Algorithms for Data Science: Lecture 5

Vasileios Nakos

National Technical University of Athens

April 17, 2021

Dimensionality Reduction

A technique for transforming data from a high-dimensional space into a low-dimensional space so that the low-dimensional representation retains some meaningful properties of the original data.

Dimensionality Reduction

A technique for transforming data from a high-dimensional space into a low-dimensional space so that the low-dimensional representation retains some meaningful properties of the original data.

$\star$ Crucial building block in most Machine Learning applications: if there is a huge number of features then the predictor for the target variable might be prohibitively slow.

Dimensionality Reduction

A technique for transforming data from a high-dimensional space into a low-dimensional space so that the low-dimensional representation retains some meaningful properties of the original data.

⋆ Crucial building block in most Machine Learning applications: if there is a huge number of features then the predictor for the target variable might be prohibitively slow.

High-dimensionality might mean hundreds, thousands, or even millions of input variables.

- Classification, pattern recognition.

Applications

- Classification, pattern recognition.
- Clustering.

Applications

- Classification, pattern recognition.
- Clustering.
- Neural networks.

Applications

- Classification, pattern recognition.
- Clustering.
- Neural networks.
- Neuroscience (maximally informative dimensions)

Applications

- Classification, pattern recognition.
- Clustering.
- Neural networks.
- Neuroscience (maximally informative dimensions)

Given vectors $x_1, x_2, \ldots, x_n \in \mathbb{R}^d$ it is an algorithm $\mathcal{C}$ which transforms those vectors to $y_1, y_2, \ldots, y_n \in \mathbb{R}^m$ for $m \ll d$, such that properties of the initial vectors (dataset) are preserved.

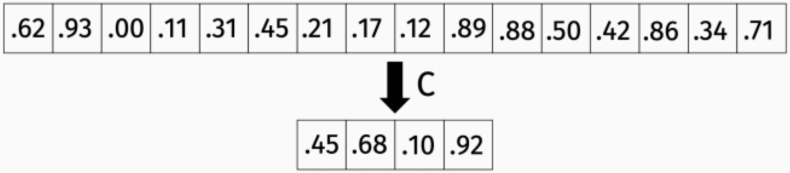
Dimensionality Reduction

Given vectors $x_1, x_2, \dots, x_n \in \mathbb{R}^d$ it is an algorithm $\mathcal{C}$ which transforms those vectors to $y_1, y_2, \dots, y_n \in \mathbb{R}^m$ for $m \ll d$, such that properties of the initial vectors (dataset) are preserved.

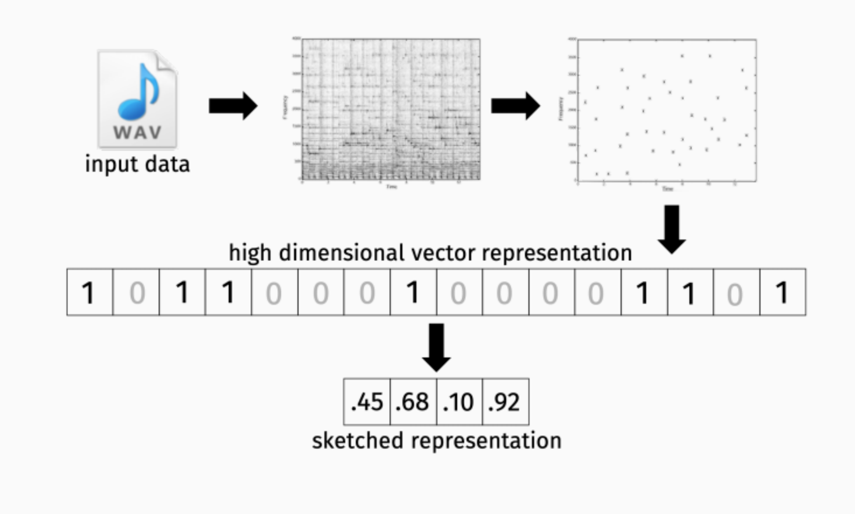
One example being Euclidean or some other distance (very useful in classification tasks).

