

Data compression by randomized algorithms

SSPM presentation

BOUAFIA Imène
COMPAIN Marie

Nantes Université

May 25th 2023



Table of contents

1 Randomized Linear Algebra

- Singular Value Decomposition (SVD)
- Randomized SVD
- Range Finders Algorithms
- Row extraction
- CUR decomposition

2 Tensor decomposition

- Generalities about tensors
- Tucker decomposition
- Quasi-optimality and HOOI
- Error Metrics
- Tucker approximation

Singular Value Decomposition

Definition :

$A \in \mathcal{M}_{m,n}(\mathbb{C})$, its singular value decomposition is :

$$A = U\Sigma V^*$$

- ⇒ $U \in \mathcal{M}_m(\mathbb{C})$ (*Left singular vectors*)
- ⇒ $V \in \mathcal{M}_n(\mathbb{C})$ (*Right singular vectors*)
- ⇒ $\Sigma \in \mathcal{M}_{m,n}(\mathbb{C})$, rectangular diagonal matrix.

Remark :

Result on SVD :

$$\min_{\text{rank}(X) \leq j} \|A - X\| = \sigma_{j+1}$$

Stage 1 :

Find a subspace E that captures the most information from A .

$$Q \in \mathbb{R}^{m \times k}$$

$$A \approx QQ^*A$$

Algorithm Randomized range finder

Require: A , k numerical range, p oversampling parameter

- 1: Draw a random test matrix $\Omega \in \mathbb{R}^{n \times (k+p)}$
 - 2: Compute $Y = A\Omega$
 - 3: Compute a QR -factorization : $(Q, _) = QR(Y)$.
 - 4: **return** Q
-

Stage 2 :

Using $Q \in \mathbb{R}^{m \times k}$ to compute the SVD on a small matrix B .

Algorithm RSVD

Require: A , the orthonormal matrix Q

Construct $B = Q^*A \in \mathbb{R}^{k \times n}$

Compute the SVD on B : $B = \tilde{U}\Sigma V^*$

Set $U = Q\tilde{U}$

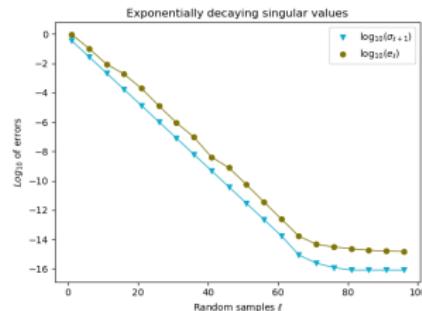
return U, Σ, V^*

Remark :

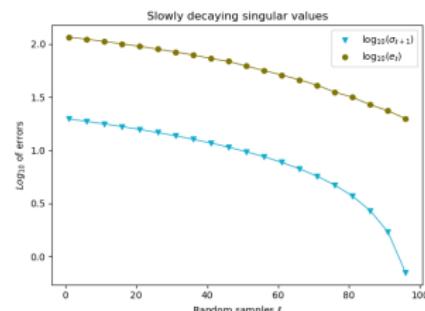
the columns of Q are the k dominant left singular vectors of A

Test

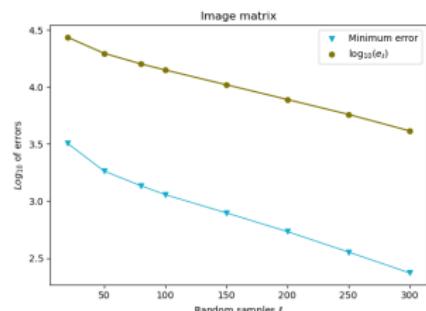
- σ_{k+1} : the theoretical minimum approximation error.
- $e_k = \|(I - Q^{(k)}Q^{(k)*})A\|$



(a) Exponential decay : 0.6^t



(b) Slow decay : $0.2t + 0.3$.



(c) Dense matrix (768, 1024).

Figure – Error of the randomized SVD on different type of matrices.

Adaptive Range Finder

Error Estimation

For $(\omega^{(i)})_{i=1}^r$, standard Gaussian vectors, $r=10$

$$\mathbb{P}(\|(I - QQ^*)A\| \leq 10 \sqrt{\frac{2}{\pi}} \max_{i=1,\dots,r} \|(I - QQ^*)Aw^{(i)}\|) = 1 - 10^{-r}$$

Given ϵ tolerance error we can construct $Q = Q^{(l)}$ incrementally such as

$$\|(I - Q^{(l)}Q^{(l)*})A\| \leq \epsilon$$

Condition to find l

$$\|\tilde{q}\| = \|(I - Q^{(l)}Q^{(l)*})A\omega\| \leq \frac{\epsilon \sqrt{\pi}}{10 \sqrt{2}}$$

