COMPSCI 514: ALGORITHMS FOR DATA SCIENCE

Cameron Musco

University of Massachusetts Amherst. Fall 2021.

Lecture 11

- Problem Set 2 is due next Friday 10/15.
- Midterm is in class on Tuesday, 10/19.
- I have posted a study guide and practice questions on the course schedule.

Last Class:

- Introduced the k-frequent elements problem – identify all elements of a stream of n elements that occur $\geq n/k$ times.
- Saw how to solve approximately in $O(k \log n/\epsilon)$ space using the Count-min sketch algorithm.
- Simple analysis based on Markov's inequality and repeated random hashing.

This Class:

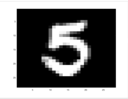
- Randomized methods for dimensionality reduction.
- The Johnson-Lindenstrauss Lemma.

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

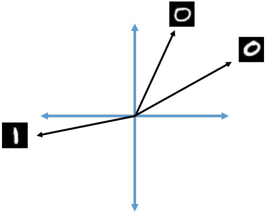
- Twitter has 321 million active monthly users. Records (tens of) thousands of measurements per user: who they follow, who follows them, when they last visited the site, timestamps for specific interactions, how many tweets they have sent, the text of those tweets, etc.
- A 3 minute Youtube clip with a resolution of 500×500 pixels at 15 frames/second with 3 color channels is a recording of ≥ 2 billion pixel values. Even a 500×500 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.

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.

ATAGCCGTAGT → x = [1 2 1 3 4 4 3 2 1 3 4]



x = [0 0 0 0 0 1 0 0 1 1 0 1 1 1...]



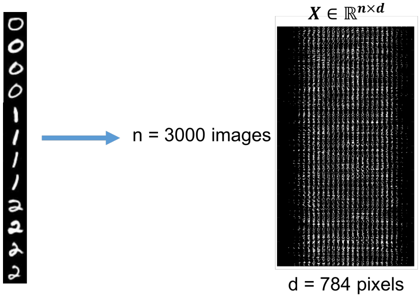
Similarities/distances between vectors (e.g., $\langle x, y \rangle$, $\|x - y\|_2$) have meaning for underlying data points.

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 $\vec{x}_i$.



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

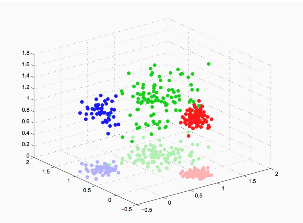
DIMENSIONALITY REDUCTION

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

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

x = [0 0 0 0 0 1 0 0 1 1 0 1 1 1...] → $\tilde{x}$ = [-5.5 4 3.2 -1]

'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)

Raw Text | **Term Document Representation** | **Latent Representation**

x_1 = [0 0 0 1 0 0 1 1 0 1 1 0 1...]
x_2 = [0 1 1 1 0 1 0 1 1 1 0 1 0...]
...
x_n = [1 0 1 0 1 0 0 1 1 0 1 0 0...]

$\tilde{x}_1$ = [1.1 2.4 0 -5]
$\tilde{x}_2$= [-1.4 5.6 7 -.1]
...
$\tilde{x}_n$= [10.6 -1 -1 2.2]

- Linear discriminant analysis
- Autoencoders

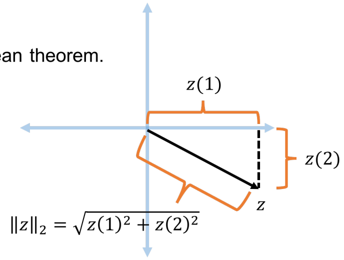
Compressing data makes it more efficient to work with. May also remove extraneous information/noise.

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

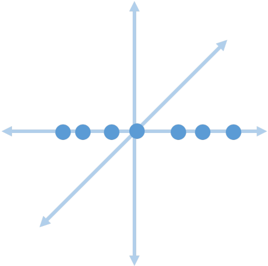
Pythagorean theorem.



d-dimensional space

m-dimensional space
(for m << d)

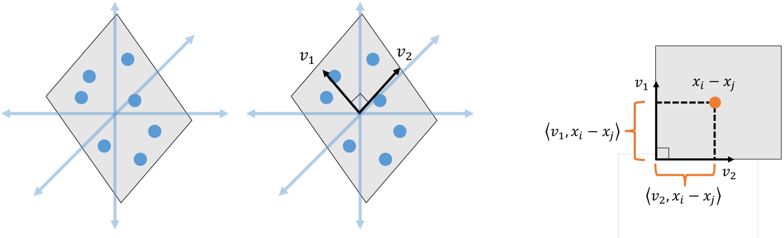
A very easy case: Assume that $\vec{x}_1, \ldots, \vec{x}_n$ all lie on the 1^{st} axis in $\mathbb{R}^d$.



