Randomized Robust Linear Regression for big data applications

Yannis Kopsinis

[1]Dept. of Informatics & Telecommunications, UoA

Thursday, Apr 16, 2015

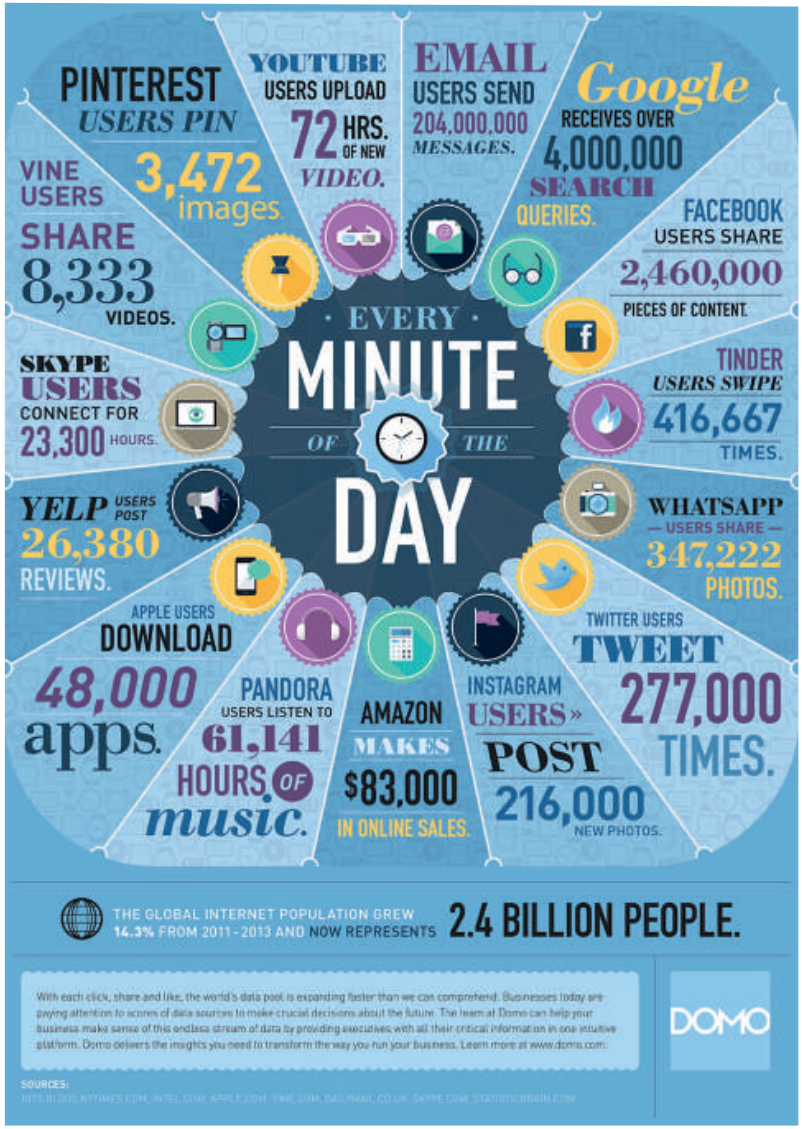
In collaboration with
S. Chouvardas, Harris Georgiou, Sergios Theodoridis

Outline

Big Data era

Why all the fuss?

- Massive Data Volumes is not a new thing - 65×10^{18} bytes flowed through telecommunication networks on 2007
- First New Thing: Established Data analysis and Machine learning techniques face Big Challenges
- Second New Thing: Novel approaches for data capturing, handling and processing emerged
- Third New Thing: New modalities and increased complexity (internet of things, cyber-physical systems, smart homes, smart cars etc.)



Big Data era

Why all the fuss?

- Marketing policies

- From Big Data to Insights $\rightarrow$ New emerging applications
- From Big Data to Insights $\rightarrow$ Big Profits
- 4.4 million data scientists needed by 2015 (IBM)
- Many challenging open problems / paradigm shift

Big Data era

What characterizes big data

- Volume (scale of data)
- Variety (different forms of data)
- Velocity (streaming data)
- Veracity (presence of outliers / corruptions)

Big Data era

How to deal with big data

- **Distributed Processing**
- Centralized approach, e.g. MapReduce/Hadoop
- Decentralized approach, e.g. ad-hoc in-network processing
- Share processing power and storage requirements
- Privacy protection

Big Data era

How to deal with big data

- **Distributed Processing**
- Centralized approach, e.g. MapReduce/Hadoop
- Decentralized approach, e.g. ad-hoc in-network processing
- Share processing power and storage requirements
- Privacy protection
- **Online Learning**
- Process data on the fly
- Limited storage demands
- Reduced computational complexity (stochastic gradient descent)
- Dealing with time-varying situations

Big Data era

How to deal with big data

- **Distributed Processing**
- Centralized approach, e.g. MapReduce/Hadoop
- Decentralized approach, e.g. ad-hoc in-network processing
- Share processing power and storage requirements
- Privacy protection
- **Online Learning**
- Process data on the fly
- Limited storage demands
- Reduced computational complexity (stochastic gradient descent)
- Dealing with time-varying situations

- **Randomized Methods**

Randomized Methods

Major Principle that governs randomized methods

Instead of working with the original large-scale data matrices, operate on compressed versions of them. The compression is realized via computationally efficient dimensionality reduction, which is performed in a randomized rather than in a deterministic way.

Some Facts!

- It is a very appealing idea!
  - Data are highly compressible
  - Low speed memory units are the major bottleneck
- It is applicable to ubiquitous data analysis and ML tasks, even to basic matrix operations
  - Matrix Multiplication
  - Linear Regression
  - Low-rank Matrix approximation (Singular Value Decomposition)

Some Facts! (cont.)

- Which is the price to pay for?
  - Provide approximate rather than exact solutions
  - There is a probability of failure

Randomized Linear Regression

Linear LS Regression

$$\underset{N\times 1}{\boldsymbol{b}} = \underset{N\times l}{\boldsymbol{A}}\ \underset{l\times 1}{\boldsymbol{x}_*} + \underset{N\times 1}{\boldsymbol{\eta}}$$

$N \gg l$, and at least N very large

$$\hat{\boldsymbol{x}}_{LS} = \arg\min_{\boldsymbol{x}\in\mathbb{R}^l} \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|_2^2$$

$$\hat{\boldsymbol{x}}_{LS} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b}$$

Computational complexity: $\mathcal{O}(Nl^2)$ via QR decomposition

Randomized Least Squares

Randomized Linear LS Regression

$$\underset{N\times 1}{\boldsymbol{b}} = \underset{N\times l}{\boldsymbol{A}}\ \underset{l\times 1}{\boldsymbol{x}_*} + \underset{N\times 1}{\boldsymbol{\eta}}$$

$$\underset{d\times 1}{\underline{\boldsymbol{b}}} = \underset{d\times N}{\boldsymbol{R}}\ \underset{N\times 1}{\boldsymbol{b}}, \quad \underset{d\times l}{\underline{\boldsymbol{A}}} = \underset{d\times N}{\boldsymbol{R}}\ \underset{N\times l}{\boldsymbol{A}}, \quad \text{where } d \ll N$$

$$\hat{\boldsymbol{x}}_R = \arg\min_{\boldsymbol{x}\in\mathbb{R}^l} \|\underline{\boldsymbol{b}} - \underline{\boldsymbol{A}}\boldsymbol{x}\|_2^2,$$

Computational complexity: $\mathcal{O}(dl^2) + \mathcal{C}(\boldsymbol{R}) + \mathcal{T}(\boldsymbol{RA})$