Adaptive Range Finder

Algorithm Adaptive randomized range finder

Require: A, ε a tolerance, r an integer

Draw Gaussian random vectors ω_i of length n , $i = 1, \dots, r$

for $i = 1, \dots, r$ **do**

 Compute $y_i = A\omega_i$

end for

Take $j = 0$, and an empty basis matrix $Q^{(0)}$

while $\max_{i=1,\dots,r} \{\|y_{j+i}\|\} \geq \frac{\varepsilon \sqrt{\pi}}{10 \sqrt{2}}$ **do**

$j = j + 1$

 Overwrite $y_j = (I - Q^{(j-1)}(Q^{(j-1)})^*)y_j$

 Set $q_j = \frac{\tilde{y}_j}{\|\tilde{y}_j\|}$

 Form $Q^{(j)} = [Q^{(j-1)} q_j]$

 Draw a standard Gaussian vector ω_{j+r} of length n

$y_{j+r} = (I - Q^{(j)}(Q^{(j)})^*)A\omega_{j+r}$

for $i = j + 1, \dots, j + r - 1$ **do**

 Overwrite $y_i = y_i - q_j \langle q_j, y_i \rangle$

end for

end while

return Q

Adaptive Range Finder

ϵ	l	Approximation error
0.01	20	0.00012
0.1	14	0.0036

Test on a matrix of shape (600, 600).

ϵ	l	Approximation error
0.01	200	$1e^{-13}$
none	199	0.896

Test on a matrix of shape (200, 200).

ϵ	l	Approximation error
0.01	769	$1e^{-12}$
none	767	12.44

Test on a matrix of shape (768, 1024).

Table – required size basis l for ϵ

Approximation error : $\|(I - Q^{(l)} Q^{(l)*})A\|$

Power Iteration

In the case where the singular spectrum decays slowly :

$$((AA^*)^q)A = (U\Sigma^2 U^*)qA = U\Sigma^{2q+1}V^*$$

Hence, this operation forces the spectrum to decay rapidly.

Algorithm Randomized power iteration

Require: p the oversampling parameter, q a small integer

Draw a random $n \times (k + p)$ test matrix Ω

Form the matrix product $Y = (AA^*)^q A\Omega$

Compute a QR -factorization : $(Q, _) = QR(Y)$

return Q

Subspace iteration

floating-point arithmetic

All information associated with singular values that are small would be extinguished by rounding errors.

Algorithm Subspace iteration

Require: $m \times n$ matrix A , integers p and q

Draw a random $n \times (k+p)$ test matrix Ω

Form the matrix product $Y_0 = A\Omega$ and compute its QR factorization : $Y_0 = Q_0R_0$

for $j = 1, 2, \dots, q$ **do**

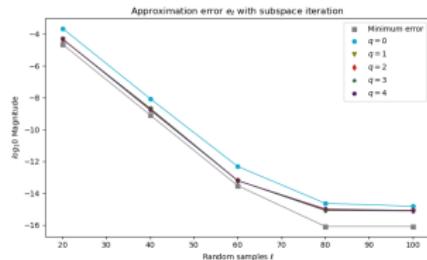
 Form $\tilde{Y}_j = A^*Q_{j-1}$ and compute its QR factorization $\tilde{Y}_j = \tilde{Q}_j\tilde{R}_j$

 Form $Y_j = A\tilde{Q}_{j-1}$ and compute QR factorisation $Y_j = Q_jR_j$

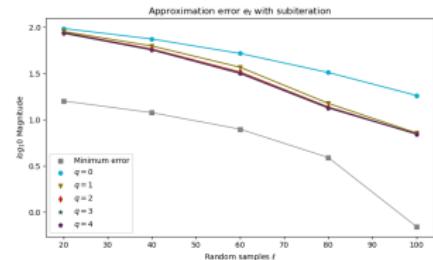
end for

return $Q = Q_q$ (via the QR factorization $Y = QR$)

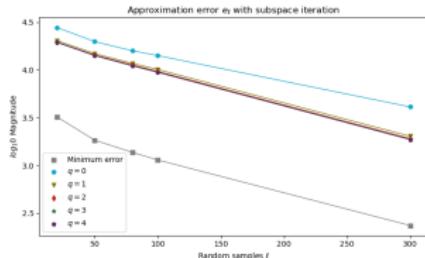
Test



(a) Exponential decay



(b) Slow decay



(c) Dense matrix

Figure – Subspace iteration algorithm with $q \in \{0, \dots, 4\}$

SVD via Row extraction

Interpolative Decomposition on rows

$A \in \mathbb{K}^{m \times n}$ matrix of rank k :

$$A = X A_{(J,:)}$$

Where

- ⇒ J , subset of k indices in $\llbracket 1, m \rrbracket$.
- ⇒ $A_{(J,:)} \in \mathbb{R}^{k \times n}$ represent J rows of A .
- ⇒ $X \in \mathbb{R}^{m \times k}$ and $X_{(J,:)} = I_k$