Given vectors $x_1, x_2, \ldots, x_n \in \mathbb{R}^d$ it is an algorithm $\mathcal{C}$ which transforms those vectors to $y_1, y_2, \ldots, y_n \in \mathbb{R}^m$ for $m \ll d$, such that properties of the initial vectors (dataset) are preserved.

One example being Euclidean or some other distance (very useful in classification tasks).



Music compression and classification



One of the cornerstones of Dimensionality Reduction

The Johnshon-Lindenstrauss (JL) Lemma

Let vectors $x_1, x_2, \dots x_n \in \mathbb{R}^d$. Then there exists a *linear* map $\Pi : \mathbb{R}^d \rightarrow \mathbb{R}^m$, where $m = O(\log n/\epsilon^2)$, such that

$$(1-\epsilon)\|x_i - x_j\|_2 \leq \|\Pi x_i - \Pi x_j\|_2 \leq (1+\epsilon)\|x_i - x_j\|_2.$$

Pairwise distances are approximately preserved by projecting to $O(\log n/\epsilon)^2$ dimensions...How's that even possible?

One of the cornerstones of Dimensionality Reduction

The Johnshon-Lindenstrauss (JL) Lemma, Version 2

Let vectors $x_1, x_2, \dots x_n \in \mathbb{R}^d$. Then there exists a *linear* map $\Pi : \mathbb{R}^d \to \mathbb{R}^m$, where $m = O(\log n/\epsilon^2)$, such that

$$(1-\epsilon)\|x_i - x_j\|_2^2 \leq \|\Pi x_i - \Pi x_j\|_2^2 \leq (1+\epsilon)\|x_i - x_j\|_2^2.$$

One of the cornerstones of Dimensionality Reduction

The Johnshon-Lindenstrauss (JL) Lemma, Version 2

Let vectors $x_1, x_2, \ldots x_n \in \mathbb{R}^d$. Then there exists a *linear* map $\Pi : \mathbb{R}^d \to \mathbb{R}^m$, where $m = O(\log n/\epsilon^2)$, such that

$$(1-\epsilon)\|x_i - x_j\|_2^2 \le \|\Pi x_i - \Pi x_j\|_2^2 \le (1+\epsilon)\|x_i - x_j\|_2^2.$$

since $(1 \pm \epsilon)^2 = 1 \pm 2\epsilon + \epsilon^2 = 1 \pm \Theta(\epsilon)$, for $\epsilon < 1/3$.

Recall that $\langle x, y \rangle = \sum_{i=1}^{d} x_i y_i = \|x\|_2 \cdot \|y\|_2 \cdot \cos(\theta)$, where θ is the angle between x and y.

Recall that $\langle x, y \rangle = \sum_{i=1}^{d} x_i y_i = \|x\|_2 \cdot \|y\|_2 \cdot \cos(\theta)$, where θ is the angle between x and y.

Question: How many pairwise orthogonal unit vectors can we pack in d dimensions?

Recall that $\langle x, y \rangle = \sum_{i=1}^{d} x_i y_i = \|x\|_2 \cdot \|y\|_2 \cdot \cos(\theta)$, where θ is the angle between x and y.

Question: How many pairwise orthogonal unit vectors can we pack in d dimensions?
Answer: d.

Some high-dimensional geometry

Recall that $\langle x, y \rangle = \sum_{i=1}^{d} x_i y_i = \|x\|_2 \cdot \|y\|_2 \cdot \cos(\theta)$, where θ is the angle between x and y.

Question: How many pairwise orthogonal unit vectors can we pack in d dimensions?
Answer: d.