Randomized Least Squares

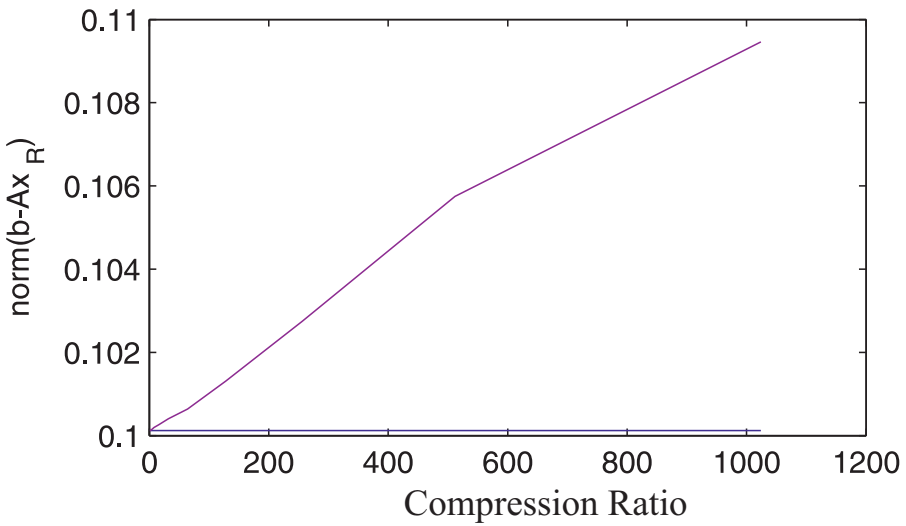
Randomized Linear LS Regression

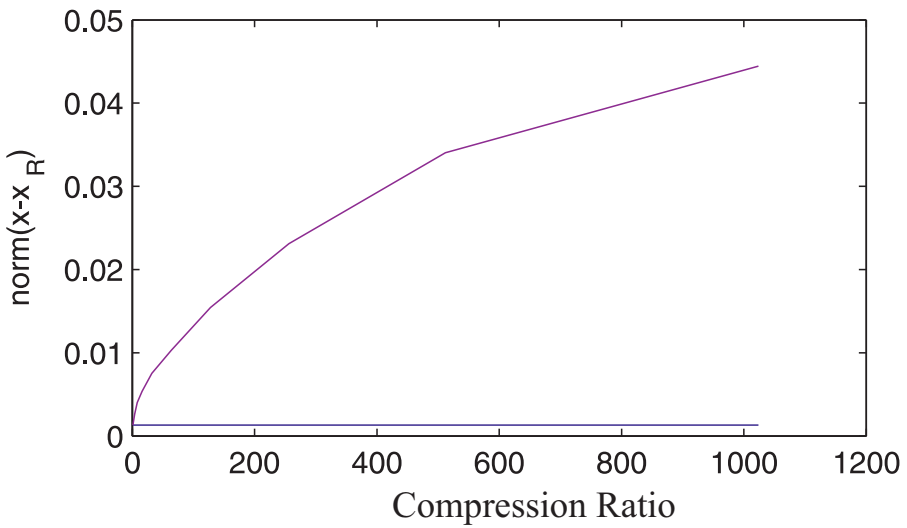
$$\underset{N\times 1}{\boldsymbol{b}} = \underset{N\times l}{\boldsymbol{A}}\ \underset{l\times 1}{\boldsymbol{x}_*} + \underset{N\times 1}{\boldsymbol{\eta}}$$

$$\underset{d\times 1}{\underline{\boldsymbol{b}}} = \underset{d\times N}{\boldsymbol{R}}\ \underset{N\times 1}{\boldsymbol{b}}, \quad \underset{d\times l}{\underline{\boldsymbol{A}}} = \underset{d\times N}{\boldsymbol{R}}\ \underset{N\times l}{\boldsymbol{A}}, \quad \text{where } d \ll N$$

$$\hat{\boldsymbol{x}}_R = \arg\min_{\boldsymbol{x}\in\mathbb{R}^l} \|\underline{\boldsymbol{b}} - \underline{\boldsymbol{A}}\boldsymbol{x}\|_2^2,$$

Computational complexity: $\mathcal{O}(dl^2) + \mathcal{C}(\boldsymbol{R}) + \mathcal{T}(\boldsymbol{RA})$





Randomized Least Squares

Randomized Linear LS Regression

$$\underset{N\times 1}{\boldsymbol{b}} = \underset{N\times l}{\boldsymbol{A}}\ \underset{l\times 1}{\boldsymbol{x}_*} + \underset{N\times 1}{\boldsymbol{\eta}}$$

$$\underset{d\times 1}{\underline{\boldsymbol{b}}} = \underset{d\times N}{\boldsymbol{R}}\ \underset{N\times 1}{\boldsymbol{b}}, \quad \underset{d\times l}{\underline{\boldsymbol{A}}} = \underset{d\times N}{\boldsymbol{R}}\ \underset{N\times l}{\boldsymbol{A}}, \quad \text{where } d \ll N$$

$$\hat{\boldsymbol{x}}_R = \arg\min_{\boldsymbol{x}\in\mathbb{R}^l} \|\underline{\boldsymbol{b}} - \underline{\boldsymbol{A}}\boldsymbol{x}\|_2^2,$$

Computational complexity: $\mathcal{O}(dl^2) + \mathcal{C}(\boldsymbol{R}) + \mathcal{T}(\boldsymbol{RA})$

Some theoretic results [Drineas 2011]

If $d = \mathcal{O}\left(l(\ln l)(\ln N) + \frac{l\ln N}{\epsilon}\right)$, then with probability 0.8

$$\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_R\|_2 \leq (1+\epsilon)\|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_{LS}\|_2$$

$$\|\boldsymbol{x}_{LS} - \boldsymbol{x}_R\|_2 \leq \sqrt{\epsilon}\left(\kappa(\boldsymbol{A})\sqrt{\gamma^{-2}-1}\|\boldsymbol{x}_{LS}\|_2\right)$$

if $N \leq e^l$, and $\|\boldsymbol{U_A}\boldsymbol{U_A}^T\boldsymbol{b}\|_2 \geq \gamma\|\boldsymbol{b}\|_2, \gamma \in (0,1]$

Johnson–Lindenstrauss (JL) seminal work (1984)

Lemma

For any set, S, of k points, $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots$ in $\mathbb{R}^N$ there exist a linear mapping $\boldsymbol{R} : \mathbb{R}^N \to \mathbb{R}^d$, with $d = \mathcal{O}(\epsilon^{-2} \log l)$, such that all the pairwise distances are approximately preserved:

$$\forall i, j \quad (1-\epsilon)\|\boldsymbol{u}_i - \boldsymbol{u}_j\|_2^2 \leq \|\boldsymbol{R}\boldsymbol{u}_i - \boldsymbol{R}\boldsymbol{u}_j\|_2^2 \leq (1+\epsilon)\|\boldsymbol{u}_i - \boldsymbol{u}_j\|_2^2$$

W.B. Johnson and J. Lindenstrauss, "Extensions of Lipshitz mapping into Hilbert space," Contemporary Mathematics, 1984.

Johnson–Lindenstrauss (JL) seminal work (1984)

Lemma

For any set, S, of k points, $\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots$ in $\mathbb{R}^N$ there exist a linear mapping $\boldsymbol{R} : \mathbb{R}^N \to \mathbb{R}^d$, with $d = \mathcal{O}(\epsilon^{-2} \log l)$, such that all the pairwise distances are approximately preserved:

$$\forall i, j \quad (1-\epsilon)\|\boldsymbol{u}_i - \boldsymbol{u}_j\|_2^2 \leq \|\boldsymbol{R}\boldsymbol{u}_i - \boldsymbol{R}\boldsymbol{u}_j\|_2^2 \leq (1+\epsilon)\|\boldsymbol{u}_i - \boldsymbol{u}_j\|_2^2$$

W.B. Johnson and J. Lindenstrauss, "Extensions of Lipshitz mapping into Hilbert space," Contemporary Mathematics, 1984.

JL Transforms ($\boldsymbol{R}$ Matrix)

