

Fast algorithms for computation of Higher Order SVD (HOSVD): Randomized algorithms

Salman Ahmadi-Asl

May 2019

Outline

- **Randomized algorithms for low-rank matrix approximation**
- **Higher Order SVD (HOSVD) and related concepts**
- **Fast algorithms for HOSVD**



Randomized Algorithms in numerical linear and multilinear algebra

Randomization is a framework for performing numerical linear and multilinear algebra tasks more efficiently.

Matrix multiplication.

Overdetermined linear systems.

Overdetermined least squares.

Nonnegative least squares.

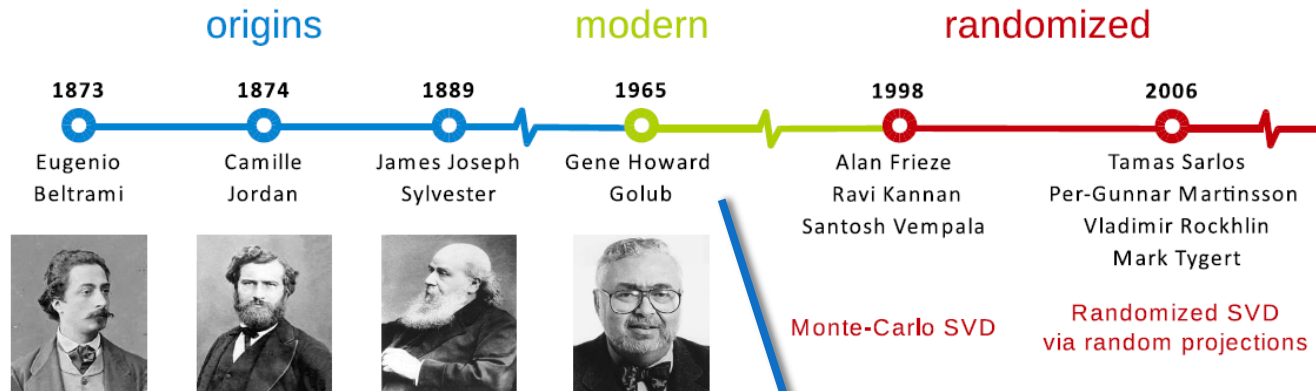
Preconditioned least squares.

•
•
•

We focus on low-rank matrix-tensor approximation



History of the evolution of randomized algorithms



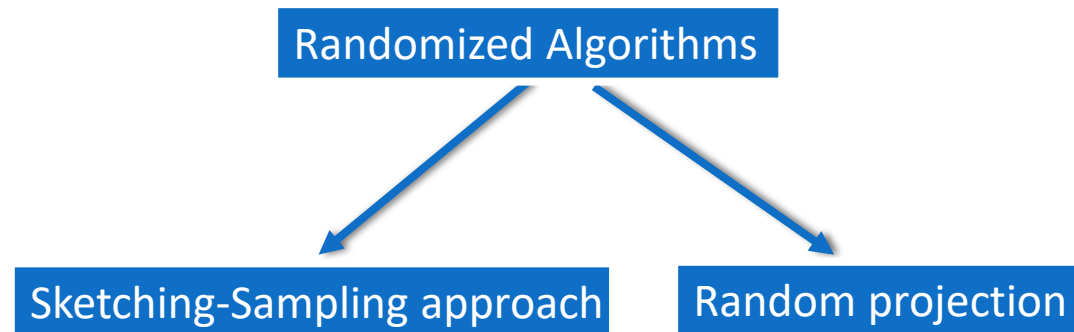
Picture was taken from:
<https://arxiv.org/pdf/1608.02148.pdf>

Krylov subspace algorithms: take advantage of structure of matrices and performing matrix-vector efficiently but **how about unstructured and dense data?**

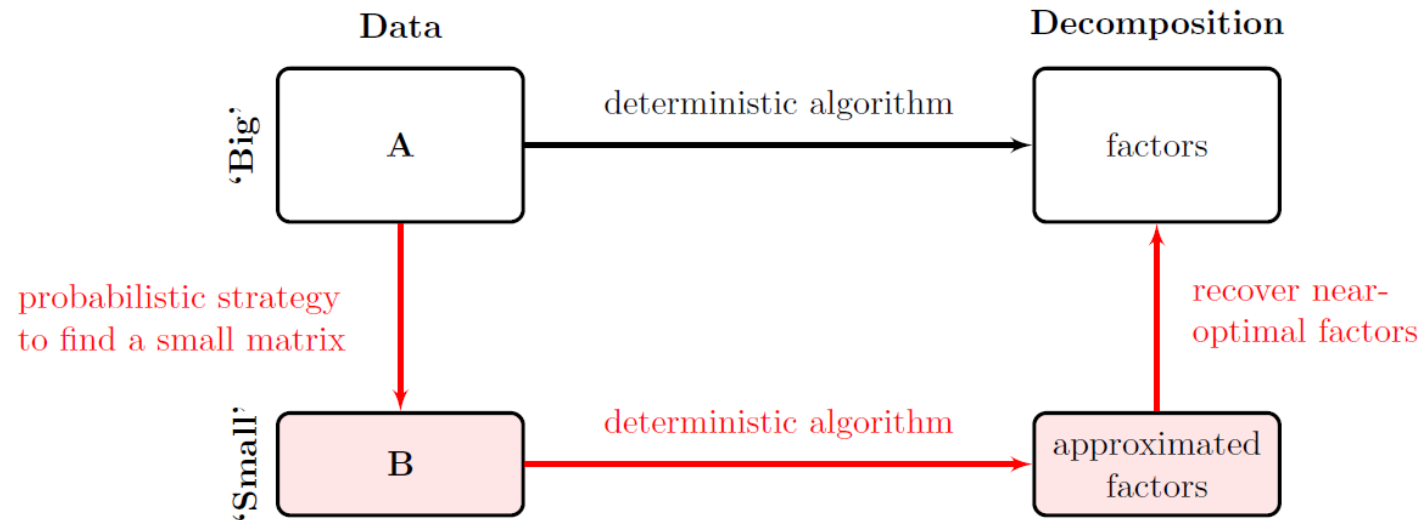
Also they need to pass the data $O(N)$ times!



