COMPSCI 514: ALGORITHMS FOR DATA SCIENCE

Cameron Musco

University of Massachusetts Amherst. Spring 2020.

Lecture 11

LOGISTICS

- Problem Set 2 is due thus upcoming Sunday 3/8.
- Midterm is next Thursday, 3/12. See webpage for study guide/practice questions.
- Let me know ASAP if you need accommodations (e.g., extended time).
- My office hours next Tuesday will focus on exam review. I will hold them at the usual time, and before class at 10:15am.
- I am rearranging the next two lectures to spend more time on the JL Lemma and randomized methods, before moving on the spectral methods (PCA, spectral clustering, etc.)

Thanks for you feedback! Some specifics:

- More details in proofs and slower pace. Will try to find a balance with this.
- Recap at the end of class.
- I will post 'compressed' versions of the slides. Not perfect, but looking into ways to improve.
- After the midterm, I might split the homework assignments into more smaller assignments to spread out the work more.

Last Class: The Johnson-Lindenstrauss Lemma

- Low-distortion embeddings for any set of points via random projection.
- Started on proof of the JL Lemma via the Distributional JL Lemma.

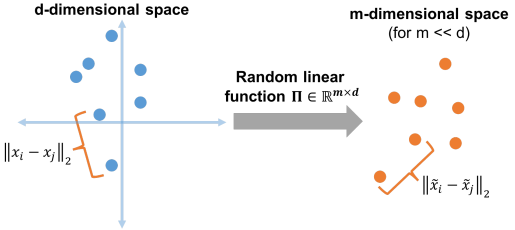
This Class:

- Finish Up proof of the JL lemma.
- Example applications to classification and clustering.
- Discuss connections to high dimensional geometry.

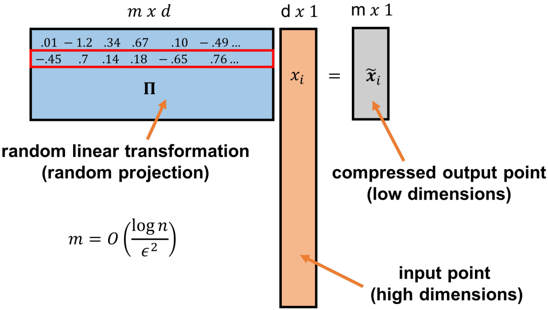
Johnson-Lindenstrauss Lemma: For any set of points $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{R}^d$ and $\epsilon > 0$ there exists a linear map $\mathbf{\Pi} : \mathbb{R}^d \to R^m$ such that $m = O\left(\frac{\log n}{\epsilon^2}\right)$ and letting $\tilde{x}_i = \mathbf{\Pi}\vec{x}_i$:

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

Further, if $\mathbf{\Pi} \in \mathbb{R}^{m \times d}$ has each entry chosen i.i.d. from $\mathcal{N}(0, 1/m)$ and $m = O\left(\frac{\log n/\delta}{\epsilon^2}\right)$, $\mathbf{\Pi}$ satisfies the guarantee with probability $\geq 1 - \delta$.



RANDOM PROJECTION



- Can store $\tilde{x}_1, \ldots, \tilde{x}_n$ in $n \cdot m$ rather than $n \cdot d$ space. What about $\mathbf{\Pi}$?
- Often don't need to store explicitly – compute it on the fly.
- `For` $i = 1 \ldots d$:
  - $\tilde{x}_j := \tilde{x}_j + \mathbf{h}(i) \cdot x_j(i)$

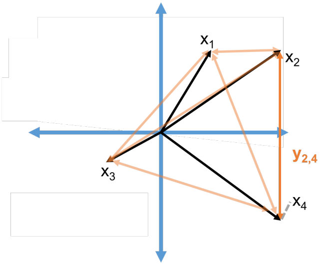
  where $\mathbf{h} : [d] \to \mathbb{R}^m$ is a random hash function outputting vectors (the columns of $\mathbf{\Pi}$).