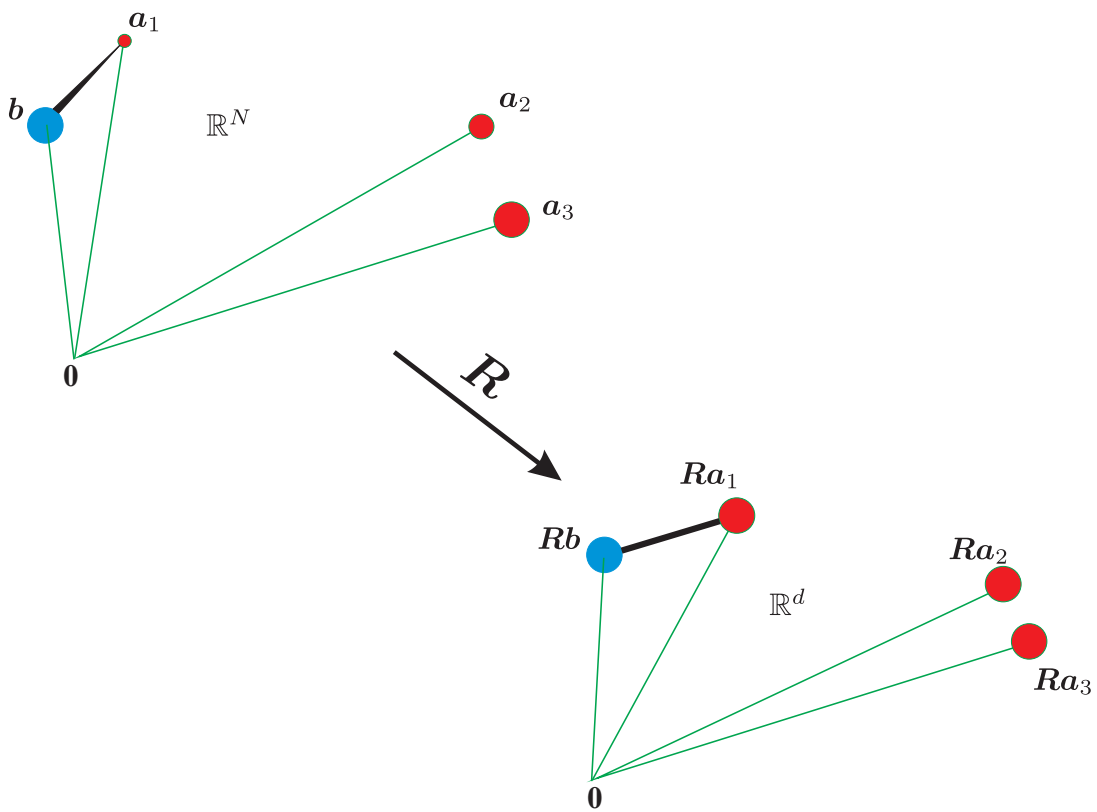
- Johnson and Lindenstrauss (1984): Choose $\boldsymbol{R}$ uniformly at random from the space of projection matrices.
- Frankl and Maehara (1988): Random orthogonal matrix
- Indyk and Motwani (1998), DasGupta and Gupta (1999): entries chosen uniformly at random from $\mathcal{N}(0, \frac{1}{N})$

Johnson–Lindenstrauss (JL) seminal work

JL geometry in the Linear Regression case

$$\underset{N\times 1}{\boldsymbol{b}} = \underset{N\times l}{\boldsymbol{A}}\ \underset{l\times 1}{\boldsymbol{x}_*} + \underset{N\times 1}{\boldsymbol{\eta}}$$

$$\hat{\boldsymbol{x}}_R = \arg\min_{\boldsymbol{x}\in\mathbb{R}^l} \|\boldsymbol{Rb} - \boldsymbol{RAx}\|_2^2,$$



Accelerating Johnson–Lindenstrauss (JL) Transforms

Achlioptas (2003)

$$a_{i,j} = \begin{cases} +\sqrt{\frac{3}{d}}, & \text{with probability } \frac{1}{6}, \\ 0, & \text{with probability } \frac{2}{3}, \\ -\sqrt{\frac{3}{d}}, & \text{with probability } \frac{1}{6}. \end{cases}$$

then if $d \geq \frac{4+2\beta}{\epsilon^2/2-\epsilon^3/3} log(l)$, each pairwise distance is preserved with probability at least $1 - l^{-\beta}$.

Accelerating Johnson–Lindenstrauss (JL) Transforms

Achlioptas (2003)

$$a_{i,j} = \begin{cases} +\sqrt{\frac{3}{d}}, & \text{with probability } \frac{1}{6}, \\ 0, & \text{with probability } \frac{2}{3}, \\ -\sqrt{\frac{3}{d}}, & \text{with probability } \frac{1}{6}. \end{cases}$$

then if $d \geq \frac{4+2\beta}{\epsilon^2/2-\epsilon^3/3} log(l)$, each pairwise distance is preserved with probability at least $1 - l^{-\beta}$.

Fast JL Transforms (e.g. Sarlos 2006, Drineas et all 2011)

$$\boldsymbol{R} = \boldsymbol{PHD}$$

- $\boldsymbol{D} \in \mathbb{R}^{N \times N}$ diagonal matrix with ± 1
- $\boldsymbol{H} \in \mathbb{R}^{N \times N}$ Hadamard matrix (normalized)
- $\boldsymbol{P} \in \mathbb{R}^{d \times N}$ a sparse matrix (or simply a Sampling Matrix)

Accelerating Johnson–Lindenstrauss (JL) Transforms

Fast JL Transforms (e.g. Sarlos 2006, Drineas et all 2011)

$$\boldsymbol{R} = \boldsymbol{PHD}$$

- $\boldsymbol{D} \in \mathbb{R}^{N \times N}$ diagonal matrix with ± 1
- $\boldsymbol{H} \in \mathbb{R}^{N \times N}$ Hadamard matrix (normalized)
- $\boldsymbol{P} \in \mathbb{R}^{d \times N}$ a sparse matrix (or simply a Sampling Matrix)

Accelerating Johnson–Lindenstrauss (JL) Transforms

Fast JL Transforms (e.g. Sarlos 2006, Drineas et all 2011)

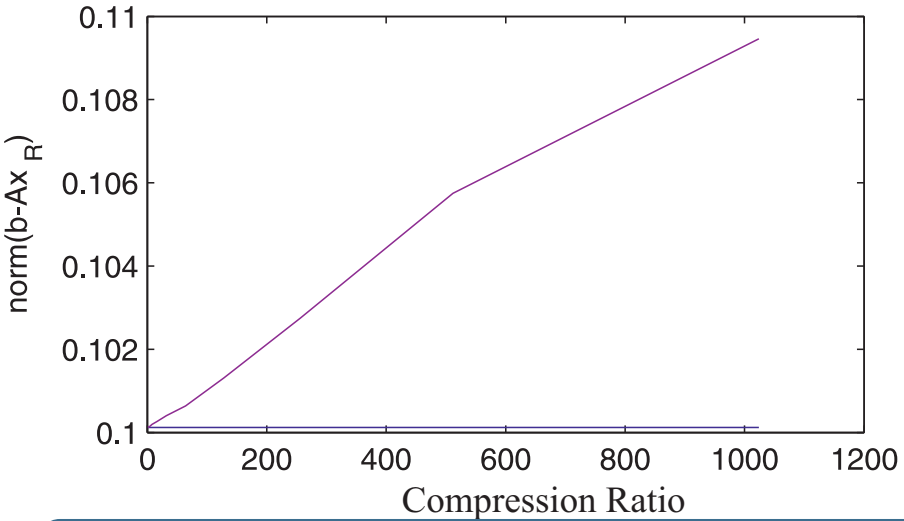
$$\boldsymbol{R} = \boldsymbol{PHD}$$

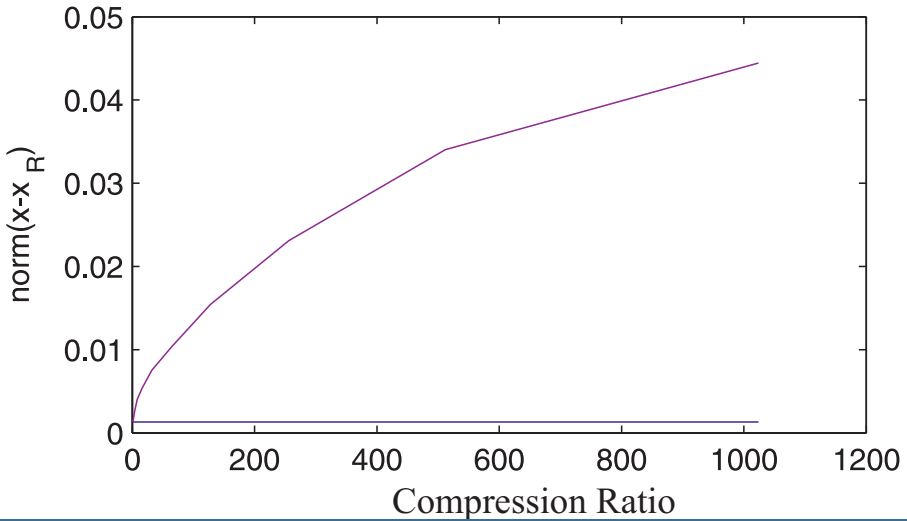
- $\boldsymbol{D} \in \mathbb{R}^{N \times N}$ diagonal matrix with ± 1
- $\boldsymbol{H} \in \mathbb{R}^{N \times N}$ Hadamard matrix (normalized)
- $\boldsymbol{P} \in \mathbb{R}^{d \times N}$ a sparse matrix (or simply a Sampling Matrix)