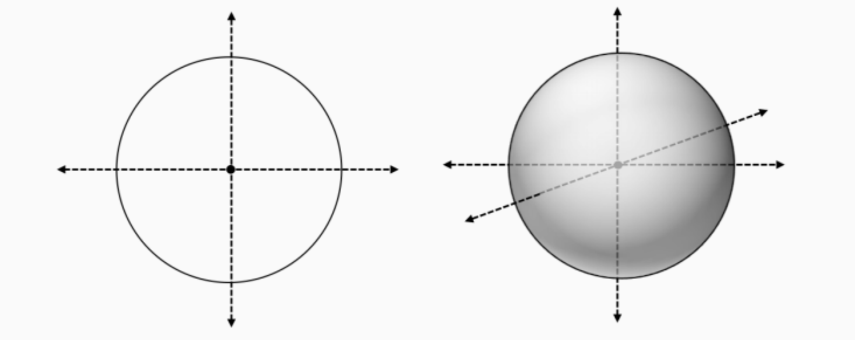
Question: How many pairwise *almost* orthogonal unit vectors (magnitude of inner product $|\langle x, y \rangle| \leq \epsilon$) can we pack in d dimension?

Recall that $\langle x, y \rangle = \sum_{i=1}^{d} x_i y_i = \|x\|_2 \cdot \|y\|_2 \cdot \cos(\theta)$, where θ is the angle between x and y.

Question: How many pairwise orthogonal unit vectors can we pack in d dimensions?
Answer: d.

Question: How many pairwise *almost* orthogonal unit vectors (magnitude of inner product $|\langle x, y \rangle| \leq \epsilon$) can we pack in d dimension?
Answer: $2^{\Theta(\epsilon^2 d)}$.

An ϵ slack gives you exponential space to pack vectors with pairwise inner product at most ϵ.



Method I: Choose $2^{\Theta(\epsilon^2 d)}$ random points on the d-dimensional sphere $\mathcal{B} = \{x \in \mathbb{R}^n : \|x\|_2 = 1\}$. If with some probability > 0 those points have small inner product, then we can infer the existence of such a collection of vectors!

Method I: Choose $2^{\Theta(\epsilon^2 d)}$ random points on the d-dimensional sphere $\mathcal{B} = \{x \in \mathbb{R}^n : \|x\|_2 = 1\}$. If with some probability > 0 those points have small inner product, then we can infer the existence of such a collection of vectors!

Method II (the proof of which we shall see): Choose $2^{\Theta(\epsilon^2 d)}$ vectors such that each coordinates of x equals $\frac{1}{\sqrt{d}}$ with probability $\frac{1}{2}$ and $x_i = -\frac{1}{\sqrt{d}}$ otherwise, i.e. the i-th coordinate is of the form $\frac{\sigma_i}{\sqrt{d}}$ for a random sign σ_i.

Method I: Choose $2^{\Theta(\epsilon^2 d)}$ random points on the d-dimensional sphere $\mathcal{B} = \{x \in \mathbb{R}^n : \|x\|_2 = 1\}$. If with some probability > 0 those points have small inner product, then we can infer the existence of such a collection of vectors!

Method II (the proof of which we shall see): Choose $2^{\Theta(\epsilon^2 d)}$ vectors such that each coordinates of x equals $\frac{1}{\sqrt{d}}$ with probability $\frac{1}{2}$ and $x_i = -\frac{1}{\sqrt{d}}$ otherwise, i.e. the i-th coordinate is of the form $\frac{\sigma_i}{\sqrt{d}}$ for a random sign σ_i.

- $\|x\|_2 = \sqrt{\frac{1}{d} + \ldots + \frac{1}{d}} = 1, \forall x.$

Method I: Choose $2^{\Theta(\epsilon^2 d)}$ random points on the d-dimensional sphere $\mathcal{B} = \{x \in \mathbb{R}^n : \|x\|_2 = 1\}$. If with some probability > 0 those points have small inner product, then we can infer the existence of such a collection of vectors!

Method II (the proof of which we shall see): Choose $2^{\Theta(\epsilon^2 d)}$ vectors such that each coordinates of x equals $\frac{1}{\sqrt{d}}$ with probability $\frac{1}{2}$ and $x_i = -\frac{1}{\sqrt{d}}$ otherwise, i.e. the i-th coordinate is of the form $\frac{\sigma_i}{\sqrt{d}}$ for a random sign σ_i.