We showed that the Johnson-Lindenstrauss Lemma follows from:

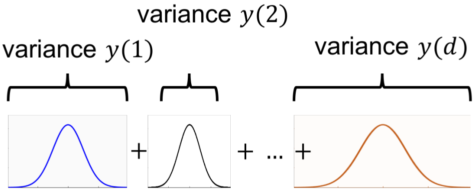
Distributional JL Lemma: Let $\mathbf{\Pi} \in \mathbb{R}^{m \times d}$ have each entry chosen i.i.d. as $\mathcal{N}(0, 1/m)$. If we set $m = O\left(\frac{\log(1/\delta)}{\epsilon^2}\right)$, then for any $\vec{y} \in \mathbb{R}^d$, with probability $\geq 1 - \delta$

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

Main Idea: Union bound over $\binom{n}{2}$ difference vectors $\vec{y}_{ij} = \vec{x}_i - \vec{x}_j$.



- Let $\tilde{\mathbf{y}}$ denote $\mathbf{\Pi}\vec{y}$ and let $\mathbf{\Pi}(j)$ denote the j^{th} row of $\mathbf{\Pi}$.
- For any j, $\tilde{\mathbf{y}}(j) = \langle \mathbf{\Pi}(j), \vec{y} \rangle = \frac{1}{\sqrt{m}} \sum_{i=1}^{d} \mathbf{g}_i \cdot \vec{y}(i)$ where $\mathbf{g}_i \sim \mathcal{N}(0,1)$.
- $\mathbf{g}_i \cdot \vec{y}(i) \sim \mathcal{N}(0, \vec{y}(i)^2)$: a normal distribution with variance $\vec{y}(i)^2$.



$$\tilde{\boldsymbol{y}}(j) = \frac{1}{\sqrt{m}}[\boldsymbol{g}_1 \cdot y(1) + \boldsymbol{g}_2 \cdot y(2) + \dots + \boldsymbol{g}_n \cdot y(d)]$$

$\tilde{\mathbf{y}}(j)$ is also Gaussian, with $\tilde{y}(j) \sim \mathcal{N}(0, \|\vec{y}\|_2^2/m)$.

$\vec{y} \in \mathbb{R}^d$: arbitrary vector, $\tilde{\mathbf{y}} \in \mathbb{R}^m$: compressed vector, $\mathbf{\Pi} \in \mathbb{R}^{m \times d}$: random projection mapping $\vec{y} \to \tilde{\mathbf{y}}$. $\mathbf{\Pi}(j)$: j^{th} row of $\mathbf{\Pi}$, d: original dimension. m: compressed dimension, $\mathbf{g}_i$: normally distributed random variable.

Up Shot: Each entry of our compressed vector $\tilde{\mathbf{y}}$ is Gaussian:

$$\tilde{\mathbf{y}}(j) \sim \mathcal{N}(0, \|\vec{y}\|_2^2/m).$$

$$\mathbb{E}[\|\tilde{\mathbf{y}}\|_2^2] = \mathbb{E}\left[\sum_{j=1}^m \tilde{\mathbf{y}}(j)^2\right] = \sum_{j=1}^m \mathbb{E}[\tilde{\mathbf{y}}(j)^2]$$
$$= \sum_{j=1}^m \frac{\|\vec{y}\|_2^2}{m} = \|\vec{y}\|_2^2$$

So $\tilde{\mathbf{y}}$ has the right norm in expectation.

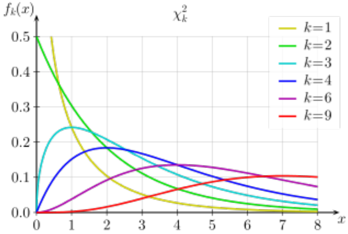
How is $\|\tilde{\mathbf{y}}\|_2^2$ distributed? Does it concentrate?

