IIT KHARAGPUR

IIT GANDHINAGAR

NPTEL ONLINE CERTIFICATION COURSES

NPTEL

Scalable Data Science

Lecture 14b: Random Projection Applications and Fast JL

Anirban Dasgupta

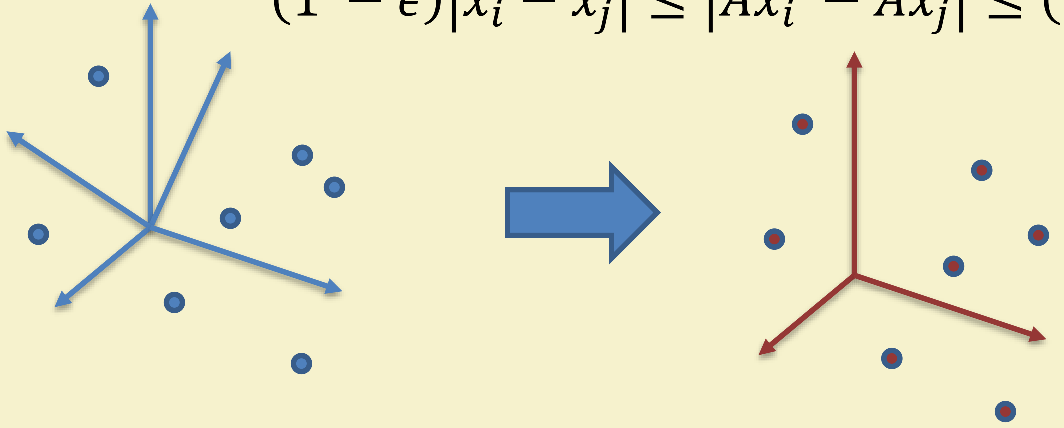
Computer Science and Engineering

IIT GANDHINAGAR

Johnson Lindenstrauss Lemma [JL84]

$\epsilon > 0$, $k \geq \frac{C}{\epsilon^2} \log(n)$. There exists a linear mapping A such that whp, for all (i, j)

$$(1 - \epsilon)|x_i - x_j| \leq |Ax_i - Ax_j| \leq (1 + \epsilon)|x_i - x_j|$$



$x_i \in \Re^d$

Other properties

- The bound $k \geq \frac{c}{\epsilon^2} \log\left(\frac{1}{\delta}\right)$ is tight
  - Alon03, JW11
- Target dimension depends only on ϵ, δ
- Such result cannot hold in distances other than Euclidean metric

Other constructions

[Achlioptas '03]

- Sampling and storing Gaussian random variables is expensive

$$A = \frac{1}{\sqrt{k}} R$$

- $R_{ij} = \begin{cases} +1 \ \ with\, prob\, \frac{1}{3} \\ 0\, with\, prob\, \frac{2}{3} \\ -1\, with\, prob\, \frac{1}{3} \end{cases}$

Other constructions

[Achlioptas '03]

- Sampling and storing Gaussian random variables is expensive

$$A = \frac{1}{\sqrt{k}}\, R$$

- $R_{ij} = \begin{cases} +1 \;\; with\, prob\, \frac{1}{3} \\ 0\; with\, prob\, \frac{2}{3} \\ -1\; with\, prob\, \frac{1}{3} \end{cases}$

R_{ij} $\sim$ any subgaussian distribution with variance 1

Example Application: PCA

- Suppose we have $A \in \Re^{n \times d}$, want to get a rank-k approximation A' so that $|A - A'|_F$ is minimized

- Optimal low-rank approximation, $A_k = U_k \Sigma_k V_k^t$
  - Takes time $O(nd \min(n, d))$

Low rank Approximation

- Projection $P_A^k = U_k U_k^t$, $\left|A - P_A^k A\right|_2 = \sigma_{k+1}$
- For any B and P_B^k,

$$\left|A - P_B^k A\right| \leq \sigma_{k+1} + \sqrt{2|AA^t - BB^t|} \quad \text{[FKV04]}$$

- Want B that is
  - Efficiently computable and small
  - Leads to low error, $|AA^t - BB^t| \leq \epsilon |AA^t|$

[Example courtesy Edo Liberty]

Cheap and effective low rank

- $A \in \Re^{n \times d}$, create $R \in \Re^{d \times k}$ as a JL matrix
- $B = AR \in \Re^{n \times k}$

Cheap and effective low rank

- $A \in \Re^{n \times d}$, create $R \in \Re^{d \times k}$ as a JL matrix
- $B = AR \in \Re^{n \times k}$
- Note that $E[BB^t] = A\,E[RR^t]A^t = AA^t$

JL in low rank approximation

- Using the JL property

$$\Pr[|yR|^2 - |y|^2 > \epsilon|y|^2] < \exp(-ck\epsilon^2)$$

Using union bound over a ϵ –net, $k = \tilde{O}\left(\frac{rank(A)}{\epsilon^2}\right)$

$$|A^tA - B^tB| = \sup_{|x|=1}\left||xA|^2 - |xAR|^2\right| \leq \epsilon|A^tA|$$