Two main categorizes of the randomized low-rank approximation algorithms



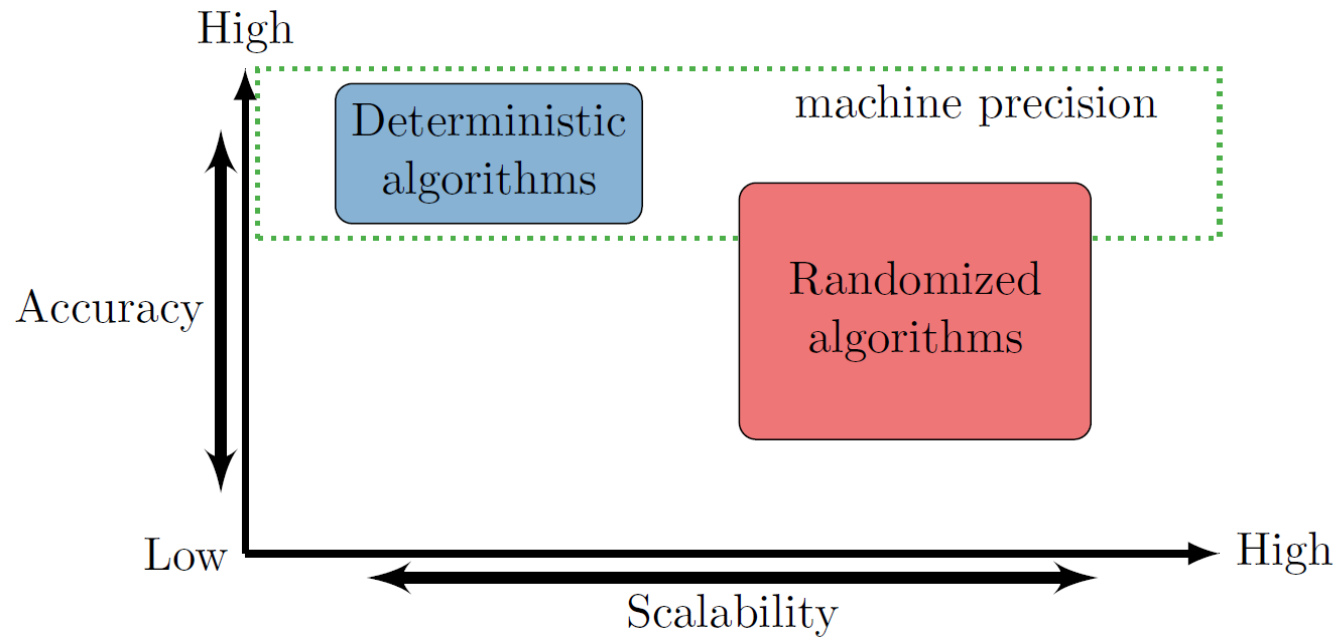
Randomized Algorithms for low rank matrix approximation



Picture was taken from:
<https://arxiv.org/pdf/1703.09074.pdf>



Randomized Algorithms for low rank matrix approximation



Picture was taken from:
<https://arxiv.org/pdf/1608.02148.pdf>

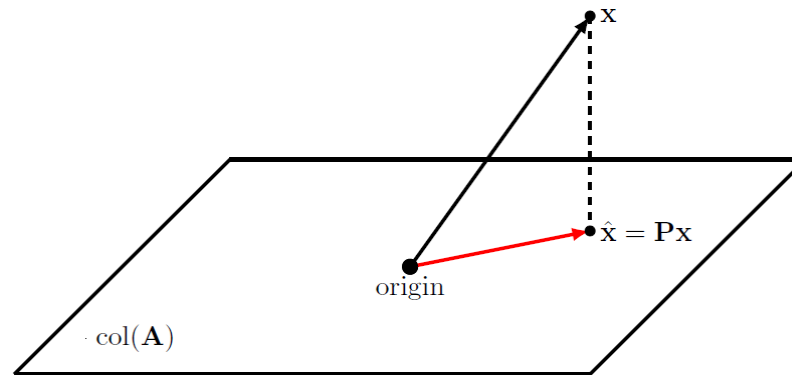


Randomized Algorithms for low-rank matrix approximation

$$A \in \mathbb{R}^{I \times J}, \quad Q \in \mathbb{R}^{I \times R}, \quad \text{rank}(A) = R$$

$QQ^T A \approx A \Rightarrow Q$ is an orthonormal basis for the range of matrix A

$P = QQ^T$ Orthogonal projector onto the range space of matrix A



$$A \approx \underbrace{QQ^T}_P A$$



$$B \in \mathbb{R}^{R \times J}$$

Randomized Algorithms for low rank matrix approximation

Algorithm 1: Randomized SVD algorithm

Input : Matrix $\mathbf{A} \in \mathbb{R}^{I \times J}$ and target rank R ;

Output: $\mathbf{U} \in \mathbb{R}^{I \times R}$, $\mathbf{S} \in \mathbb{R}^{R \times R}$, $\mathbf{V} \in \mathbb{R}^{J \times R}$

- 1 Generate a random matrix $\mathbf{\Omega} \in \mathbb{R}^{J \times R}$ with prescribed probability distribution ;
 - 2 Form $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$;  **One pass**
 - 3 Compute the QR decomposition of matrix $\mathbf{Y} = \mathbf{Q}\mathbf{R}$;
 - 4 Compute $\mathbf{B} = \mathbf{Q}^T \mathbf{A}$;  **One pass**
 - 5 Compute the full SVD of the matrix $\mathbf{B} = \mathbf{\bar{U}} \mathbf{\bar{S}} \mathbf{\bar{V}}^T$;
 - 6 $\tilde{\mathbf{U}} = \mathbf{Q} \mathbf{\bar{U}}$;
 - 7 Set $\mathbf{U} = \tilde{\mathbf{U}}(:, 1 : R)$, $\mathbf{S} = \mathbf{\bar{S}}(:, 1 : R)$, $\mathbf{V} = \mathbf{\bar{V}}(:, 1 : R)$;
-