$\vec{y} \in \mathbb{R}^d$: arbitrary vector, $\tilde{\mathbf{y}} \in \mathbb{R}^m$: compressed vector, $\mathbf{\Pi} \in \mathbb{R}^{m\times d}$: random projection mapping $\vec{y} \to \tilde{\mathbf{y}}$. d: original dimension. m: compressed dimension, $\mathbf{g}_i$: normally distributed random variable

So Far: Each entry of our compressed vector $\tilde{\mathbf{y}}$ is Gaussian with :

$$\tilde{\mathbf{y}}(j) \sim \mathcal{N}(0, \|\vec{y}\|_2^2/m) \text{ and } \mathbb{E}[\|\tilde{\mathbf{y}}\|_2^2] = \|\vec{y}\|_2^2$$

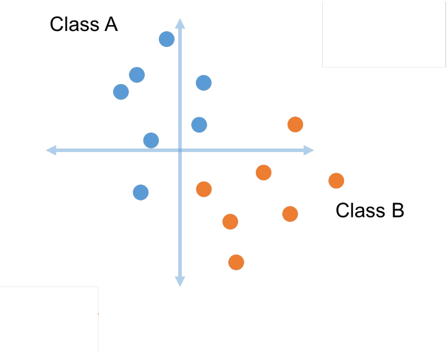
$\|\tilde{\mathbf{y}}\|_2^2 = \sum_{i=1}^m \tilde{\mathbf{y}}(j)^2$ a Chi-Squared random variable with m degrees of freedom (a sum of m squared independent Gaussians)

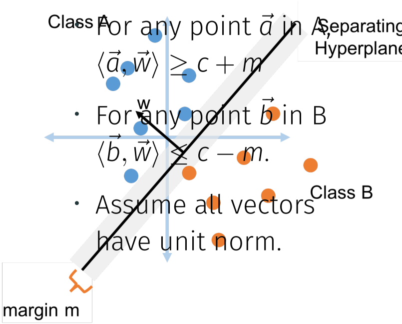


Lemma: (Chi-Squared Concentration) Letting $\mathbf{Z}$ be a Chi-Squared random variable with m degrees of freedom,

$$\Pr\left[|\mathbf{Z} - \mathbb{E}\mathbf{Z}| \geq \epsilon \mathbb{E}\mathbf{Z}\right] \leq 2e^{-m\epsilon^2/8}.$$

Support Vector Machines: A classic ML algorithm, where data is classified with a hyperplane.





- For any point $\vec{a}$ in A, $\langle \vec{a}, \vec{w} \rangle \geq c + m$
- For any point $\vec{b}$ in B $\langle \vec{b}, \vec{w} \rangle \leq c - m$.
- Assume all vectors have unit norm.

JL Lemma implies that after projection into $O\left(\frac{\log n}{m^2}\right)$ dimensions, still have $\langle \tilde{\mathbf{a}}, \tilde{\mathbf{w}} \rangle \geq c + m/2$ and $\langle \tilde{\mathbf{b}}, \tilde{\mathbf{w}} \rangle \leq c - m/2$.

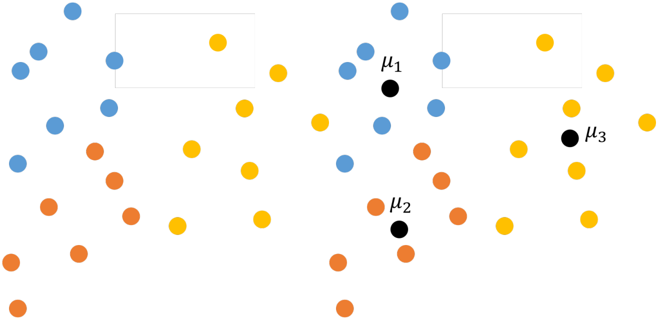
Upshot: Can random project and run SVM (much more efficiently) in the lower dimensional space to find separator $\tilde{\mathbf{w}}$.

Claim: After random projection into $O\left(\frac{\log n}{m^2}\right)$ dimensions, if $\langle \vec{a}, \vec{w} \rangle \geq c + m \geq 0$ then $\langle \tilde{\mathbf{a}}, \tilde{\mathbf{w}} \rangle \geq c + m/2$.