Set $m = 1$ and $\tilde{x}_i = [\vec{x}_i(1)]$ (i.e., $\tilde{x}_i$ contains just a single number).

- $\|\tilde{x}_i - \tilde{x}_j\|_2 = \sqrt{[\vec{x}_i(1) - \vec{x}_j(1)]^2} = |\vec{x}_i(1) - \vec{x}_j(1)| = \|\vec{x}_i - \vec{x}_j\|_2$.
- An embedding with no distortion from any d into $m = 1$.

Assume that $\vec{x}_1, \ldots \vec{x}_n$ lie in any k-dimensional subspace $\mathcal{V}$ of $\mathbb{R}^d$.



- Let $\vec{v}_1, \vec{v}_2, \ldots \vec{v}_k$ be an orthonormal basis for $\mathcal{V}$ and let $\mathbf{V} \in \mathbb{R}^{d \times k}$ be the matrix with these vectors as its columns.
- For all i, j we have $\vec{x}_i - \vec{x}_j \in \mathcal{V}$ and (a good exercise!):

$$\|\vec{x}_i - \vec{x}_j\|_2 = \sqrt{\sum_{\ell=1}^{k} \langle v_\ell, \vec{x}_i - \vec{x}_j \rangle^2} = \|\mathbf{V}^T(\vec{x}_i - \vec{x}_j)\|_2.$$

- If we set $\tilde{x}_i \in \mathbb{R}^k$ to $\tilde{x}_i = \mathbf{V}^T \vec{x}_i$ we have:

What about when we don't make any assumptions on $\vec{x}_1, \ldots, \vec{x}_n$. I.e., they can be scattered arbitrarily around d-dimensional space?

- Can we find a no-distortion embedding into $m \ll d$ dimensions? No. Require $m = d$.
- Can we find an ϵ-distortion embedding into $m \ll d$ dimensions for $\epsilon > 0$? Yes! Always, with m depending on ϵ.

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

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 $\tilde{x}_i = \mathbf{\Pi}\vec{x}_i$:

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

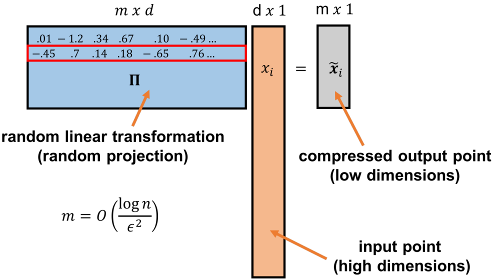
Further, if $\mathbf{\Pi} \in \mathbb{R}^{m \times d}$ has each entry chosen i.i.d. from $\mathcal{N}(0, 1/m)$, it satisfies the guarantee with high probability.

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.

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.

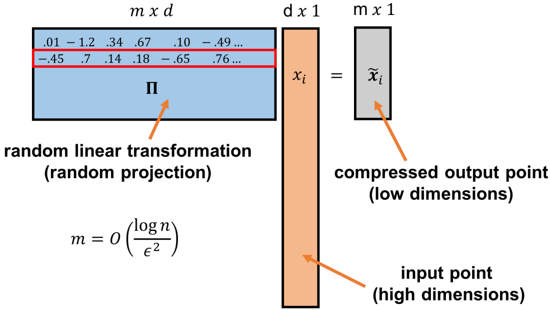
- Many alternative constructions: ± 1 entries, sparse (most entries 0), Fourier structured, etc. $\implies$ more efficient computation of $\tilde{\mathbf{x}}_i = \mathbf{\Pi}\vec{x}_i$.
- Data oblivious property means that once $\mathbf{\Pi}$ is chosen, $\tilde{\mathbf{x}}_1, \ldots, \tilde{\mathbf{x}}_n$ 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.

CONNECTION TO SIMHASH

Compression operation is $\tilde{\mathbf{x}}_i = \mathbf{\Pi}\vec{x}_i$, so for any j,

$$\tilde{\mathbf{x}}_i(j) = \langle \mathbf{\Pi}(j), \vec{x}_i \rangle = \sum_{k=1}^{d} \mathbf{\Pi}(j,k) \cdot \vec{x}_i(k).$$

$\mathbf{\Pi}(j)$ is a vector with independent random Gaussian entries.



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.

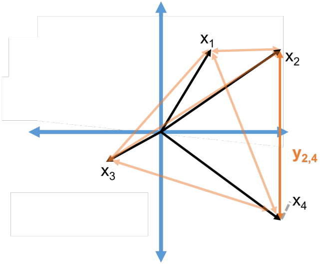
$\mathbf{\Pi} \in \mathbb{R}^{m \times d}$: random projection matrix. d: original dimension. m: compressed dimension, ϵ: embedding error, δ: embedding failure prob.

Questions?

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:

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.

Questions?