- $\|x\|_2 = \sqrt{\frac{1}{d} + \ldots + \frac{1}{d}} = 1, \forall x.$
- $\mathbb{E}(\langle x, y\rangle) = \mathbb{E}(\sum_i (\sigma_i^{(x)})/\sqrt{d} \cdot (\sigma_i^{(x)})/\sqrt{d}) = \sum_i \mathbb{E}(\sigma_i^{(x)}) \cdot \mathbb{E}(\sigma_i^{(x)}) \cdot \frac{1}{d} = 0.$

$$\mathbb{P}\left(|\langle x, y\rangle| > \epsilon\right) = \mathbb{P}(\frac{1}{d}\sum_{i=1}^{d}|\sigma_i^{(x)} \cdot \sigma_i^{(y)}| > \epsilon) =$$

$$\mathbb{P}\left(|\sum_{i=1}^{d}\sigma_i'| > \epsilon \cdot d\right) =$$

$$\mathbb{P}\left(|\sum_{i=1}^{d}\sigma_i' - \mathbb{E}(\sum_{i=1}^{d'}\sigma_i)| > \epsilon \cdot d\right) \leq$$

$$\mathbb{P}_{g\sim\mathcal{N}(0,d)}\left(|g| > \epsilon \cdot d\right) \leq e^{-c\epsilon^2 d^2/d} = e^{-c\epsilon^2 d}.$$

$$\mathbb{P}\left(|\langle x, y\rangle| > \epsilon\right) = \mathbb{P}\left(\frac{1}{d}\sum_{i=1}^{d} |\sigma_i^{(x)} \cdot \sigma_i^{(y)}| > \epsilon\right) =$$

$$\mathbb{P}\left(|\sum_{i=1}^{d} \sigma_i'| > \epsilon \cdot d\right) =$$

$$\mathbb{P}\left(|\sum_{i=1}^{d} \sigma_i' - \mathbb{E}(\sum_{i=1}^{d'} \sigma_i)| > \epsilon \cdot d\right) \leq$$

$$\mathbb{P}_{g\sim\mathcal{N}(0,d)}\left(|g| > \epsilon \cdot d\right) \leq e^{-c\epsilon^2 d^2/d} = e^{-c\epsilon^2 d}.$$

(alternatively, $\sum_{i=1}^{d} \sigma_i'$ can be transformed to a sum of $0 - 1$ random variables and then the Chernoff bound can be applied).

What does this mean?

For random x, y generated by the above process we have that $|\langle x, y\rangle| > \epsilon$ with probability at most $e^{-c\epsilon^2 d}$.

What does this mean?

For random x, y generated by the above process we have that $|\langle x, y \rangle| > \epsilon$ with probability at most $e^{-c\epsilon^2 d}$.
Let $N := 2^{-c\epsilon^2 d/4}$, and for $i, j \in [N]$ let $\mathcal{E}_{ij}$ be the event that the i-th and the j-th vectors created by this process have inner product in magnitude larger than ϵ.

What does this mean?

For random x, y generated by the above process we have that $|\langle x, y\rangle| > \epsilon$ with probability at most $e^{-c\epsilon^2 d}$.
Let $N := 2^{-c\epsilon^2 d/4}$, and for $i, j \in [N]$ let $\mathcal{E}_{ij}$ be the event that the i-th and the j-th vectors created by this process have inner product in magnitude larger than ϵ.

$$\text{union} - \text{bound}$$

$$\mathbb{P}(\exists i, j : \mathcal{E}_{ij}) \leq \sum_{i,j \in [N]} \mathbb{P}(\mathcal{E}_{ij}) \leq \binom{N}{2} e^{-c\epsilon^2 d} \leq e^{-c\epsilon^2 d/2}$$

Thus, with probability $1 - e^{-c\epsilon^2 d/2}$ none of the "bad" events hold, so all the pairwise inner products are small!

Back to Dimensionality Reduction

In fact, the abundance of pairwise almost orthogonal vectors is very tied to the Johnson-Lindenstrauss Lemma!

In fact, the abundance of pairwise almost orthogonal vectors is very tied to the Johnson-Lindenstrauss Lemma!