By JL Lemma: applied with $\epsilon = m/4$,

$$\begin{aligned}
\|\tilde{\mathbf{a}} - \tilde{\mathbf{w}}\|_2^2 &\leq \left(1 + \frac{m}{4}\right) \|\vec{a} - \vec{w}\|_2^2 \\
\|\tilde{\mathbf{a}}\|_2^2 + \|\tilde{\mathbf{w}}\|_2^2 - 2\langle \tilde{\mathbf{a}}, \tilde{\mathbf{w}} \rangle &\leq \left(1 + \frac{m}{4}\right) \left(\|\vec{a}\|_2^2 + \|\vec{w}\|_2^2 - 2\langle \vec{a}, \vec{w} \rangle\right) \\
\left(1 + \frac{m}{4}\right) 2\langle \vec{a}, \vec{w} \rangle - 4 \cdot \frac{m}{4} &\leq 2\langle \tilde{\mathbf{a}}, \tilde{\mathbf{w}} \rangle \\
\langle \vec{a}, \vec{w} \rangle - \frac{m}{2} &\leq \langle \tilde{\mathbf{a}}, \tilde{\mathbf{w}} \rangle \\
c + m - \frac{m}{2} &\leq \langle \tilde{\mathbf{a}}, \tilde{\mathbf{w}} \rangle.
\end{aligned}$$

Goal: Separate n points in d dimensional space into k groups.



k-means Objective: $Cost(\mathcal{C}_1, \ldots, \mathcal{C}_k) = \min_{\mathcal{C}_1, \ldots \mathcal{C}_k} \sum_{j=1}^{k} \sum_{\vec{x} \in \mathcal{C}_k} \|\vec{x} - \mu_j\|_2^2.$

Write in terms of distances:

$$Cost(\mathcal{C}_1, \ldots, \mathcal{C}_k) = \min_{\mathcal{C}_1, \ldots \mathcal{C}_k} \sum_{j=1}^{k} \sum_{\vec{x}_1, \vec{x}_2 \in \mathcal{C}_k} \|\vec{x}_1 - \vec{x}_2\|_2^2$$

k-means Objective: $Cost(\mathcal{C}_1, \ldots, \mathcal{C}_k) = \min_{\mathcal{C}_1, \ldots \mathcal{C}_k} \sum_{j=1}^{k} \sum_{\vec{x}_1, \vec{x}_2 \in \mathcal{C}_k} \|\vec{x}_1 - \vec{x}_2\|_2^2$ If we randomly project to $m = O\left(\frac{\log n}{\epsilon^2}\right)$ dimensions, for all pairs $\vec{x}_1, \vec{x}_2$,

$$(1-\epsilon)\|\tilde{\mathbf{x}}_1 - \tilde{\mathbf{x}}_2\|_2^2 \leq \|\vec{x}_1 - \vec{x}_2\|_2^2 \leq (1+\epsilon)\|\tilde{\mathbf{x}}_1 - \tilde{\mathbf{x}}_2\|_2^2 \implies$$

Letting $\overline{Cost}(\mathcal{C}_1, \ldots, \mathcal{C}_k) = \min_{\mathcal{C}_1, \ldots \mathcal{C}_k} \sum_{j=1}^{k} \sum_{\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2 \in \mathcal{C}_k} \|\tilde{\mathbf{x}}_1 - \tilde{\mathbf{x}}_2\|_2^2$

$$(1-\epsilon)Cost(\mathcal{C}_1, \ldots, \mathcal{C}_k) \leq \overline{Cost}(\mathcal{C}_1, \ldots, \mathcal{C}_k) \leq (1+\epsilon)Cost(\mathcal{C}_1, \ldots, \mathcal{C}_k)$$

Upshot: Can cluster in m dimensional space (much more efficiently) and minimize $\overline{Cost}(\mathcal{C}_1, \ldots, \mathcal{C}_k)$. The optimal set of clusters will have true cost within $1 + c\epsilon$ times the true optimal.