 A multiple-pass algorithm

Randomized Algorithms

Single-pass algorithms

Multiple-pass algorithms

Randomized Algorithms

Fixed-Rank algorithms

Fixed-Precision algorithms



A single-pass algorithm

Random matrices

$$A \in \mathbb{R}^{I \times J}, Y = A\Omega \in \mathbb{R}^{I \times R} \quad \text{and} \quad W = \Psi A \in \mathbb{R}^{R' \times J}$$

$$(Q, \sim) = qr(Y, 0);$$

$$Q^T A \approx (\Psi Q)^\dagger W$$

$$W = \Psi(QQ^T A) + \Psi(A - QQ^T A) \approx (\Psi Q)(Q^T A)$$



$$Q^T A \approx (\Psi Q)^\dagger W$$



Solving a well-conditioned
least-squares problem



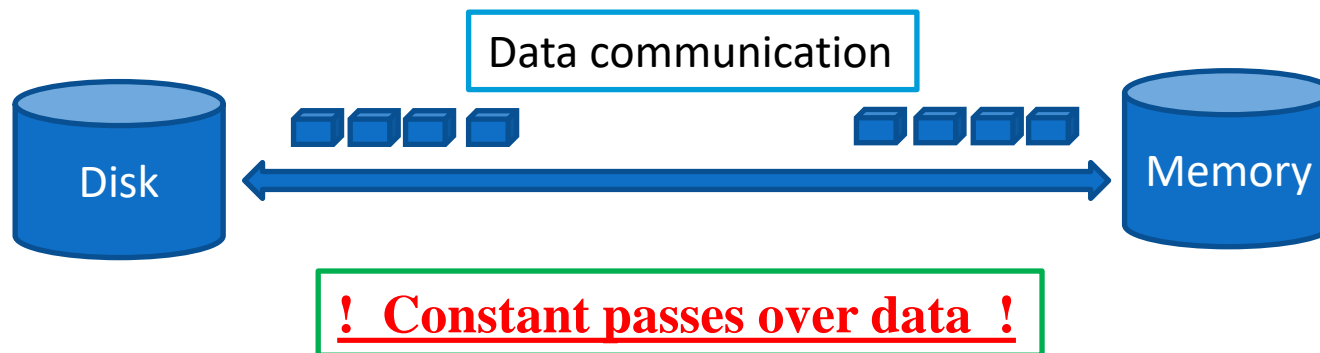
Memory requirement for matrices and tensors

Data	Memory requirement
$10^4 \times 10^4$	0.7451 GB
$10^4 \times 10^5$	7.4506 GB
$10^5 \times 10^5$	74.5058 GB
$500 \times 500 \times 500$	0.9313 GB
$500 \times 500 \times 500 \times 500$	465.6613 GB
$360 \times 480 \times 3 \times 2248$	8.6826 GB

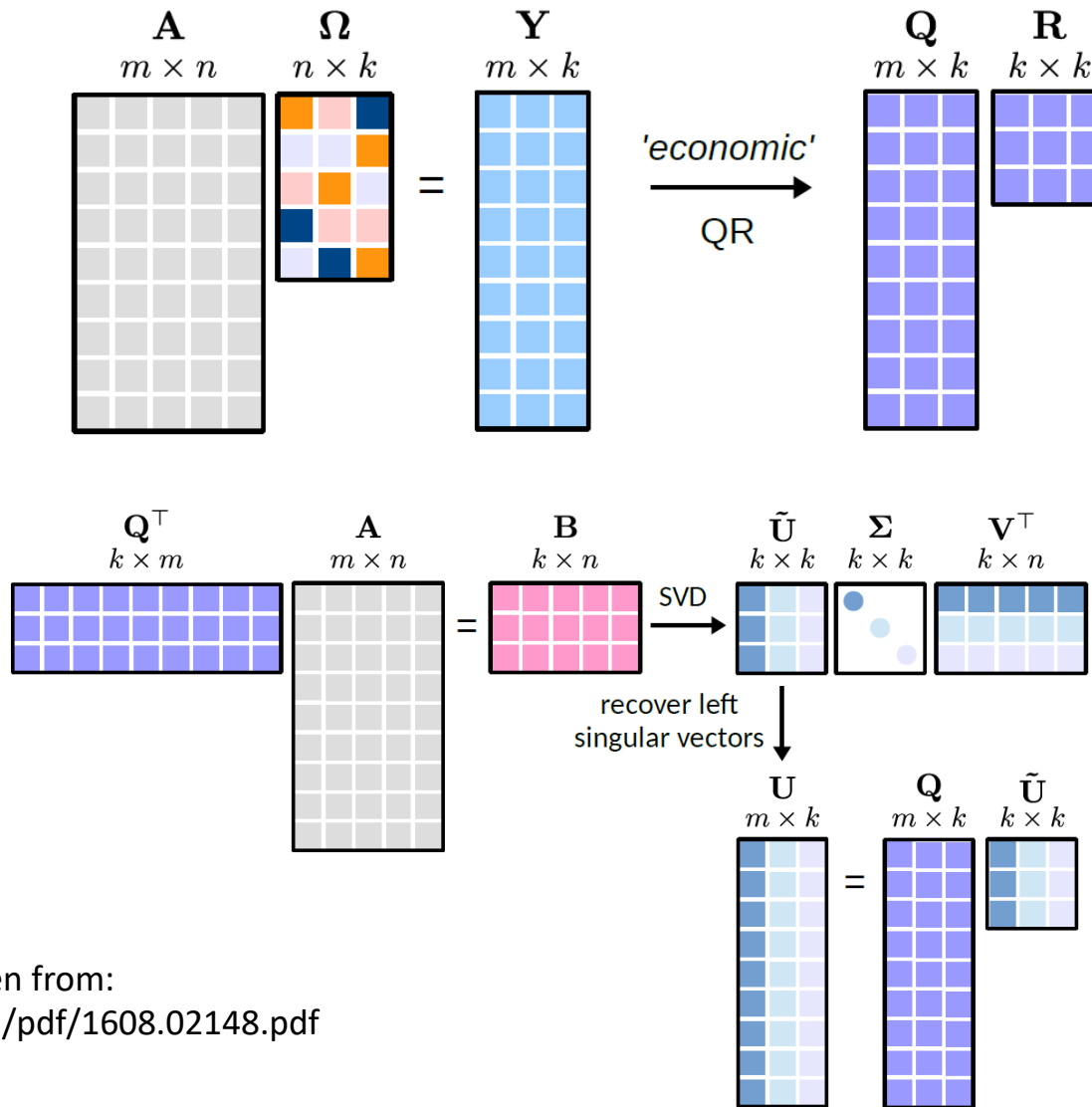
TABLE 1.2

Memory requirement for matrices and tensors of different sizes (GB \equiv Gigabyte).

