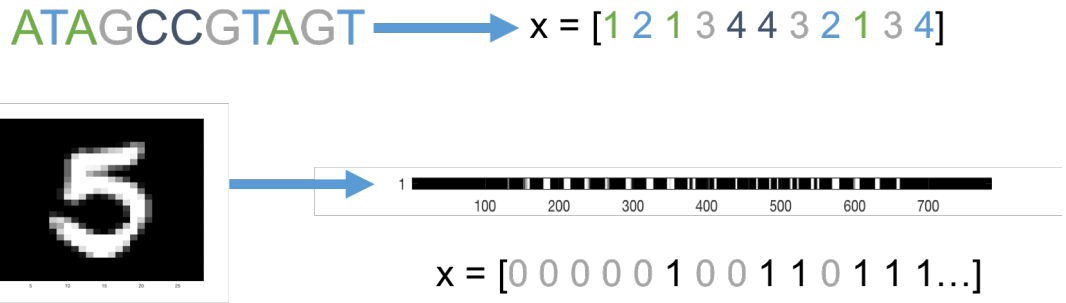
# High Dimensional Data

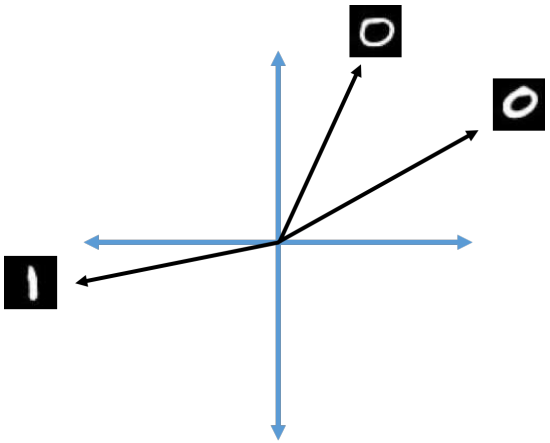
'Big Data' means not just many data points, but many measurements per data point. I.e., very high dimensional data.

- Social media websites record (tens of) thousands of measurements per user: who they follow, who follows them, when they last visited the site, timestamps for specific interactions, how many posts they have made, the content of those posts, etc.
- A 3 minute Youtube clip with a resolution of $500 \times 500$ pixels at 15 frames/second with 3 color channels is a recording of $\geq 2$ billion pixel values. Even a $500 \times 500$ pixel color image has $750,000$ pixel values.
- The human genome contains 3 billion+ base pairs. Genetic datasets often contain information on 100s of thousands+ mutations and genetic markers.

# Data as Vectors and Matrices

In data analysis and machine learning, data points with many attributes are often stored, processed, and interpreted as high dimensional vectors, with real valued entries.



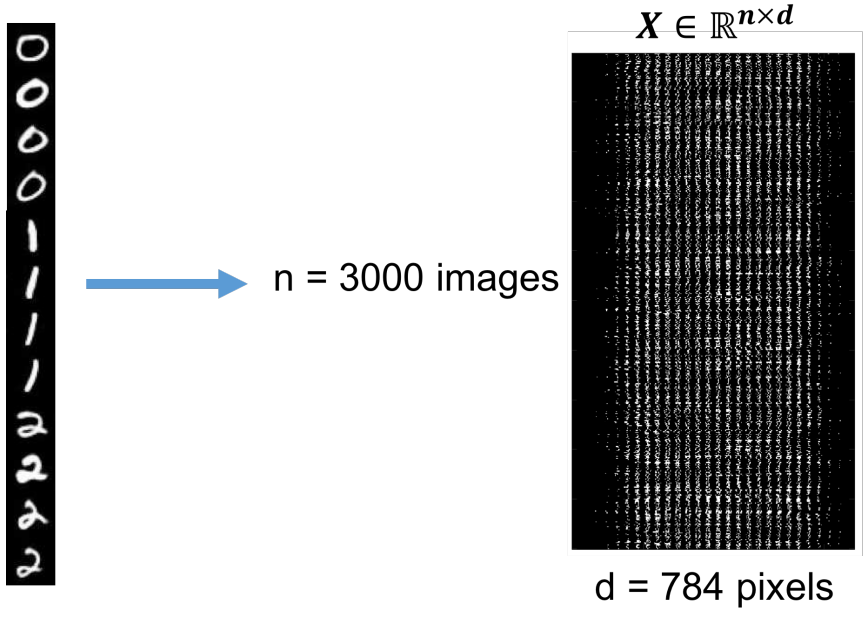
Similarities/distances between vectors (e.g., $\langle x, y \rangle$, $\|x - y\|_2$) have meaning for underlying data points.