The Johnson-Lindenstrauss Lemma and High Dimensional Geometry

- High-dimensional Euclidean space looks *very different* from low-dimensional space. So how can JL work?
- Are distances in high-dimensional meaningless, making JL useless?

What is the largest set of mutually orthogonal unit vectors in d-dimensional space? Answer: d.

What is the largest set of unit vectors in d-dimensional space that have all pairwise dot products $|\langle \vec{x}, \vec{y} \rangle| \leq \epsilon$? (think $\epsilon = .01$)

1. d 2. $\Theta(d)$ 3. $\Theta(d^2)$ 4. $2^{\Theta(d)}$

In fact, an exponentially large set of random vectors will be nearly pairwise orthogonal with high probability!

Proof: Let $\vec{x}_1, \ldots, \vec{x}_t$ each have independent random entries set to $\pm 1/\sqrt{d}$.

- $\vec{x}_i$ is always a unit vector.
- $\mathbb{E}[\langle \vec{x}_i, \vec{x}_j \rangle] = ?0$.
- By a Chernoff bound, $\Pr[|\langle \vec{x}_i, \vec{x}_j \rangle| \geq \epsilon] \leq 2e^{-\epsilon^2 d/3}$.
- If we chose $t = \frac{1}{2}e^{\epsilon^2 d/6}$, using a union bound over all $\leq t^2 = \frac{1}{4}e^{\epsilon^2 d/3}$ possible pairs, with probability $\geq 1/2$ all with be nearly orthogonal.

Up Shot: In d-dimensional space, a set of $2^{\Theta(\epsilon^2 d)}$ random unit vectors have all pairwise dot products at most ϵ (think $\epsilon = .01$)

$$\|\vec{x}_i - \vec{x}_j\|_2^2 = \|\vec{x}_i\|_2^2 + \|\vec{x}_j\|_2^2 - 2\vec{x}_i^T\vec{x}_j \geq 1.98.$$

Even with an exponential number of random vector samples, we don't see any nearby vectors.

- Can make methods like nearest neighbor classification or clustering useless.

Curse of dimensionality for sampling/learning functions in high dimensional space – samples are very 'sparse' unless we have a huge amount of data.

- Only hope is if we lots of structure (which we typically do...)

Recall: The Johnson Lindenstrauss lemma states that if $\mathbf{\Pi} \in \mathbb{R}^{m\times d}$ is a random matrix (linear map) with $m = O\left(\frac{\log n}{\epsilon^2}\right)$, for $\vec{x}_1, \ldots, \vec{x}_n \in \mathbb{R}^d$ with high probability, for all i, j:

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

If $\vec{x}_1, \ldots, \vec{x}_n$ are random unit vectors in d-dimensions, can show that $\mathbf{\Pi}\vec{x}_1, \ldots, \mathbf{\Pi}\vec{x}_n$ are essentially random unit vectors in m-dimensions.

$x_1, \ldots, x_n$ are sampled from the surface of $\mathcal{B}_d$ and $\mathbf{\Pi}x_1, \ldots, \mathbf{\Pi}x_n$ are (approximately) sampled from the surface of $\mathcal{B}_m$.

- In d dimensions, $2^{\epsilon^2 d}$ random unit vectors will have all pairwise dot products at most ϵ with high probability.
- For any set of n near orthogonal vectors, $\vec{x}_1, \ldots, \vec{x}_n$, after JL projection, $\mathbf{\Pi}\vec{x}_1, \ldots, \mathbf{\Pi}\vec{x}_n$ will still have pairwise dot products at most $c\epsilon$ with high probability.
- In $m = O\left(\frac{\log n}{\epsilon^2}\right)$ dimensions, $2^{(c\epsilon)^2 m} = 2^{O(\log n)} > n$ random unit vectors will have all pairwise dot products at most $c\epsilon$ with high probability (i.e., still be near orthogonal).
- m is chosen just large enough so that the odd geometry of d-dimensional space will still hold on the n points in question.

Questions?