Curse of
dimensionality



Graphical representation of random projection algorithm



Picture was taken from:
<https://arxiv.org/pdf/1608.02148.pdf>



Different types of random matrices can be utilized

- ✓ Gaussian distribution with zero mean and variance 1
- ✓ Uniform distribution over $(0,1)$ → Preserving data non-negativity
- ✓ Bernoulli distribution
- ✓ Rademacher distribution → Improving storage and arithmetic costs.
- ✓ Structured random matrices (sparse random matrices)
→ Preserving data sparsity and efficient matrix-matrix multiplications

In general, we have not seen any significant differences among different kinds of distributions **for our test and uses datasets.**



Two tricks for improving the accuracy of randomized algorithms

Oversampling strategy: Here more columns are used in the random projection step, to better capture the range of the matrix. For example, instead of R , we can use $R+P$ columns. For more accurate solution, $R+2P$ should be used.

Power iteration scheme: Here the original data matrix is replaced with a new one with the same right and left singular vectors but a faster decay rate of singular vectors.



The power iteration scheme can provide better approximations for those matrices whose singular values decay very slowly.

$$\|A - A_k\|_F^2 = \sum_{i=k+1}^R \sigma_i^2$$



Randomized Algorithms for low rank matrix approximation

It is better to apply Randomized SVD on the following modified matrices

$$\mathbf{A} = \mathbf{U}\mathbf{S}\mathbf{V}^T$$

$$\mathbf{A}^q \text{ or } \mathbf{A}^{(q)} = (\mathbf{A}\mathbf{A}^T)^q \mathbf{A}$$
$$\mathbf{A}^q = \mathbf{U}\mathbf{S}^q\mathbf{V}^T$$
$$\mathbf{A}^{(q)} = \mathbf{U}\mathbf{S}^{2q+1}\mathbf{V}^T$$

Algorithm 2: Computing $\mathbf{A}^{(q)}\mathbf{\Omega}$ in an efficient way

Input : Matrix $\mathbf{A} \in \mathbb{R}^{I \times J}$ and a target rank R ;

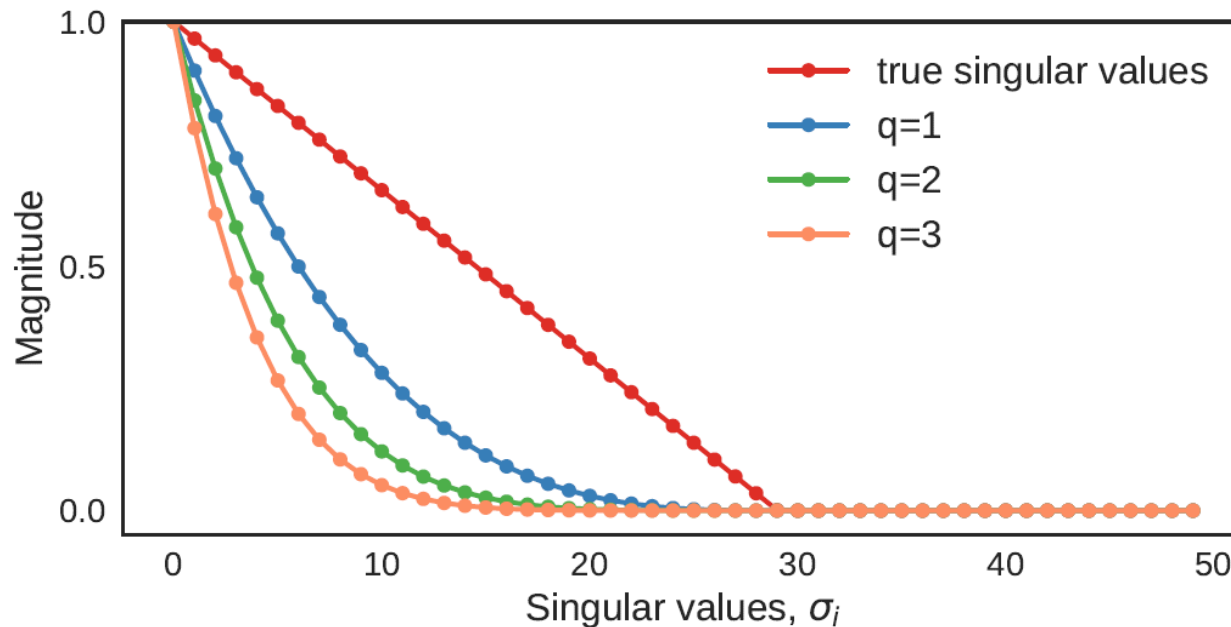
Output: $\mathbf{Y} = \mathbf{A}^{(q)}\mathbf{\Omega}$;

- 1 Generate a random matrix $\mathbf{\Omega} \in \mathbb{R}^{J \times R}$ with prescribed probability distribution;
 - 2 Form $\mathbf{Y} = \mathbf{A}\mathbf{\Omega}$;
 - 3 **for** $i = 1, \dots, q$ **do**
 - 4 Compute matrix $\mathbf{Y} = \mathbf{A}^T\mathbf{Y}$;
 - 5 Compute matrix $\mathbf{Y} = \mathbf{A}\mathbf{Y}$;
 - 6 **end**
-

QR or LU decompositions for make the algorithm more robust against round-off error