Time taken = $O(ndk)$

Summary

- JL random projections a versatile tool
  - Tight bound on the number
- Number of ways to construct, will see more
- Saw a specific application

References:

- Primary references
  - The Random Projection Method, Santosh Vempala, AMS.
  - Foundations of Data Science, Blum, Hopcroft, Kannan, *https://www.cs.cornell.edu/jeh/book.pdf*
  - Survey by Long Chen, https://www.math.uci.edu/~chenlong/RNLA/JL.pdf

Time taken for projection

- Matrix vector multiplication = $O(kd)$
  - $k = \Omega\left(\frac{1}{\epsilon^2}\right)$
- We also know that there is a lower bound on the target dimension
- Can we make the projection faster?

Thought Experiment

- Suppose projection matrix is very sparse

$$A_{ij} = \begin{cases} 0 \; w.p. 1 - p \\ N\left(0, \frac{1}{\sqrt{p}}\right) \; w.p. p \end{cases}$$

- Set $p \sim \frac{1}{d}$. Time now is only $O(k)$. But….

Thought Experiment

- Suppose projection matrix is very sparse

$$A_{ij} = \begin{cases} 0 \; w.p. 1 - p \\ N\left(0, \frac{1}{\sqrt{p}}\right) \; w.p. p \end{cases}$$

- Set $p \sim \frac{1}{d}$. Time now is only $O(k)$. But….
  - Fails to preserve norm, esp. for sparse vectors
  - Works fine if the vector is all dense!

Ingredients

- Hadamard matrices

$$H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \qquad H_{2^{k+1}} = \begin{pmatrix} H_{2^k} & H_{2^k} \\ H_{2^k} & -H_{2^k} \end{pmatrix}$$

Hadamard Matrices

- Defined only when dimension d is of the form 2^k
  - Assume this
- Multiplying a vector by H_d takes time $O(d \log d)$

$$T(d) = 2T\left(\frac{d}{2}\right) + O(d)$$

Densifying using Hadamard

$x \in R^d$

$D \in \Re^{d \times d}$ diagonal with $D_{ii} = \begin{cases} +1 \; w.p. \frac{1}{2} \\ -1 \; w.p. \frac{1}{2} \end{cases}$

$$y = HDx$$

Calculating y takes time $O(d \log d)$

Intuition

- H itself is a rotation
  - sparse vectors are rotated to dense vectors (Uncertainty principle)
  - but, it is a rotation, hence some (dense) vectors will be pre-image of sparse vectors
  - need to ensure that adversary cannot choose such vectors as input
  - randomization using the diagonal achieves this

Densification claim

$x \in \Re^d$, $|x|_2 = 1$

<u>Claim:</u> $\max_i |(HDx)_i| \leq O\left(\frac{\log(nd)}{d}\right)^{1/2}$

Application of Chernoff style tail inequality per coordinate and union bound

Projecting a dense vector

$y = HDx,\ \max_i |y_i| \approx O(\sqrt{\log(nd)/d})$

$$P = \begin{cases} 0, w.p.\ 1-q \\ N\left(0, \frac{1}{\sqrt{q}}\right), w.p.\ q \end{cases},\ P \in \Re^{k \times d}, k = O\left(\frac{1}{\epsilon^2}\log\left(\frac{1}{\delta}\right)\right)$$

$z = PHDx$ is the final projected vector

IIT Gandhinagar
Indian Institute of Technology Gandhinagar

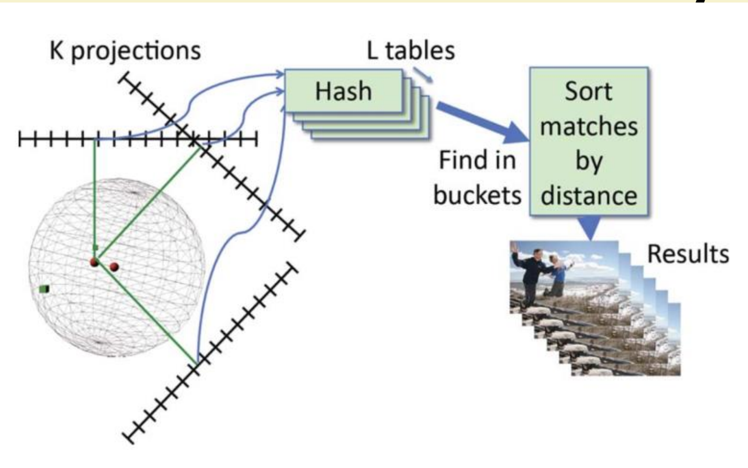
NPTEL ONLINE CERTIFICATION COURSES

Fast JL Transform [AC09]

If $q = O\left(|x|_\infty^2\right) = O\left(\frac{\log(nd)}{d}\right)$, PHD satisfies JL property

Calculating $y = PHDx$ takes time $O(d \log d \; + k log(nd))$, potentially much faster than original Gaussian construction

Locality Sensitive Hashing



Given input data, radius r, approx factor c and confident δ

<u>Output:</u> if there is any point at distance $\leq r$ then w.p. $1-\delta$ return one at distance $\leq cr$

$$h_i(x) = \left\lfloor \frac{x \cdot v_i + b_i}{w} \right\rfloor$$

Picture courtesy Slaney et al.