The Johnshon-Lindenstrauss (JL) Lemma

Let vectors $x_1, x_2, \ldots x_n \in \mathbb{R}^d$. Then there exists a *linear* map $\Pi : \mathbb{R}^d \to \mathbb{R}^m$, where $m = O(\log n/\epsilon^2)$, such that

$$(1-\epsilon)\|x_i - x_j\|_2 \leq \|\Pi x_i - \Pi x_j\|_2 \leq (1+\epsilon)\|x_i - x_j\|_2.$$

If we apply JL on vectors $\{0, e_1, e_2, \ldots, e_N\}$, then the obtained vectors $y_1, y_2, \ldots, y_{N+1}$ live in dimension $m = O(\log N/\epsilon^2)$ and have pairwise product at most ϵ; thus they are an enormous collection of pairwise almost orthogonal vectors in dimension m.

Construction of Johnson-Lindenstauss embeddings

All constructions are oblivious to the dataset, i.e. do not even need to look at $x_1, x_2, \ldots$

- (Dense Gaussian matrix) $\Pi_{ij} \sim \mathcal{N}(0, \frac{1}{m})$.

Construction of Johnson-Lindenstauss embeddings

All constructions are oblivious to the dataset, i.e. do not even need to look at $x_1, x_2, \ldots$

- (Dense Gaussian matrix) $\Pi_{ij} \sim \mathcal{N}(0, \frac{1}{m})$.
- (Dense random sign matrix) $\Pi_{ij} \sim \frac{\sigma_{ij}}{\sqrt{m}}$.

Construction of Johnson-Lindenstauss embeddings

All constructions are oblivious to the dataset, i.e. do not even need to look at $x_1, x_2, \ldots$

- (Dense Gaussian matrix) $\Pi_{ij} \sim \mathcal{N}(0, \frac{1}{m})$.
- (Dense random sign matrix) $\Pi_{ij} \sim \frac{\sigma_{ij}}{\sqrt{m}}$.
- (Achlioptas sign matrix, 2001, implemented in Matlab) Only $1/3$ of the matrix is non-zero and the non-zero (i, j) entries satisfy $\Pi_{ij} \sim \frac{\sigma_i}{\sqrt{m}}$

Construction of Johnson-Lindenstauss embeddings

All constructions are oblivious to the dataset, i.e. do not even need to look at $x_1, x_2, \dots$

- (Dense Gaussian matrix) $\Pi_{ij} \sim \mathcal{N}(0, \frac{1}{m})$.
- (Dense random sign matrix) $\Pi_{ij} \sim \frac{\sigma_{ij}}{\sqrt{m}}$.
- (Achlioptas sign matrix, 2001, implemented in Matlab) Only $1/3$ of the matrix is non-zero and the non-zero (i, j) entries satisfy $\Pi_{ij} \sim \frac{\sigma_i}{\sqrt{m}}$
- (Sparse JL, Nelson-Kane) Each column in Π has exactly $s = O(\epsilon^{-1} \log n)$ non-zeros, and those are $\frac{\sigma_{ij}}{\sqrt{s}}$.

Construction of Johnson-Lindenstauss embeddings

All constructions are oblivious to the dataset, i.e. do not even need to look at $x_1, x_2, \ldots$

- (Dense Gaussian matrix) $\Pi_{ij} \sim \mathcal{N}(0, \frac{1}{m})$.
- (Dense random sign matrix) $\Pi_{ij} \sim \frac{\sigma_{ij}}{\sqrt{m}}$.
- (Achlioptas sign matrix, 2001, implemented in Matlab) Only 1/3 of the matrix is non-zero and the non-zero (i, j) entries satisfy $\Pi_{ij} \sim \frac{\sigma_i}{\sqrt{m}}$
- (Sparse JL, Nelson-Kane) Each column in Π has exactly $s = O(\epsilon^{-1} \log n)$ non-zeros, and those are $\frac{\sigma_{ij}}{\sqrt{s}}$.
- (Ailon-Chazelle) $\Pi = PFD$, where D is a diagonal matrix with random signs, F is the Discrete Fourier transform, and P is a matrix with only one non-zero per column.