Impact of power iteration in accelerating the decay rate of singular values

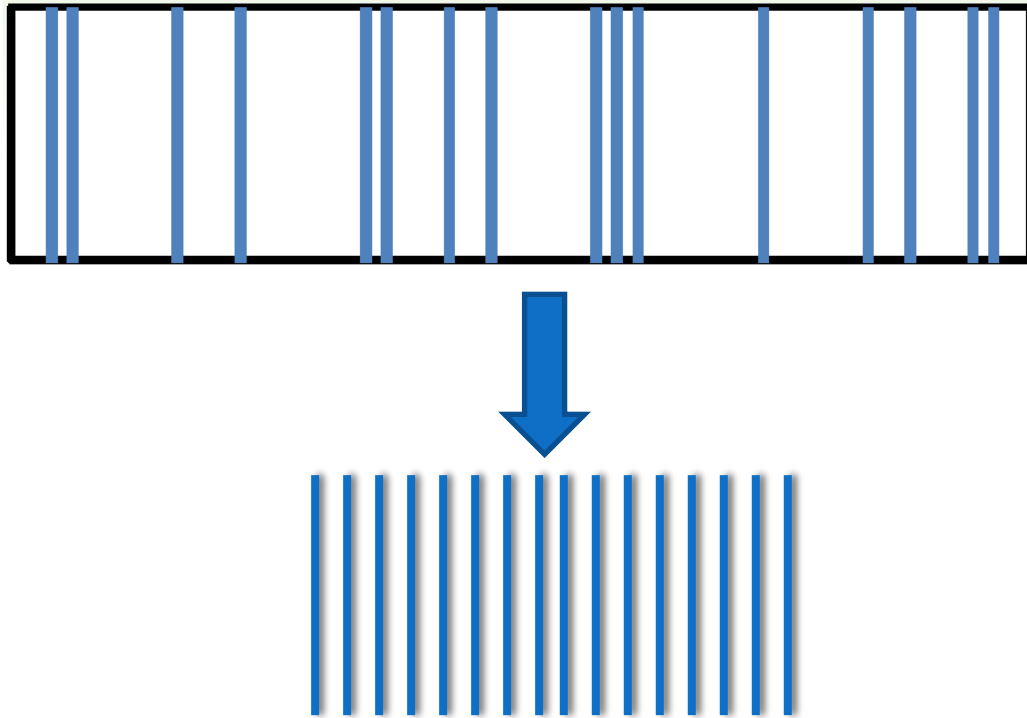


Picture was taken from:
<https://arxiv.org/pdf/1608.02148.pdf>

In practice $P=5$ and $q=1,2,3$ are enough for achieving a good accuracy.



Low rank approximation via sampling columns/rows



Sampling techniques as a low rank approximation

Here different types of probability distributions can be utilized to sample columns/rows such as uniform, length-squared sampling and also with/without replacement.

Uniform distribution

$$p_j = \frac{\|A(:, j)\|_2^2}{\|A\|_F^2}, \quad j = 1, \dots, J$$

$$\|A - B\|_F^2 \leq \|A - A_R\|_F^2 + \text{Term} \quad \longrightarrow \quad \text{Additive error norm}$$

Probability based on the leverage scores

$$p_j = \frac{l_j}{R}, \quad j = 1, \dots, J$$

$$l_j = \|\mathbf{V}_R(j, :)\|_2^2, \quad \mathbf{V}_R \in \mathbb{R}^{J \times R}$$

$$\|A - B\|_F^2 \leq \varepsilon \|A - A_R\|_F^2 \quad \longrightarrow \quad \text{Multiplicative error norm}$$

$$\sum_{j=1}^J l_j = \|\mathbf{V}_R\|_F^2 = \text{trace} \begin{pmatrix} \mathbf{V}_R^T \mathbf{V}_R \\ \mathbf{I}_R \end{pmatrix} = R$$



- If $p_j = \frac{\|\mathbf{A}(:,j)\|_2^2}{\|\mathbf{A}\|_F^2}$ and sampling is performed with replacement

$$\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^T \mathbf{A}\|_F^2 \leq \|\mathbf{A} - \mathbf{A}_R\|_F^2 + \sqrt{\frac{4R}{\delta c}} \|\mathbf{A}\|_F^2.$$

This sampling minimizes the variance for the error of the approximation.

- If $p_j = \frac{1}{J}$ and sampling is performed with replacement,

$$\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^T \mathbf{A}\|_F^2 \leq \|\mathbf{A} - \mathbf{A}_R\|_F^2 + \sqrt{\frac{4R}{\delta} \frac{J}{c} \sum_{j=1}^J \|\mathbf{A}(:,j)\|_2^4}.$$

- If $p_j = \frac{1}{J}$ and sampling is performed without replacement,

$$\|\mathbf{A} - \mathbf{Q}\mathbf{Q}^T \mathbf{A}\|_F^2 \leq \|\mathbf{A} - \mathbf{A}_R\|_F^2 + \sqrt{\frac{4R}{\delta} \left(\frac{J}{c} - 1\right) \sum_{j=1}^J \|\mathbf{A}(:,j)\|_2^4}.$$

\mathbf{A}_R is the best rank- R approximation of matrix \mathbf{A} in least-squares sense, for any $c \leq J$, with probability at least $1 - \delta$



Few practical comments concerning sampling strategy

For simplicity of theoretical derivation, selection with replacement is assumed but our simulations show that this is less efficient than sampling without replacement. More precisely, more columns are required to capture the range of a matrix if sampling with replacement is utilized especially for wide and fat matrices.

Our simulations on low-rank Gaussian and uniform random data and also real data including images and videos indicate the effectiveness of uniform sampling without replacement though we are not the first one reporting this fact!



Williams, C. K. I., & Seeger, M. (2000). Using the Nystrom method to speed up kernel machines. NIPS (pp. 682–688).

de Silva, V., & Tenenbaum, J. (2003). Global versus local methods in nonlinear dimensionality reduction. NIPS (pp. 705–712).

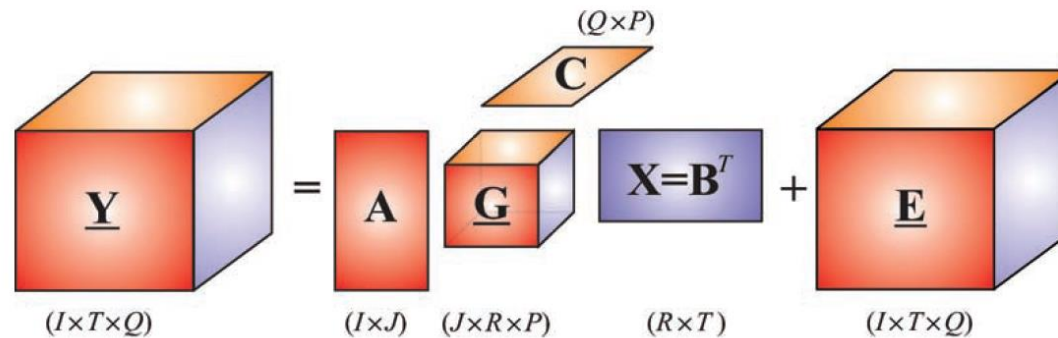
Kumar, S., Mohri, M., & Talwalkar, A. (2009). Sampling techniques for the Nystrom method. AISTATS (pp. 304–311). Clearwater Beach, Florida: JMLR: W&CP 5.