Computational Complexity / Facts

- It is called Randomized Hadamard Transform
- Multiplication with D is just selective sign changes
- $\boldsymbol{Ha} \rightarrow \mathcal{O}(N \log k)$, where $k \leq N$, is the number Hadamard components needed
- Overall, $\boldsymbol{RA}$ takes $\mathcal{O}(lN \log k)$

Fast LS approximation Example





Randomized Hadamard Transform

Recall: $d = \mathcal{O}\left(l(\ln l)(\ln N) + \frac{l \ln N}{\epsilon}\right)$

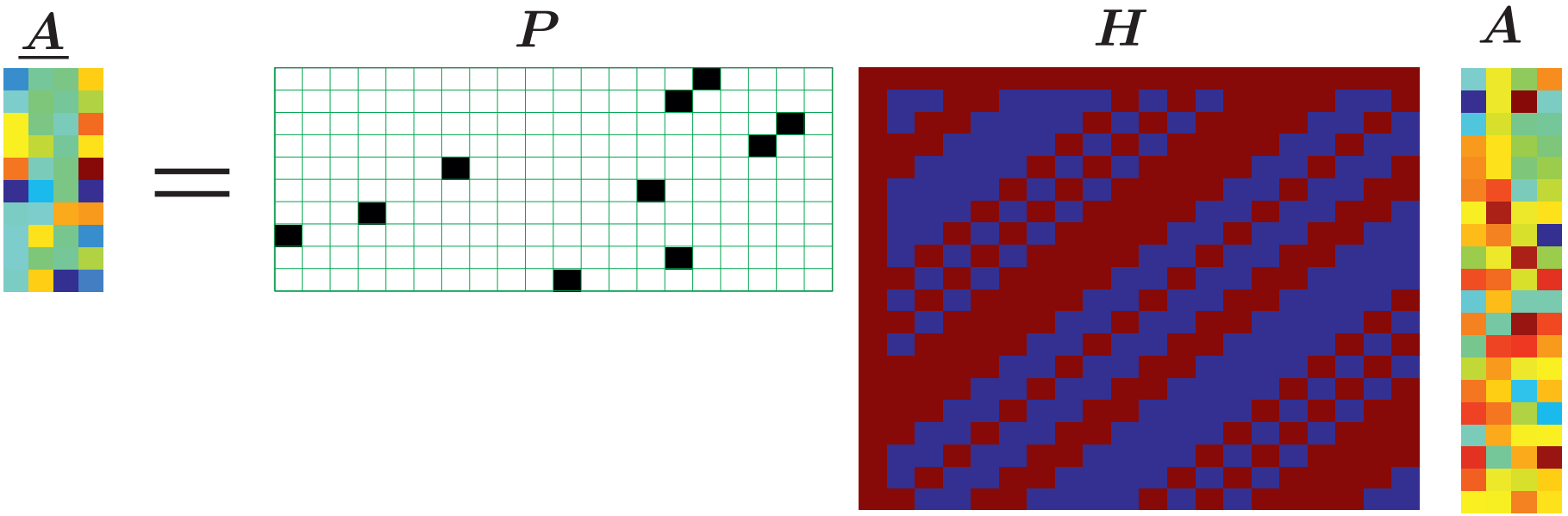
- Example 1: $N = 10^6,\ l = 200,\ \epsilon = 0.1.$

$$\frac{Nl^2}{dl^2 + lN\log(d)} = 10, \qquad \frac{N}{d} = 23$$

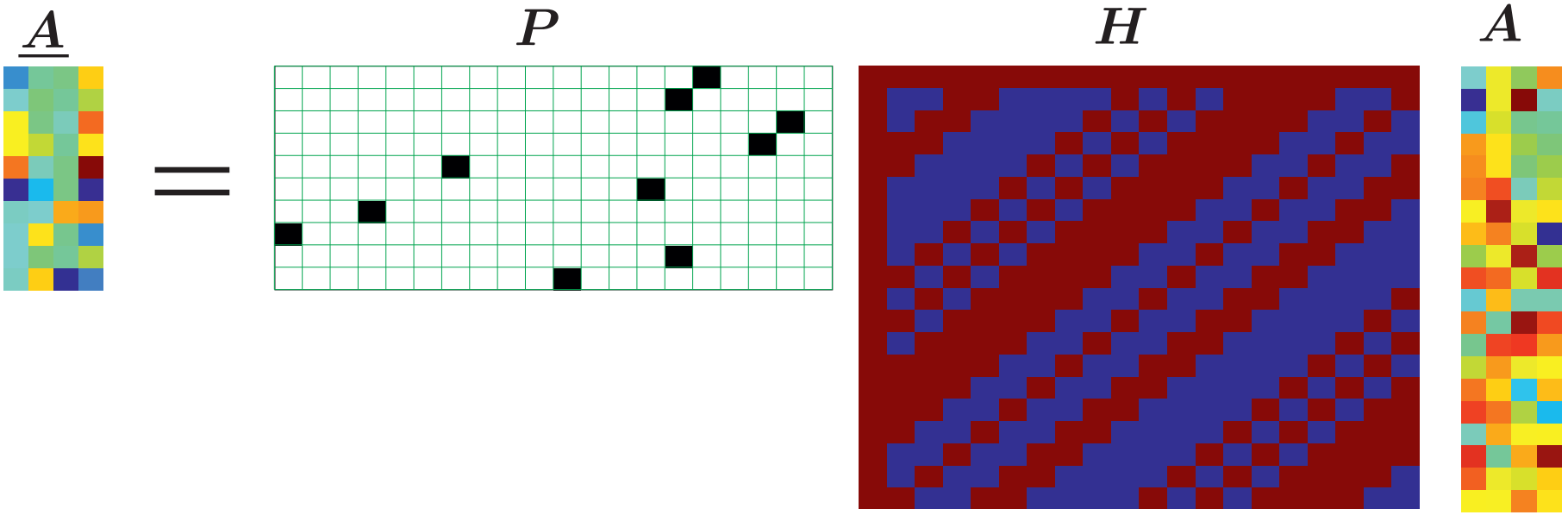
- Example 2: $N = 10^8,\ l = 1000,\ \epsilon = 0.1.$

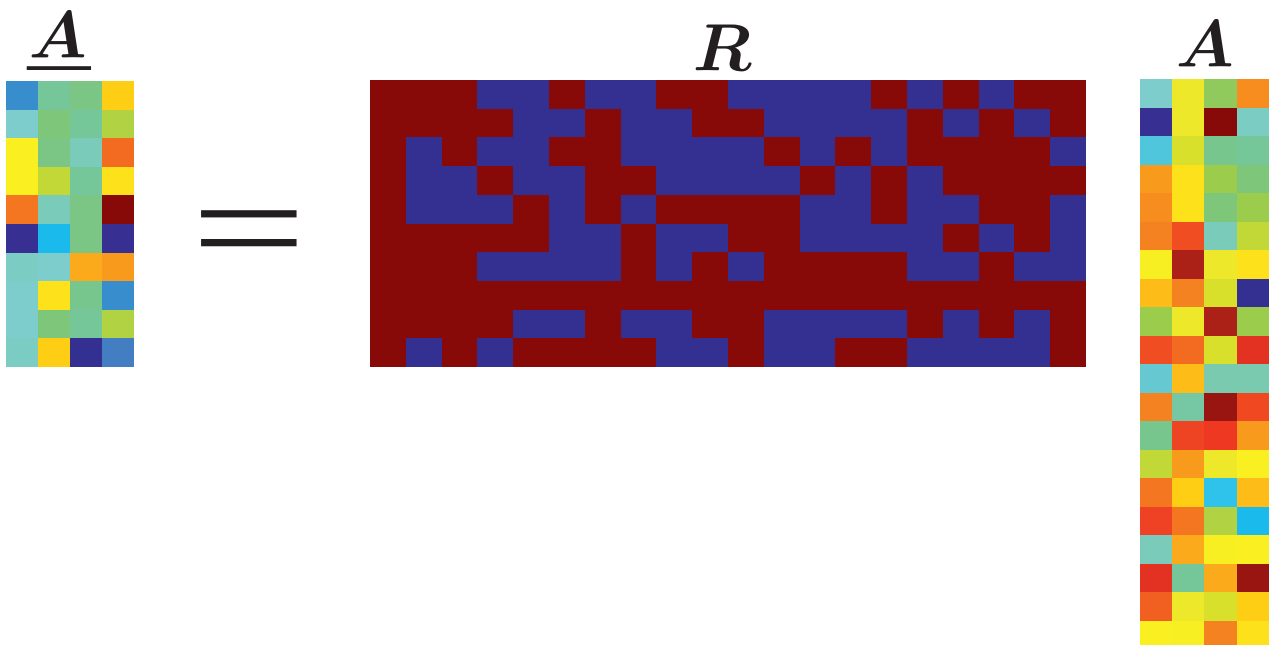
$$\frac{Nl^2}{dl^2 + lN\log(d)} = 63, \qquad \frac{N}{d} = 321$$