# Datasets as Vectors and Matrices

Data points are interpreted as high dimensional vectors, with real valued entries. Data set is interpreted as a matrix.

**Data Points:** $\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_n \in \mathbb{R}^d$.

**Data Set:** $X \in \mathbb{R}^{n \times d}$ with $i^{th}$ row equal to $\vec{x}_i^T$.




Many data points $n \implies$ tall. Many dimensions $d \implies$ wide.

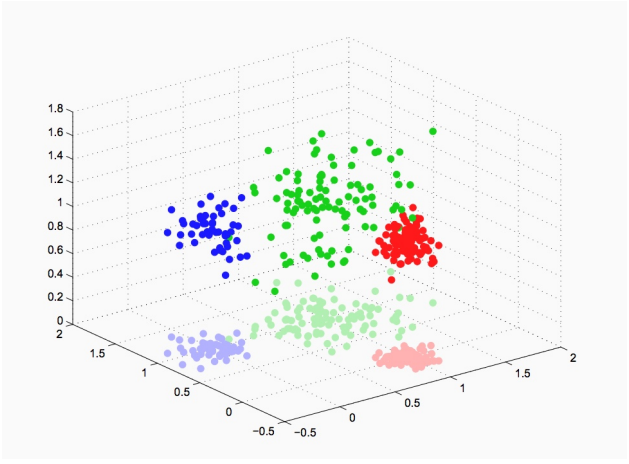
# Dimensionality Reduction

**Dimensionality Reduction:** Compress data points so that they lie in many fewer dimensions.

$$\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{R}^d \to \tilde{x}_1, \ldots, \tilde{x}_n \in \mathbb{R}^m \text{ for } m \ll d.$$




'Lossy compression' that still preserves important information about the relationships between $\vec{x}_1, \ldots, \vec{x}_n$.

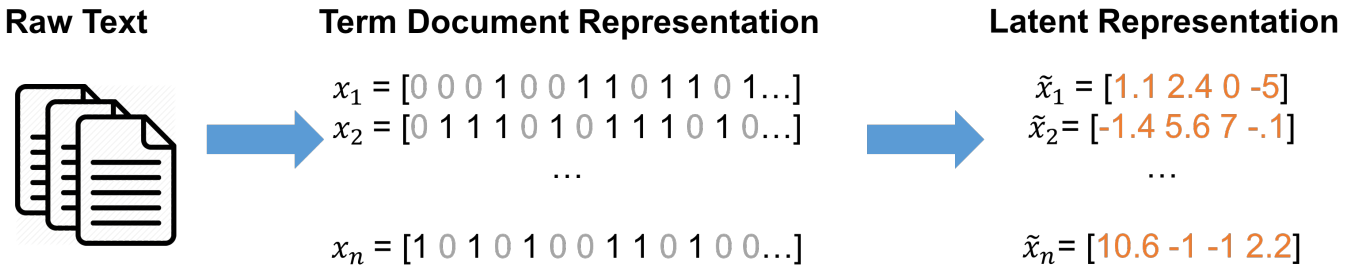


Generally will not consider directly how well $\tilde{x}_i$ approximates $\vec{x}_i$.

# Dimensionality Reduction

Dimensionality reduction is one of the most important techniques in data science. What methods have you heard of?