The Distributional JL Lemma

DJL

There exist distributions over matrix $\Pi \in \mathbb{R}^{m \times n}$, where $m = O(\epsilon^{-2} \log(1/\delta))$ such that

$$\forall x \in \mathbb{R}^n : \mathbb{P}(\|\Pi x\|_2^2 \notin [1 - \epsilon, 1 + \epsilon] \cdot \|x\|_2^2) \leq \delta.$$

DJL

There exist distributions over matrix $\Pi \in \mathbb{R}^{m \times n}$, where $m = O(\epsilon^{-2} \log(1/\delta))$ such that

$$\forall x \in \mathbb{R}^n : \mathbb{P}(\|\Pi x\|_2^2 \notin [1-\epsilon, 1+\epsilon] \cdot \|x\|_2^2) \leq \delta.$$

From DJL one can obtain JL by setting $\delta = \frac{1}{2\binom{n}{2}}$ and applying the union-bound (exercise!).

DJL

There exist distributions over matrix $\Pi \in \mathbb{R}^{m\times n}$, where $m = O(\epsilon^{-2}\log(1/\delta))$ such that

$$\forall x \in \mathbb{R}^n : \mathbb{P}(\|\Pi x\|_2^2 \notin [1-\epsilon, 1+\epsilon] \cdot \|x\|_2^2) \leq \delta.$$

From DJL one can obtain JL by setting $\delta = \frac{1}{2\binom{n}{2}}$ and applying the union-bound (exercise!).In the previous constructions you may think of n as $\approx \frac{1}{\delta}$.

Dense Gaussian Matrix $\Pi_{ij} \sim \mathcal{N}(0, \frac{1}{m})$

The *i*-th entry of the low-dimensional version is

$$\sum_{i=1}^{n} \Pi_{ij} \cdot x_j \sim \mathcal{N}(0, \frac{1}{m}\|x\|_2^2).$$

Dense Gaussian Matrix $\Pi_{ij} \sim \mathcal{N}(0, \frac{1}{m})$

The *i*-th entry of the low-dimensional version is

$$\sum_{i=1}^{n} \Pi_{ij} \cdot x_j \sim \mathcal{N}(0, \frac{1}{m} \|x\|_2^2).$$

2-stability of gausians: $\mathcal{N}(0, \sigma^2) + \mathcal{N}(0, \tau^2) \sim \mathcal{N}(0, \sigma^2 + \tau^2)$

Dense Gaussian Matrix $\Pi_{ij} \sim \mathcal{N}(0, \frac{1}{m})$

The *i*-th entry of the low-dimensional version is

$$\sum_{i=1}^{n} \Pi_{ij} \cdot x_j \sim \mathcal{N}(0, \frac{1}{m}\|x\|_2^2).$$

2-stability of gausians: $\mathcal{N}(0, \sigma^2) + \mathcal{N}(0, \tau^2) \sim \mathcal{N}(0, \sigma^2 + \tau^2)$

$$Pr_{g \sim \mathcal{N}(0,\sigma^2)}(|g| \geq \lambda) \leq e^{-c \cdot \frac{\lambda^2}{\sigma^2}}.$$

Dense Gaussian Matrix $\Pi_{ij} \sim \mathcal{N}(0, \frac{1}{m})$

The *i*-th entry of the low-dimensional version is

$$\sum_{i=1}^{n} \Pi_{ij} \cdot x_j \sim \mathcal{N}(0, \frac{1}{m}\|x\|_2^2).$$

2-stability of gausians: $\mathcal{N}(0, \sigma^2) + \mathcal{N}(0, \tau^2) \sim \mathcal{N}(0, \sigma^2 + \tau^2)$

$$Pr_{g \sim \mathcal{N}(0,\sigma^2)}(|g| \geq \lambda) \leq e^{-c \cdot \frac{\lambda^2}{\sigma^2}}.$$

In other words, if we want failure probability δ we can ensure that $|g|$ can be at most $O(\sqrt{\log(1/\delta)} \cdot \sigma)$.