Randomized projections vs Randomized sampling

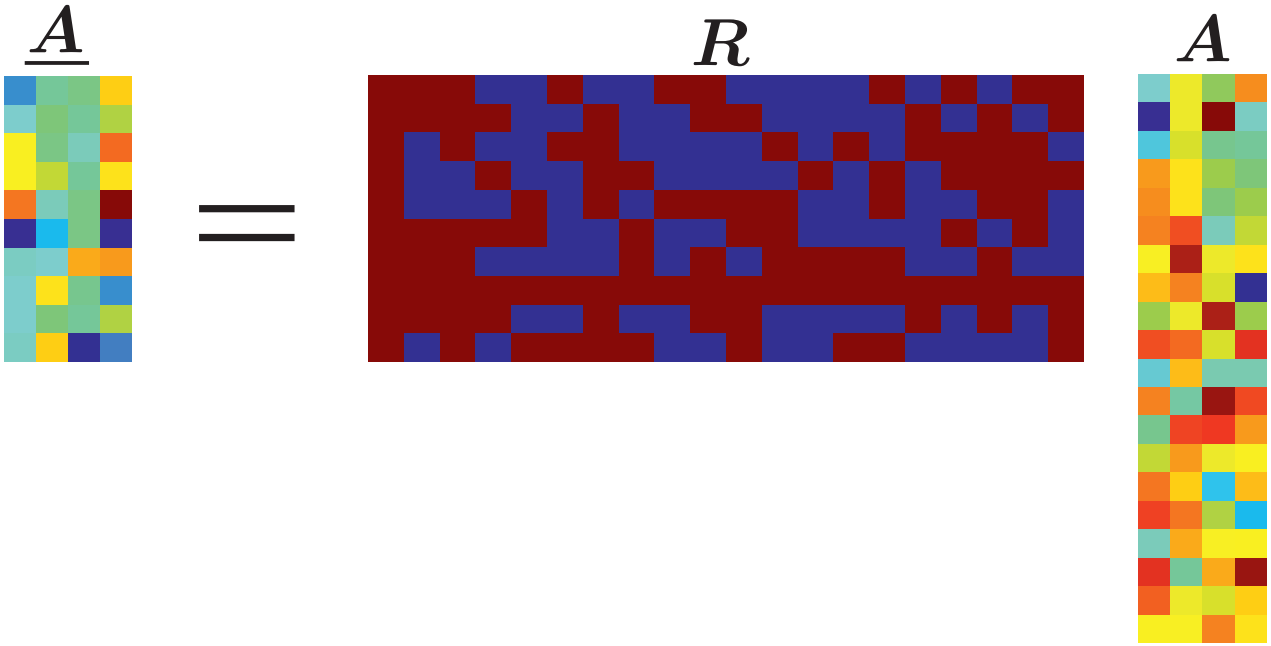


Randomized projections vs Randomized sampling

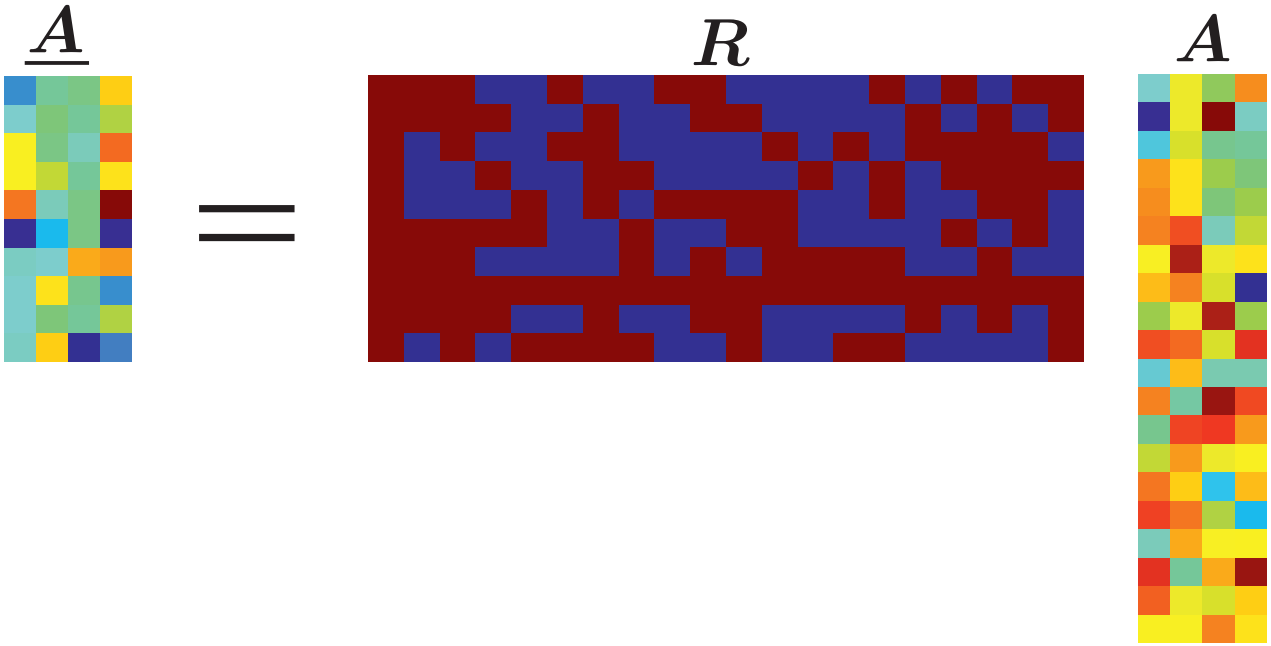


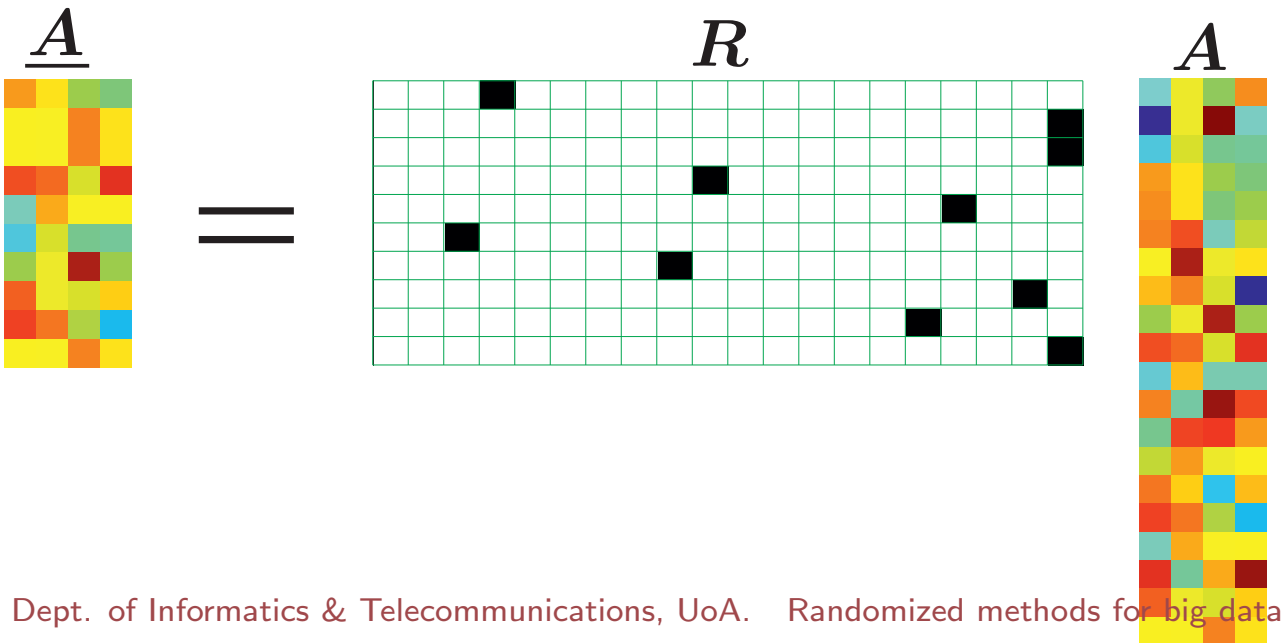


Randomized projections vs Randomized sampling



Randomized projections vs Randomized sampling





Statistical Leverage

Hat Matrix

$$\boldsymbol{b} = \boldsymbol{A} \quad \boldsymbol{x}_* + \boldsymbol{\eta}$$

$$\hat{\boldsymbol{x}}_{LS} = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b}$$

$$\hat{\boldsymbol{b}} = \boldsymbol{A}\hat{\boldsymbol{x}}_{LS} = \boldsymbol{A}(\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{b} \Rightarrow \hat{\boldsymbol{b}} = \boldsymbol{H}\boldsymbol{b}$$