- Principal component analysis
- Latent semantic analysis (LSA)




- Linear discriminant analysis
- Autoencoders

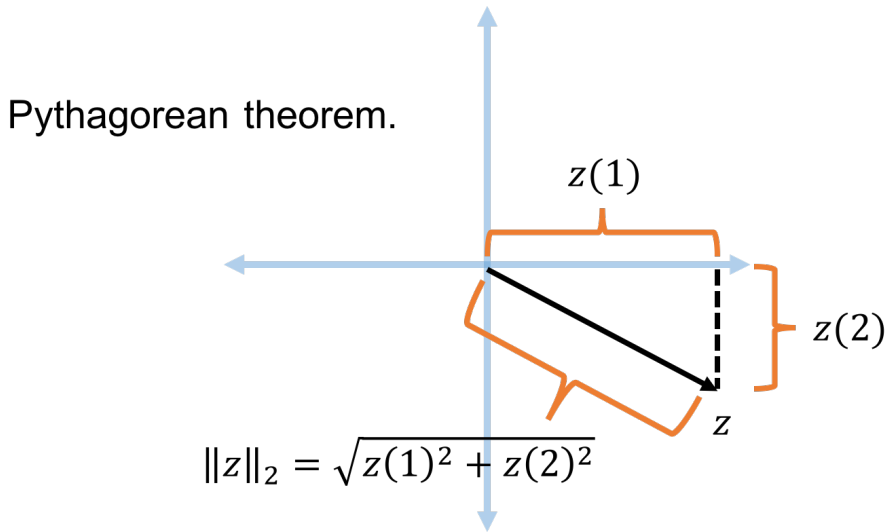
Compressing data makes it more efficient to work with. May also remove extraneous information/noise.

# Embeddings for Euclidean Space

**Euclidean Low Distortion Embedding:** Given $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{R}^d$ and error parameter $\epsilon \geq 0$, find $\tilde{x}_1, \ldots, \tilde{x}_n \in \mathbb{R}^m$ (where $m \ll d$) such that for all $i, j \in [n]$:

$$(1-\epsilon)\|\vec{x}_i - \vec{x}_j\|_2 \leq \|\tilde{x}_i - \tilde{x}_j\|_2 \leq (1+\epsilon)\|\vec{x}_i - \vec{x}_j\|_2.$$

Recall that for $\vec{z} \in \mathbb{R}^n$, $\|\vec{z}\|_2 = \sqrt{\sum_{i=1}^{n} \vec{z}(i)^2}$.




**d-dimensional space**

**m-dimensional space**
(for m << d)

# The Johnson-Lindenstrauss Lemma

The Johnson-Lindenstrauss Lemma tells us that for any set of points $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{R}^d$ and any $\epsilon > 0$, we can find an $\epsilon$-distortion embedding into $m$ dimensions, where $m$ depends only on the error parameter $\epsilon$ and the number of points $n$, but not the original dimension $d$.

# The Johnson-Lindenstrauss Lemma

**Johnson-Lindenstrauss Lemma:** For any set of points $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{R}^d$ and $\epsilon > 0$ there exists a linear map $\mathbf{\Pi} : \mathbb{R}^d \to \mathbb{R}^m$ such that $m = O\left(\frac{\log n}{\epsilon^2}\right)$ and letting $\tilde{x}_i = \mathbf{\Pi}\vec{x}_i$:

$$\text{For all } i, j: \ (1-\epsilon)\|\vec{x}_i - \vec{x}_j\|_2 \leq \|\tilde{x}_i - \tilde{x}_j\|_2 \leq (1+\epsilon)\|\vec{x}_i - \vec{x}_j\|_2.$$

Further, if $\mathbf{\Pi} \in \mathbb{R}^{m \times d}$ has each entry chosen i.i.d. from $\mathcal{N}(0, 1/m)$, it satisfies the guarantee with high probability.