A practical comment concerning the sampling strategy

Although leverage score scheme provides approximations with a multiplicative error but they are of less practical interest due to the high computational cost of computing top singular vectors. Here it is suggested to randomized SVD to approximate them.



Higher Order SVD (HOSVD)



Picture was taken from:

A. Cichocki, R. Zdunek, A. H. Phan, S.-i. Amari, *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*, John Wiley & Sons, 2009.



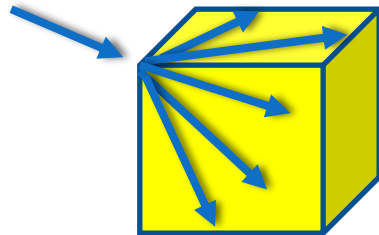
L DE LATHAUWER - 2000- [Cited by 3155](#)

Why HOSVD????

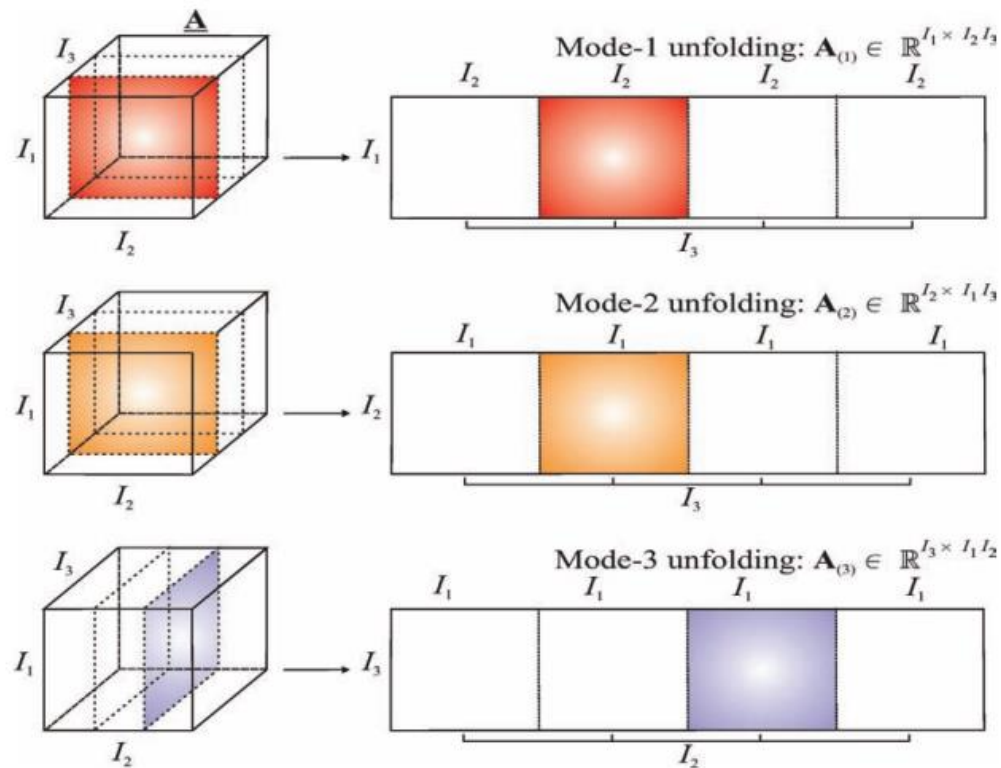
LR Tucker - 1966 - [Cited by 2696](#)

- ✓ Orthogonality property makes much more easier developing algorithms for Tensor decomposition.
- ✓ Explicit error representation of approximation In terms of singular values of unfolding matrices.
- ✓ Pseudo-diagonal property of the core tensors. **!Great Property!**

Density in these directions are reduced.