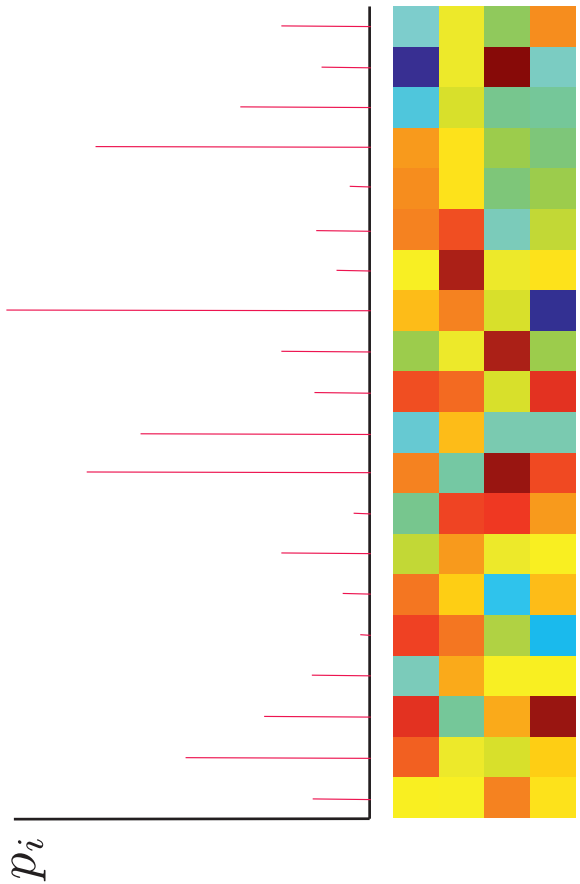
Statistical Leverage Scores

- $\boldsymbol{H}_{ij}$ measures the influence exerted on the prediction $\hat{\boldsymbol{b}_i}$ by observation $\boldsymbol{b}_j$
- $\ell_i = \boldsymbol{H}_{ii}$ measures the importance of $\boldsymbol{b}_i$ in determining the best LS fit.
- $\ell_i,\ i = 1 \ldots N$ are referred to as statistical leverage scores
- $\boldsymbol{H} = P_{\boldsymbol{A}} = \boldsymbol{U}\boldsymbol{U}^T$, for any orthogonal matrix spanning the column space of $\boldsymbol{A}$.
- $\ell_i = \|\boldsymbol{U}_{i,.}\|_2^2$
- Very large $\boldsymbol{H}_{ii}$ are indicators for outliers in $\boldsymbol{A}$.

Randomized Sampling

Sampling Strategy

- Construct an importance sampling distribution $\{p_i\}_{i=1}^N$, with $p_i = \frac{\ell_i}{l}$.
- Intuitively, the larger the p_i is the higher the probability of selecting the ith data sample $(\boldsymbol{b}_i, \boldsymbol{A}_{i,.})$.
- Start with a zero-matrix $\boldsymbol{R} \in \mathbb{R}^{d \times N}$. Then successively fill a single entry of each row, say the ith as follows
  - Sample a random value, say $\rho \in [1, \ldots, N]$, from the importance sampling distribution.
  - Set $\boldsymbol{R}_{i,\rho} = \frac{1}{dp_\rho}$.
- Via $\underline{\boldsymbol{A}} = \boldsymbol{R}\boldsymbol{A}$, $\underline{\boldsymbol{A}}$ comprises rescaled rows of $\boldsymbol{A}$ randomly sampled with replacement.



Computation of the Statistical Leverage

"Naive way"

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$$

then $\boldsymbol{U}$ is orthogonal spanning the column space of $\boldsymbol{A}$.
Alas, complexity $\mathcal{O}(Nl^2)$

Fast approximations (Drineas 2012)

- Exploit the fact that

$$\ell_i = \|(\boldsymbol{U}\boldsymbol{U}^T)_{i,.}\|_2^2 = \|(\boldsymbol{A}\boldsymbol{A}^\dagger)_{i,.}\|_2^2$$

- Construct two fast JL transform matrices (e.g. randomized Hadamard transforms), $\boldsymbol{\Pi}_1 \in \mathbb{R}^{r_1 \times N}$, $\boldsymbol{\Pi}_2 \in \mathbb{R}^{r_2 \times r_1}$
- Estimate leverage scores as

$$\hat{\ell}_i = \|(\boldsymbol{A}(\boldsymbol{\Pi}_1\boldsymbol{A})^\dagger\boldsymbol{\Pi}_2)_{i,.}\|_2^2$$

- it is proved that $|\ell_i - \hat{\ell}_i| \leq \epsilon\ell_i, \ \forall i$

Randomized projections vs Randomized sampling

common ground!

- Random projections "uniformize" the leverage scores (so simple random sampling is adequate)
- Without random projection-based preprocessing, advanced sampling is needed (and Leverage scores-based importance sampling is doing the job!)

Robust Randomized Linear Regression

Robust Linear Regression

Recall Veracity!

$$\boldsymbol{b} = \boldsymbol{A} \ \ \boldsymbol{x}_* + \boldsymbol{\eta}, \qquad \boldsymbol{\eta} = \boldsymbol{n} + \boldsymbol{o}$$

$$\hat{\boldsymbol{x}}_{LAD} = \arg\min_{\boldsymbol{x} \in \mathbb{R}^l} \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|_1$$

- Least Absolute Deviations do not admit a closed form solution
- Linear programming using, e.g. interior-point methods $\mathcal{O}(\text{poly}(N))$
- Use approximate, iterative solutions, e.g. ADMM [Boyed 2011].

Robust Randomized Linear Regression

Robust Linear Regression

Recall Veracity!

$$\boldsymbol{b} = \boldsymbol{A}\ \ \boldsymbol{x}_* + \boldsymbol{\eta}, \qquad \boldsymbol{\eta} = \boldsymbol{n} + \boldsymbol{o}$$

$$\hat{\boldsymbol{x}}_{LAD} = \arg\min_{\boldsymbol{x} \in \mathbb{R}^l} \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|_1$$

- Least Absolute Deviations do not admit a closed form solution
- Linear programming using, e.g. interior-point methods $\mathcal{O}(\text{poly}(N))$
- Use approximate, iterative solutions, e.g. ADMM [Boyed 2011].

A Hard time for Fast JL transforms

$$\boldsymbol{Rb} = \boldsymbol{RA} + \boldsymbol{Rn} + \boldsymbol{Ro}$$

- The sparsity property is missing from $\boldsymbol{Ro}$
- The energy of the nonzero values of $\boldsymbol{o}$ is spread across all d dimensions
- LAD is not appropriately anymore

Robust Randomized Linear Regression

Robust Linear Regression

Recall Veracity!

$$\boldsymbol{b} = \boldsymbol{A}\ \ \boldsymbol{x}_* + \boldsymbol{\eta}, \qquad \boldsymbol{\eta} = \boldsymbol{n} + \boldsymbol{o}$$

$$\hat{\boldsymbol{x}}_{LAD} = \arg\min_{\boldsymbol{x}\in\mathbb{R}^l} \|\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}\|_1$$

- Least Absolute Deviations do not admit a closed form solution
- Linear programming using, e.g. interior-point methods $\mathcal{O}(\text{poly}(N))$
- Use approximate, iterative solutions, e.g. ADMM [Boyed 2011].

Randomized Sampling is Still OK

- $\boldsymbol{Ro}$ is still sparse
- LAD can be applied
- Harder (at least in theory) to compute the approximate leverage scores

Randomized Sampling for LAD

$\ell^{(1)}$ leverage scores

- Recall the LS case: $\ell_i^{(2)} = \|\boldsymbol{U}_{i,.}\|_2^2$ where $\boldsymbol{U}$ could be *any* orthogonal base spanning the column space of $\boldsymbol{A}$.
- LAD regression case: Leverage scores: $\ell_i^{(1)} = \|\boldsymbol{U}_{i,.}\|_1$
- $\|\boldsymbol{U}_{i,.}\|_1$ is not invariant under rotation so, a well conditioned $\boldsymbol{U}$ need to be used
- Cauchy distributed variables / submatrices are needed.
- In practice, benefits over the $\ell^{(2)}$ construction are observed when N is way much larger than l

Reasoning behind our approach

Proposed Approach

- Apply a fast JL transform, $\underline{\boldsymbol{b}} = \boldsymbol{R}\boldsymbol{b}$, $\underline{\boldsymbol{A}} = \boldsymbol{R}\boldsymbol{A}$.
- Progressively clean the data from outliers in the reduced dimensional space
- Obtain final solution with ordinary LS.