Dense Gaussian Matrix $\Pi_{ij} \sim \mathcal{N}(0, \frac{1}{m})$

The *i*-th entry of the low-dimensional version is

$$\sum_{i=1}^{n} \Pi_{ij} \cdot x_j \sim \mathcal{N}(0, \frac{1}{m} \|x\|_2^2).$$

Dense Gaussian Matrix $\Pi_{ij} \sim \mathcal{N}(0, \frac{1}{m})$

The i-th entry of the low-dimensional version is

$$\sum_{i=1}^{n} \Pi_{ij} \cdot x_j \sim \mathcal{N}(0, \frac{1}{m}\|x\|_2^2).$$

Thus,

$$\|\Pi x\|_2^2 = \sum_{i=1}^{m} g_i^2, \quad g_i \sim N(0, \frac{1}{m}\|x\|_2^2)$$

$$\mathbb{E}(\|\Pi x\|_2^2) = \mathbb{E}(\sum_{i=1}^{m} g_i^2) = \sum_{i=1}^{m} \mathbb{E}(g_i^2) =$$

$$m \cdot \frac{1}{m}\|x\|_2^2 = \|x\|_2^2.$$

Dense Gaussian Matrix $\Pi_{ij} \sim \mathcal{N}(0, \frac{1}{m})$

The i-th entry of the low-dimensional version is

$$\sum_{i=1}^{n} \Pi_{ij} \cdot x_j \sim \mathcal{N}(0, \frac{1}{m}\|x\|_2^2).$$

Thus,

$$\|\Pi x\|_2^2 = \sum_{i=1}^{m} g_i^2, \quad g_i \sim N(0, \frac{1}{m}\|x\|_2^2)$$

$$\mathbb{E}(\|\Pi x\|_2^2) = \mathbb{E}(\sum_{i=1}^{m} g_i^2) = \sum_{i=1}^{m} \mathbb{E}(g_i^2) =$$

$$m \cdot \frac{1}{m}\|x\|_2^2 = \|x\|_2^2.$$

$$Pr(|\|\Pi x\|_2^2 - \mathbb{E}(\|\Pi x\|_2^2)| \geq \epsilon \cdot \|x\|_2^2) =?$$

We need

$$Pr(|\|\Pi x\|_2^2 - \mathbb{E}(\|\Pi x\|_2^2)| \geq \epsilon \cdot \|x\|_2^2) \leq \delta.$$

We need

$$Pr(|\|\Pi x\|_2^2 - \mathbb{E}(\|\Pi x\|_2^2)| \geq \epsilon \cdot \|x\|_2^2) \leq \delta.$$

Recall that $\|\Pi x\|_2^2 = \sum_{i=1}^{m} g_i^2, g_i \sim N(0, \frac{1}{m}\|x\|_2^2)$ is a sum of squared normal random variables, following a χ^2 distribution.

We need

$$Pr(|\|\Pi x\|_2^2 - \mathbb{E}(\|\Pi x\|_2^2)| \geq \epsilon \cdot \|x\|_2^2) \leq \delta.$$

Recall that $\|\Pi x\|_2^2 = \sum_{i=1}^m g_i^2, g_i \sim N(0, \frac{1}{m}\|x\|_2^2)$ is a sum of squared normal random variables, following a χ^2 distribution.

It can be proved using the same technique as in the Chernoff bound proof (exercise) that if we pick $m = O(\epsilon^{-1} \log(1/\delta))$ the desired inequality holds, hence the DJL Lemma.

Johnson-Lindenstrauss and Python libraries

```
>>> import numpy as np
>>> from sklearn import random_projection
>>> X = np.random.rand(100, 10000)
>>> transformer = random_projection.GaussianRandomProjection()
>>> X_new = transformer.fit_transform(X)
>>> X_new.shape
(100, 3947)
```

```
>>> from sklearn.random_projection import johnson_lindenstrauss_min_dim
>>> johnson_lindenstrauss_min_dim(n_samples=1e6, eps=0.5)
663
>>> johnson_lindenstrauss_min_dim(n_samples=1e6, eps=[0.5, 0.1, 0.01])
array([    663,   11841, 1112658])
>>> johnson_lindenstrauss_min_dim(n_samples=[1e4, 1e5, 1e6], eps=0.1)
array([ 7894,  9868, 11841])
```

We’ve only scratched the surface of dimensionality reduction.
Thousands of papers and work on the topic the past decade.

Thank you!