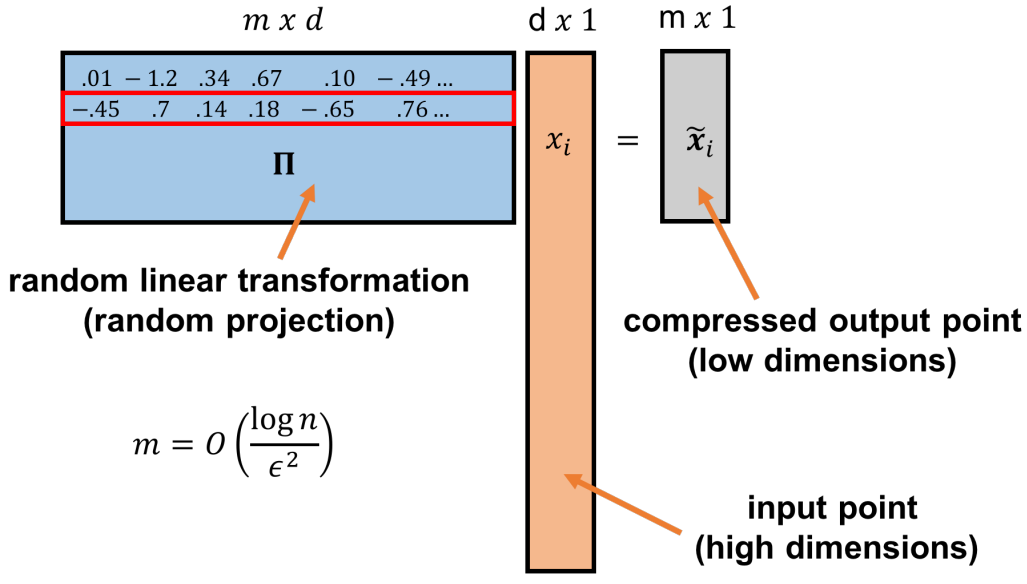
For $d = 1$ trillion, $\epsilon = .05$, and $n = 100,000$, $m \approx 6600$.

Very surprising! Powerful result with a simple construction: applying a random linear transformation to a set of points preserves distances between all those points with high probability.

# Random Projection

For any $\vec{x}_1, \ldots, \vec{x}_n$ and $\mathbf{\Pi} \in \mathbb{R}^{m \times d}$ with each entry chosen i.i.d. from $\mathcal{N}(0, 1/m)$, with high probability, letting $\tilde{\mathbf{x}}_i = \mathbf{\Pi}\vec{x}_i$:

$$\text{For all } i, j: \ (1-\epsilon)\|\vec{x}_i - \vec{x}_j\|_2 \leq \|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j\|_2 \leq (1+\epsilon)\|\vec{x}_i - \vec{x}_j\|_2.$$




- $\mathbf{\Pi}$ is known as a random projection. It is a random linear function, mapping length $d$ vectors to length $m$ vectors.
- $\mathbf{\Pi}$ is data oblivious. Stark contrast to methods like PCA.

# Algorithmic Considerations

- Many alternative constructions: $\pm 1$ entries, sparse (most entries 0), Fourier structured, etc. $\implies$ more efficient computation of $\tilde{\mathbf{x}}_i = \mathbf{\Pi}\vec{x}_i$.
- Data oblivious property means that once $\mathbf{\Pi}$ is chosen, $\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_n$ can be computed in a stream with little memory.
- Memory needed is just $O(d + nm)$ vs. $O(nd)$ to store the full data set.
- Compression can also be easily performed in parallel on different servers.
- When new data points are added, can be easily compressed, without updating existing points.

# Johnson-Lindenstrauss Lemma Proof

# Distributional JL

The Johnson-Lindenstrauss Lemma is a direct consequence of a closely related lemma:

**Distributional JL Lemma:** Let $\mathbf{\Pi} \in \mathbb{R}^{m \times d}$ have each entry chosen i.i.d. as $\mathcal{N}(0, 1/m)$. If we set $m = O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$, then for any $\vec{y} \in \mathbb{R}^d$, with probability $\geq 1 - \delta$

$$(1-\epsilon)\|\vec{y}\|_2 \leq \|\mathbf{\Pi}\vec{y}\|_2 \leq (1+\epsilon)\|\vec{y}\|_2$$

Applying a random matrix $\mathbf{\Pi}$ to any vector $\vec{y}$ preserves $\vec{y}$'s norm with high probability.

- Like a low-distortion embedding, but for the length of a compressed vector rather than distances between vectors.
- Can be proven from first principles.