In an ideal world...(I)

- Let $\Lambda \subset \{1, \cdots, N\}$ be the index set indicating the corrupted data. Assume Λ is known.
- Then $\boldsymbol{A}_{\Lambda^c,.}$, $\boldsymbol{b}_{\Lambda^c}$ are the outlier-free data.
- Ideal solution: $\hat{\boldsymbol{x}}_{LS} = \arg\min_{\boldsymbol{x}\in\mathbb{R}^l} \|\boldsymbol{R}\boldsymbol{b}_{\Lambda^c} - \boldsymbol{R}\boldsymbol{A}_{\Lambda^c,.}\boldsymbol{x}\|_2$
- Cleaning the compressed data directly in the low dim domain.

$$\boldsymbol{R}\boldsymbol{b}_{\Lambda^c} = \underline{\boldsymbol{b}} - \boldsymbol{R}_{.,\Lambda}\boldsymbol{b}_{\Lambda}$$
$$\boldsymbol{R}\boldsymbol{A}_{\Lambda^c,.} = \underline{\boldsymbol{A}} - \boldsymbol{R}_{.,\Lambda}\boldsymbol{A}_{\Lambda,.}$$

Reasoning behind our approach

In an ideal world...(II)

- Let randomized Hadamard transform be applied to the full data set

$$\underline{\boldsymbol{b}} = \underline{\boldsymbol{A}}\boldsymbol{x}_* + \underline{\boldsymbol{n}} + \boldsymbol{R}\boldsymbol{o}$$

  where $\underline{\boldsymbol{n}} = \boldsymbol{R}\boldsymbol{n}$.

- Assume that $\boldsymbol{x}_*$ can be estimated exactly. Then

$$\boldsymbol{z} = \boldsymbol{R}\boldsymbol{o} + \underline{\boldsymbol{n}} \qquad (1)$$

  where $\boldsymbol{z}$ is computed as $\underline{\boldsymbol{b}} - \underline{\boldsymbol{A}}\boldsymbol{x}_*$.

- Request: Is it possible to estimate the support of $\boldsymbol{o}$, in the reduced dimensional space, based on (1)?

- Indeed, this is a typical compressed sensing scenario.

$$\hat{\boldsymbol{o}} = \min_{\boldsymbol{o}}(\|\boldsymbol{z} - \boldsymbol{R}\boldsymbol{o}\|_2) \; s.t. \; \|\boldsymbol{o}\|_0 \leq K$$

- We only need to estimate the support (or a subset of it)

Reasoning behind our approach

Back in reality...

- Let randomized Hadamard transform is applied to the full data set

$$\underline{\boldsymbol{b}} = \underline{\boldsymbol{A}}\boldsymbol{x}_* + \underline{\boldsymbol{n}} + \boldsymbol{R}\boldsymbol{o}$$

  where $\underline{\boldsymbol{b}} = \boldsymbol{R}\boldsymbol{b}$, $\underline{\boldsymbol{A}} = \boldsymbol{R}\boldsymbol{A}$, $\underline{\boldsymbol{n}} = \boldsymbol{R}\boldsymbol{n}$.

- $\boldsymbol{x}_*$ is not known but $\boldsymbol{R}\boldsymbol{o}$ is likely to be Normal distributed. so

$$\hat{\boldsymbol{x}} = \arg\min_{\boldsymbol{x}\in\mathbb{R}^l} \|\underline{\boldsymbol{b}} - \underline{\boldsymbol{A}}\boldsymbol{x}\|_2$$

  where $\hat{\boldsymbol{x}} = \boldsymbol{x}_* + \boldsymbol{x}_e$. Then, $\boldsymbol{z} = \underline{\boldsymbol{b}} - \underline{\boldsymbol{A}}\hat{\boldsymbol{x}}$

$$\boldsymbol{z} = \boldsymbol{R}\boldsymbol{o} + \underline{\boldsymbol{n}}'$$

- Request: Estimate any part of the support of $\boldsymbol{o}$.
- Suggestion: Just use the CoSaMP proxy, $\boldsymbol{\psi} = \boldsymbol{R}^T\boldsymbol{z}$, $\Lambda = \text{Supp}(|\boldsymbol{\psi}, K)$

The full picture

Iterative Randomized Robust LS: Concept

- Compress data: $\underline{\boldsymbol{b}} = \boldsymbol{R}\boldsymbol{b}$, $\underline{\boldsymbol{A}} = \boldsymbol{R}\boldsymbol{A}$
- **Start Iterations**
- Get a tentative estimate $\hat{\boldsymbol{x}}$ via $\arg\min_{\boldsymbol{x}\in\mathbb{R}^l} \|\underline{\boldsymbol{b}} - \underline{\boldsymbol{A}}\boldsymbol{x}\|_2$
- Compute $\boldsymbol{\psi} = \boldsymbol{R}^T(\underline{\boldsymbol{b}} - \underline{\boldsymbol{A}}\hat{\boldsymbol{x}})$
- Define Λ as the set of indices of the K larger (in magnitude) components of $\boldsymbol{\psi}$.

  Key remark: We are happy if Λ contains some, not necessarily K, outlier indices
- Exclude / Clear the data indexed in Λ from the compressed data set

$$\boldsymbol{R}\boldsymbol{b}_{\Lambda^c} = \underline{\boldsymbol{b}} - \boldsymbol{R}_{\cdot,\Lambda}\boldsymbol{b}_{\Lambda}$$

$$\boldsymbol{R}\boldsymbol{A}_{\Lambda^c,\cdot} = \underline{\boldsymbol{A}} - \boldsymbol{R}_{\cdot,\Lambda}\boldsymbol{A}_{\Lambda,\cdot}$$

  Key remark: Note that some healthy data might be omitted as well.
- **Return** to hopefully get an improved $\hat{\boldsymbol{x}}$ **or stop**