Time taken

- Total query time = time to hash + time to check all candidates
  - Calculating k-hash indices takes time $O(kd)$
  - Calculating indices for L buckets takes time $O(kdL)$

- Can we reduce query time?

Creating hash indices

- Looking at LSH as random projection + quantization
  - $A \in R^{k \times d}$, $b \in R^{k}$ is a Gaussian JL matrix
  - We first project and then bucketize
  - calculate $\left\lfloor \frac{Ax+b}{w} \right\rfloor$, k-index key calculated at once

- Time taken by matrix-vector multiplication = $O(kd)$ per hash table

Collision Probability

- $p(u) = \Pr[h_i(q) = h_i(q)]$ when $|p - q| = u$
- $p(u) = \int_0^W \frac{1}{u} f(t/u) \left(1 - \frac{t}{W}\right) dt$, $f(v) = \text{pdf of } |N(0,1)|$
- This is decreasing with increasing u

ACHash [DKS11]

- When calculating a k-tuple hash bucket index
  - $\left\lfloor \frac{Ax+b}{w} \right\rfloor$, use $A = PHD$
  - $q \approx O\left(\frac{\log(d)}{d}\right)$
  - Projection time = $O(d\ log\ d + kL \log^2 d\ )$

ACHash

$p_{AC}(u) =$ probability that a k-tuple hash bucket has same value for two points at distance u

We can show that

$$-(k+1)\delta + p^k((1+\epsilon)u) \leq p_{\mathrm{AC}}(u) \leq p^k((1-\epsilon)u) + (k+1)\delta.$$

i.e. collision prob does not change much

DHHash

We can make the projection faster

D = random diagonal matrix of ± 1

G = random diagonal matrix, $G_{ii} \sim N(0, 1)$

M = random permutation matrix

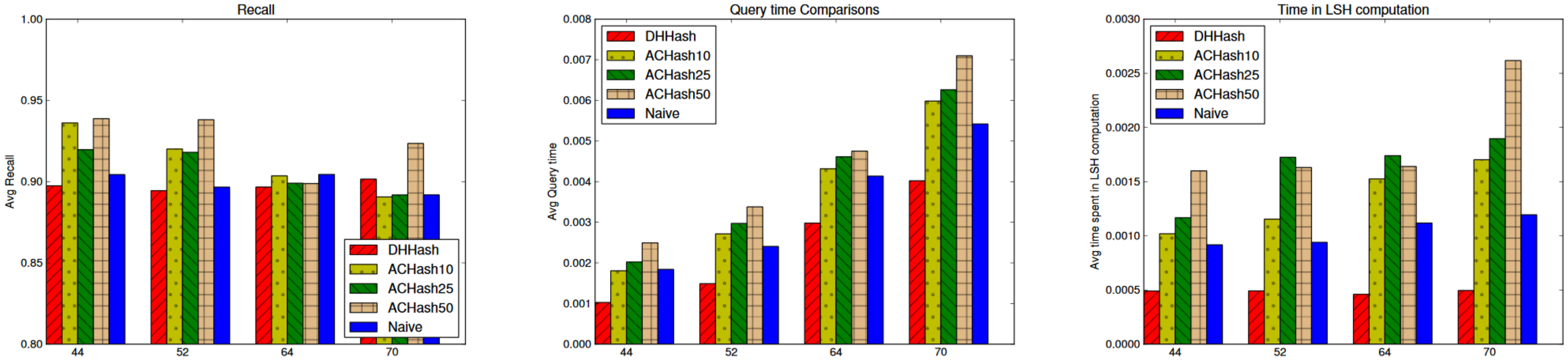
Hash value calculated as $\left\lfloor \frac{HGMHDx + b}{w} \right\rfloor$

DHHash

- The above creates d bits
- Sample kL indices and create L hash bucket ids, each of size k

- Total calculation time for all L bucket-ids = $O(d \log d + kL)$
- Also performs nicely in practice

Experiments: faster query time with more or less same recall



(d) Recall, LSH query time, and LSH computation time for P53.

References:

- Primary references for this lecture
    - The Random Projection Method, Santosh Vempala, AMS.
    - Foundations of Data Science, Blum, Hopcroft, Kannan, *https://www.cs.cornell.edu/jeh/book.pdf*
    - Survey by Long Chen, https://www.math.uci.edu/~chenlong/RNLA/JL.pdf

    - Fast Johnson Lindenstrauss Transform, Ailon and Chazelle, SIAM J Computing 2009.
    - Fast Locality Sensitive Hashing, Dasgupta, Kumar, Sarlos, KDD 2011.

Thank You!!