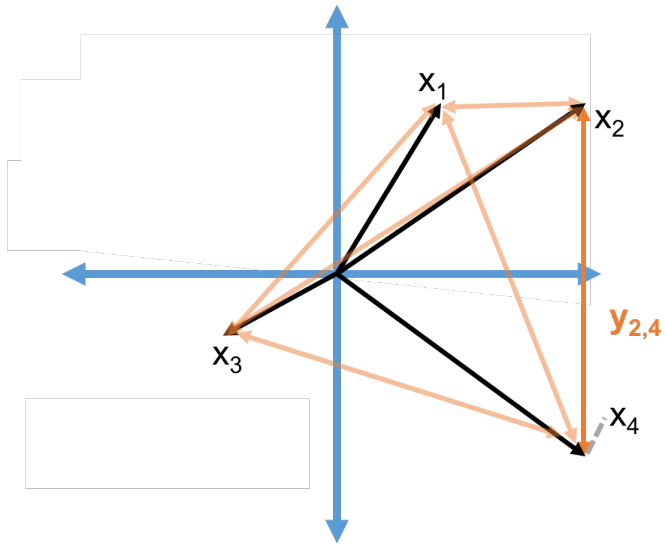
$\mathbf{\Pi} \in \mathbb{R}^{m \times d}$: random projection matrix. $d$: original dimension. $m$: compressed dimension, $\epsilon$: embedding error, $\delta$: embedding failure prob.

# Distributional JL $\implies$ JL

**Distributional JL Lemma $\implies$ JL Lemma:** Distributional JL show that a random projection $\mathbf{\Pi}$ preserves the norm of any $y$. The main JL Lemma says that $\mathbf{\Pi}$ preserves distances between vectors.

Since $\mathbf{\Pi}$ is linear these are the same thing!

**Proof:** Given $\vec{x}_1, \ldots, \vec{x}_n$, define $\binom{n}{2}$ vectors $\vec{y}_{ij}$ where $\vec{y}_{ij} = \vec{x}_i - \vec{x}_j$.




- If we choose $\mathbf{\Pi}$ with $m = O\left(\frac{\log 1/\delta}{\epsilon^2}\right)$, for each $\vec{y}_{ij}$ with probability $\geq 1 - \delta$ we have:

$$(1-\epsilon)\|\vec{y}_{ij}\vec{x}_i - \vec{x}_j\|_2 \leq \|\mathbf{\Pi}\vec{y}_{ij}\mathbf{\Pi}(\vec{x}_i - \vec{x}_j)\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j\|_2 \leq (1+\epsilon)\|\vec{y}_{ij}\vec{x}_i - \vec{x}_j\|_2$$

# Distributional JL $\implies$ JL

**Claim:** If we choose $\mathbf{\Pi}$ with i.i.d. $\mathcal{N}(0, 1/m)$ entries and $m = O\left(\frac{\log(1/\delta')}{\epsilon^2}\right)$, letting $\tilde{\mathbf{x}}_i = \mathbf{\Pi}\vec{x}_i$, for each pair $\vec{x}_i, \vec{x}_j$ with probability $\geq 1 - \delta'$ we have:

$$(1-\epsilon)\|\vec{x}_i - \vec{x}_j\|_2 \leq \|\tilde{\mathbf{x}}_i - \tilde{\mathbf{x}}_j\|_2 \leq (1+\epsilon)\|\vec{x}_i - \vec{x}_j\|_2.$$

With what probability are all pairwise distances preserved?

**Union bound:** With probability $\geq 1 - \binom{n}{2} \cdot \delta'$ all pairwise distances are preserved.

Apply the claim with $\delta' = \delta/\binom{n}{2}$. $\implies$ for $m = O\left(\frac{\log(1/\delta')}{\epsilon^2}\right)$, all pairwise distances are preserved with probability $\geq 1 - \delta$.

$$m = O\left(\frac{\log(1/\delta')}{\epsilon^2}\right) = O\left(\frac{\log(\binom{n}{2}/\delta)}{\epsilon^2}\right) = O\left(\frac{\log(n^2/\delta)}{\epsilon^2}\right) = O\left(\frac{\log(n/\delta)}{\epsilon^2}\right)$$

Yields the JL lemma.