Computational complexity analysis

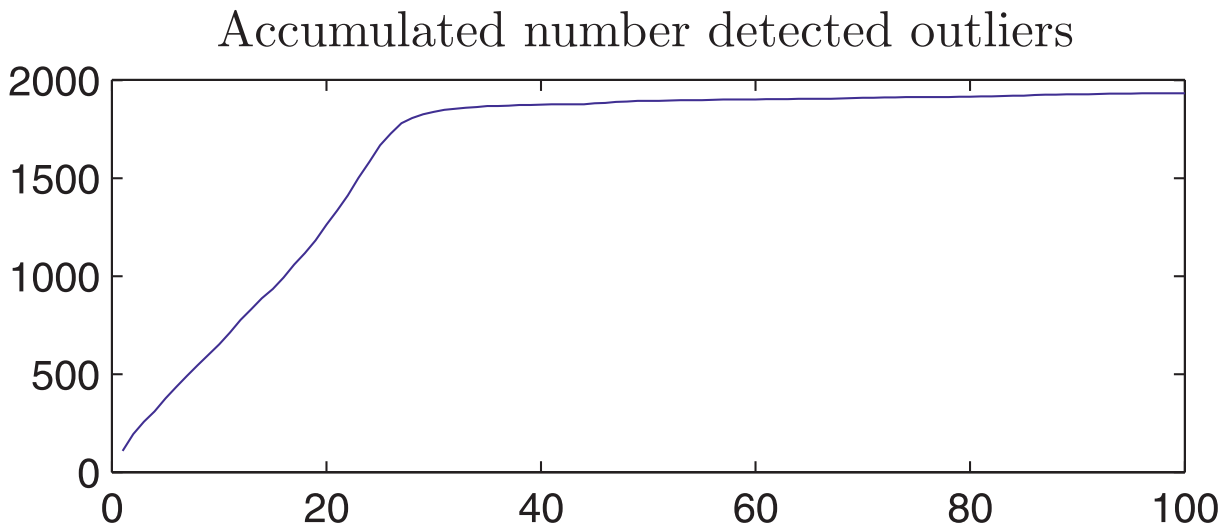
Proposed

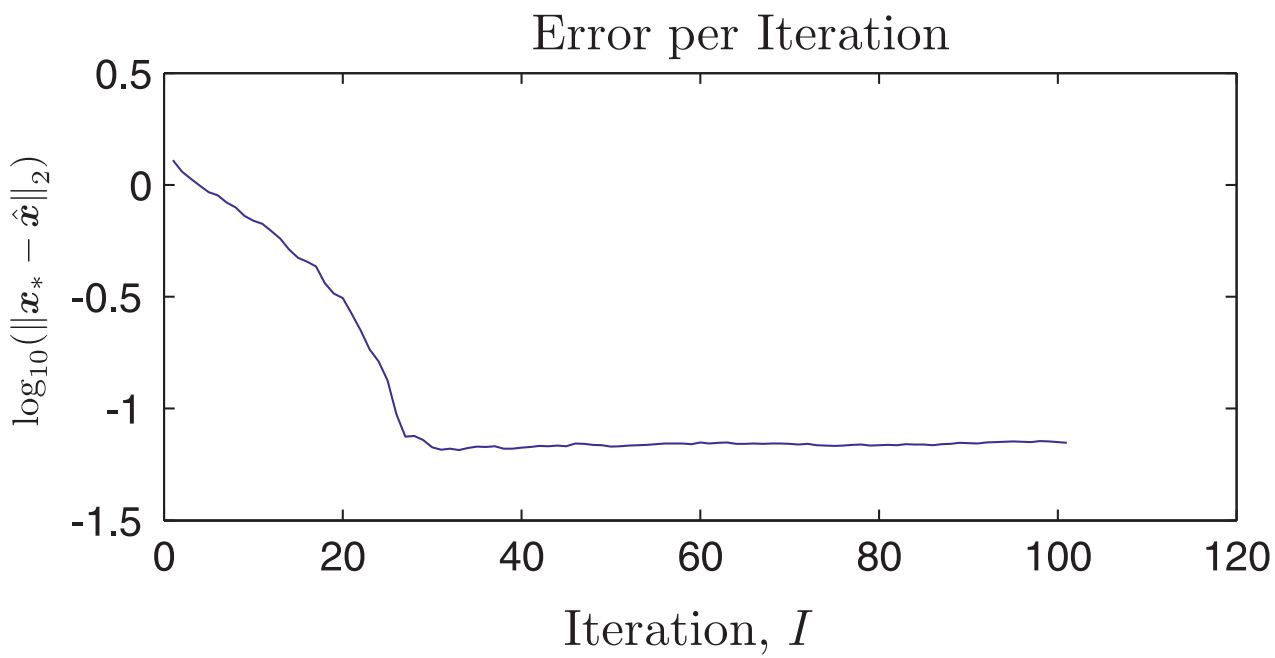
- Once: $\mathcal{O}((l+1)N\log d)$
- Per iteration: $\mathcal{O}(dl^2) + d(l+1) + \mathcal{O}(Nd) + \mathcal{O}(N) + (dKl)$

Random Sampling

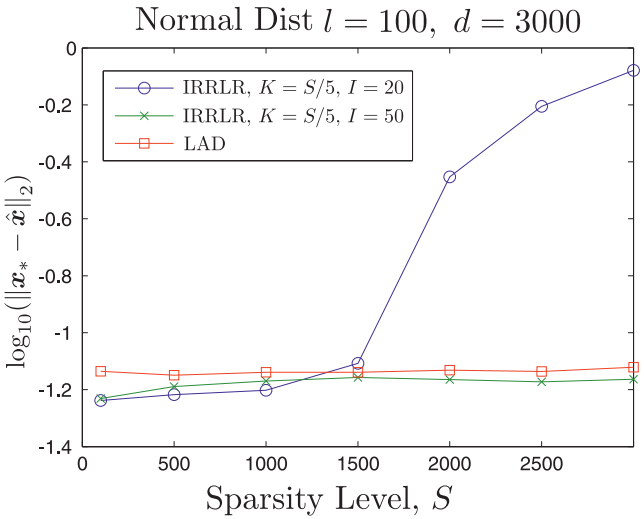
- For the leverage Scores: $\mathcal{O}((l+1)N\log r_1 + lNr_2 + r_1 l^2 + r_2 l^2)$
- $r_1 = d$ and $r_2 = \mathcal{O}(\log l)$
- For LAD: poly(d)

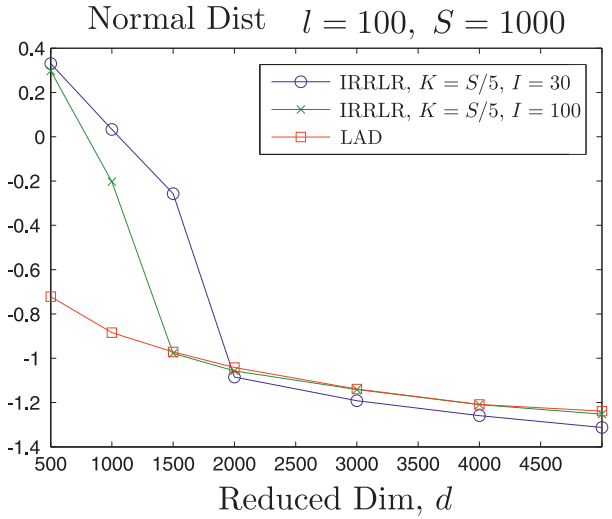
Some Results

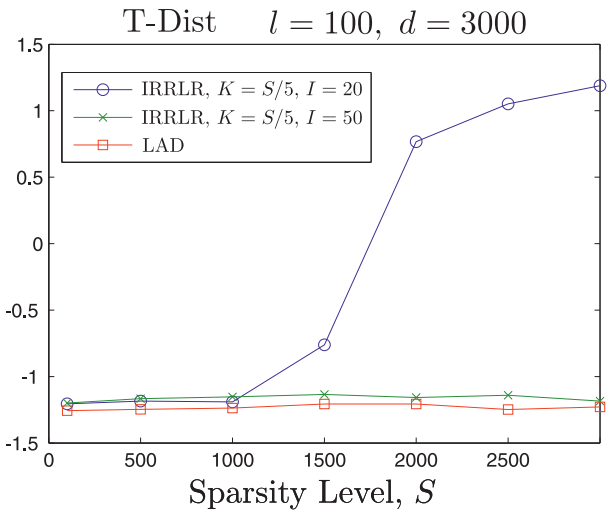


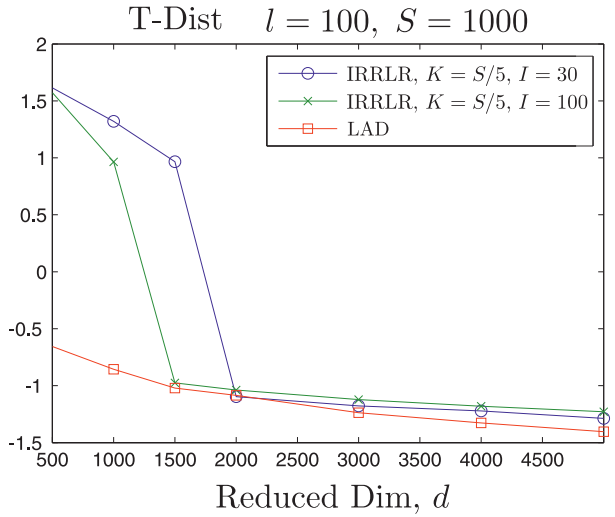


Some Results









Randomized Methods for Low Rank approximation

Sampling the column space is the key...

- Task: Let $\boldsymbol{A} \in \mathbb{R}^{n \times m}$, . $\min_{\boldsymbol{X}:\text{rank}(\boldsymbol{X}=k)} \|\boldsymbol{A} - \boldsymbol{X}\|_F$
- **Randomized Projection based Range finder**
  - Generate matrix $\boldsymbol{R} \in \mathbb{R}^{m \times d}$
  - and compress: $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{R}$
  - some housekeeping: Replace $\boldsymbol{Y}$ with $\boldsymbol{Q}$ whose columns form an orthonormal basis for the range of $\boldsymbol{Y}$
- **SVD estimation in 3 steps**
- $\boldsymbol{B} = \boldsymbol{Q}^T \boldsymbol{A}$
- Compute low dimensional SVD: $\boldsymbol{B} = \tilde{\boldsymbol{U}} \boldsymbol{\Sigma} \boldsymbol{V}^T$
- $\boldsymbol{U} = \boldsymbol{Q}\tilde{\boldsymbol{U}}$