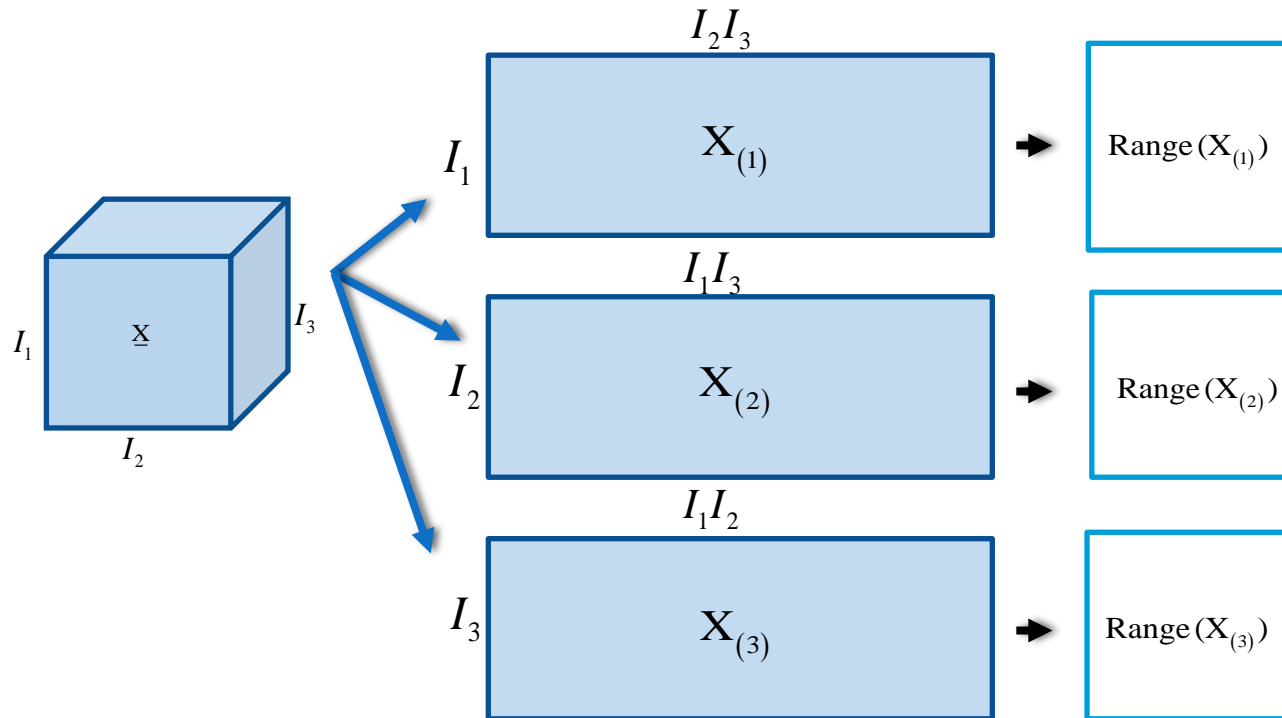
Unfolding matrices



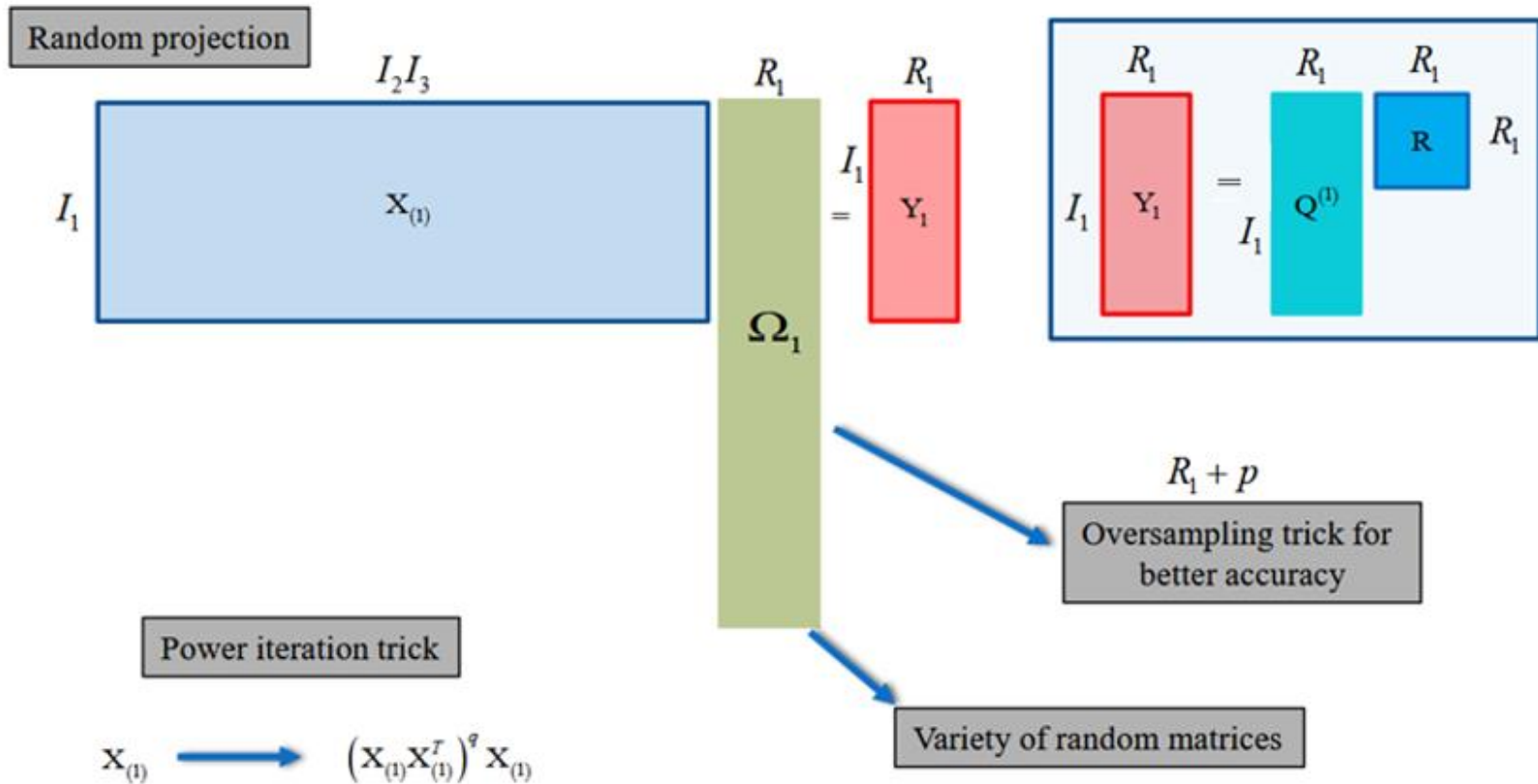
Randomized algorithms for low rank approximations are applied to the unfolding matrices to make the deterministic algorithms faster.



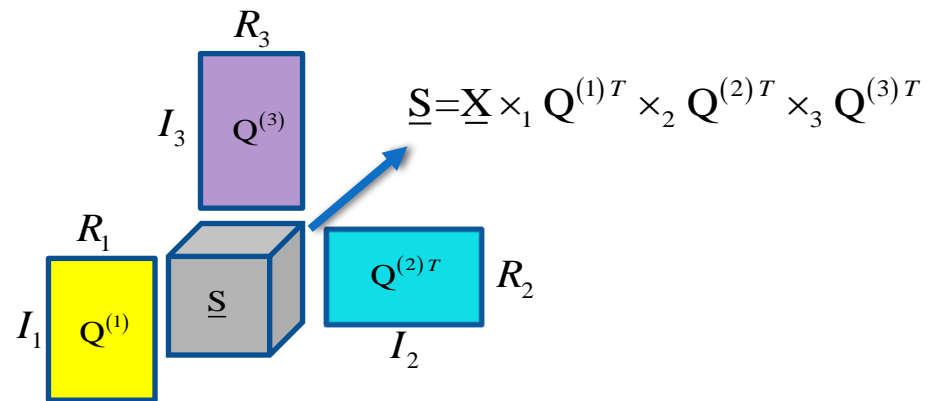
Randomized algorithms for HOSVD



Randomized algorithms for HOSVD

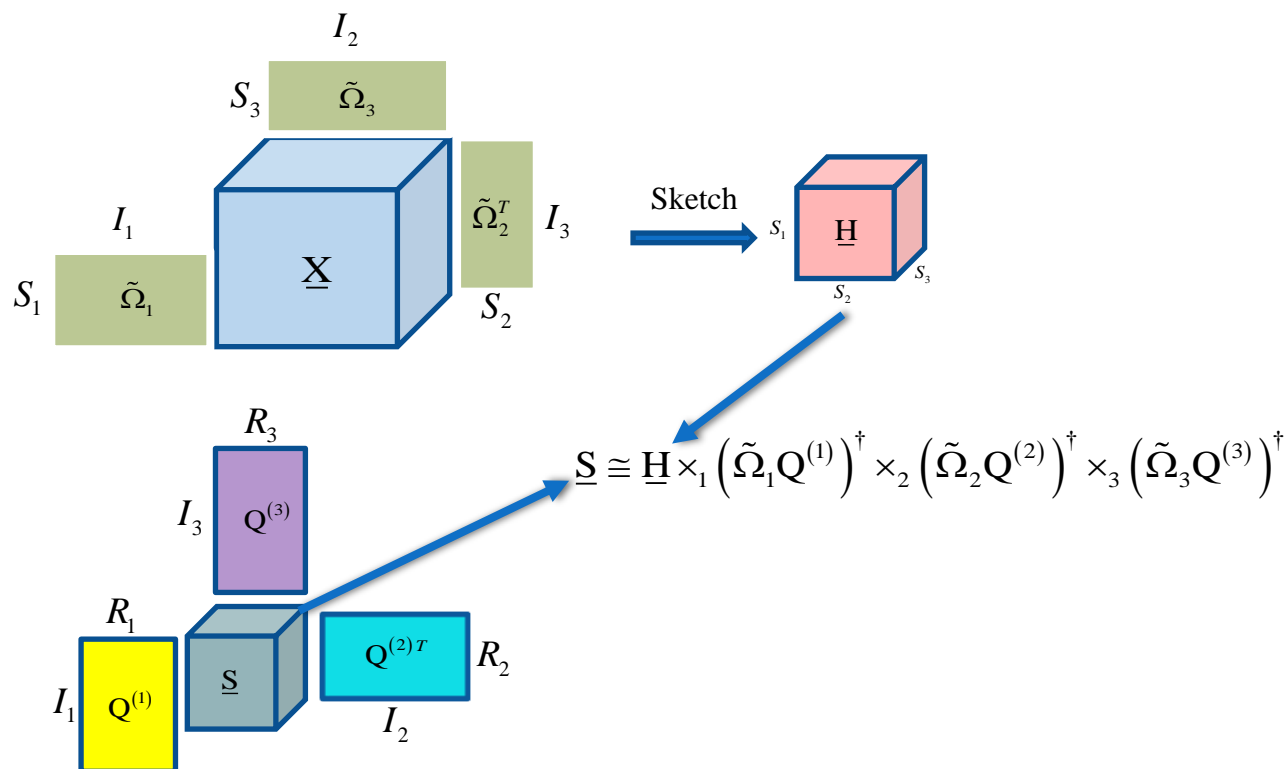


Randomized algorithms for HOSVD



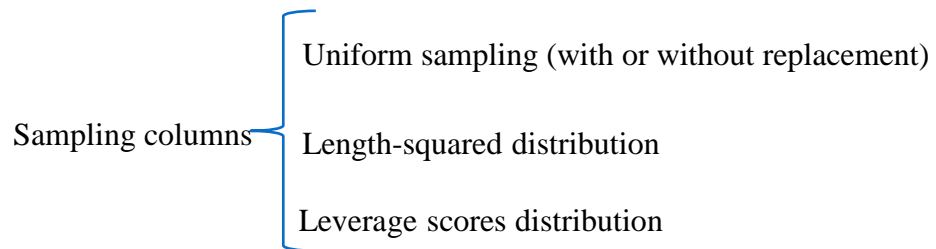
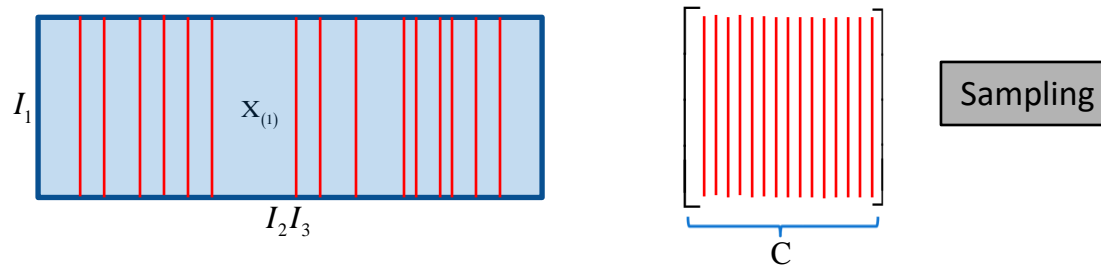
Randomized algorithms for HOSVD

A single-pass algorithm



Randomized algorithms for HOSVD

Factor estimation by sampling fibers in the unfolding matrices



Randomized algorithms for HOSVD

$$X_{(n)} \in \mathbb{R}^{I_n \times \prod_{m \neq n} I_m} \quad \begin{array}{c} \text{j-th column leverage score} \\ \nearrow \\ l_j = \|V_R(j, :)\|_2^2, \quad j = 1, 2, \dots, \prod_{m \neq n} I_m \end{array} \quad \sum_{j=1}^{\prod_{m \neq n} I_m} l_j = R$$



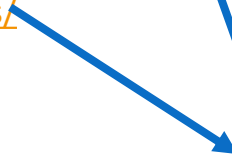
$$\text{i-th row leverage score} \quad \leftarrow l_i = \|U_R(i, :)\|_2^2, \quad i = 1, 2, \dots, I_n \quad \sum_{i=1}^{I_n} l_i = R$$

$$\text{Leverage score distribution for column selection} \quad \rightarrow \quad p_j = \frac{l_j}{R}, \quad j = 1, 2, \dots, \prod_{m \neq n} I_m$$

The maximum of the leverage score is called coherence of a matrix.



MATLAB codes for the randomized HOSVD

- **mlsvd_rsi**- Sequentially truncated MLSVD using randomized subspace  Random projection
- <https://github.com/OsmanMalik/tucker-tensorsketch> 
- <https://tsourakakis.com/mining-tensors/>  Sampling



Thanks for your attention!