Algorithm SVD via Row Extraction

Require: A and Q such as $\|A - QQ^*A\| \leq \varepsilon$

Compute ID factorization $Q = XQ(J, :)$

Extract $A_{(J,:)}$ and compute the QR factorization $A_{(J,:)} = R^* W^*$

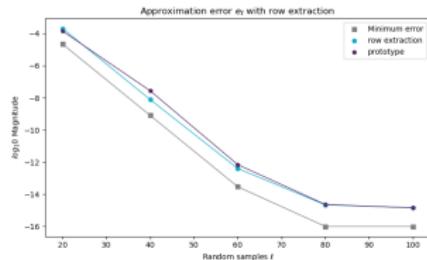
Form the product $Z = XR^*$

Compute the SVD on $Z : Z = U\Sigma\tilde{V}^*$

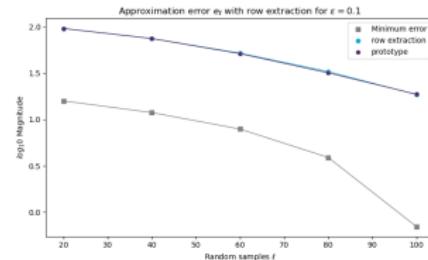
Form $V = W\tilde{V}$

return U, Σ and V

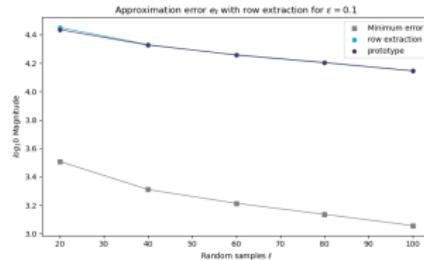
Test



(a) Exponential decay



(b) Slow decay



(c) Dense matrix

Figure – Row extraction scheme with $\epsilon = 0.1$

Image Reconstruction



(a) Original matrix



(b) RSVD with $k = 50$



(d) RSVD with $k = 200$



(c) RSVD with $k = 100$



(e) RSVD with $k = 300$

Figure – Reconstruction Image (768, 1024) with the RSVD

CUR decomposition

Definition

$A \in \mathcal{M}_{n \times n}(\mathbb{C})$, then :

$$A = CUR$$

- ⇒ $C \in \mathcal{M}_{m \times k}(\mathbb{K})$ such as : $C = A(:, J_s)$ (J_s subset of columns of A);
- ⇒ $R \in \mathcal{M}_{k \times n}(\mathbb{K})$ such as : $R = A(I_s, :)$ (I_s subset of rows of A);
- ⇒ $U \in \mathcal{M}_{k \times k}(\mathbb{K})$.

Algorithm for CUR decomposition

Algorithm Randomized CUR

Require: an $m \times n$ matrix A , a target rank k .

Compute the column-ID decomposition of $A : C, Z = \text{column_ID}(A, k)$.

Compute the QR decomposition of $C^T : S, P = \text{qr}(C^T)$.

Extract the top k row indices of $P : I = P(1 : k)$.

Extract the top k row from the input matrix : $R = A(I, :)$

Compute : $U = ZR^\dagger$

return A $k \times k$ matrix U , a $m \times k$ matrix C and a $k \times n$ matrix R .

Principle :

$$A = CZ \implies \text{Finding } U \text{ such as } Z = UR$$

Numerical results



(a) Original matrix



(b) CUR with $k = 50$



(d) CUR with $k = 200$



(c) CUR with $k = 100$



(e) CUR with $k = 300$

Figure – Results for different target ranks after using the CUR decomposition

Numerical results

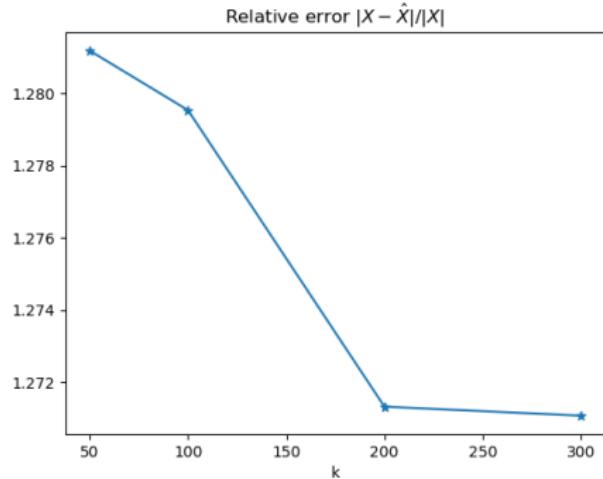


Figure – Relative errors $\frac{\|X - \hat{X}\|_F}{\|X\|_F}$ for each target rank k

Numerical results

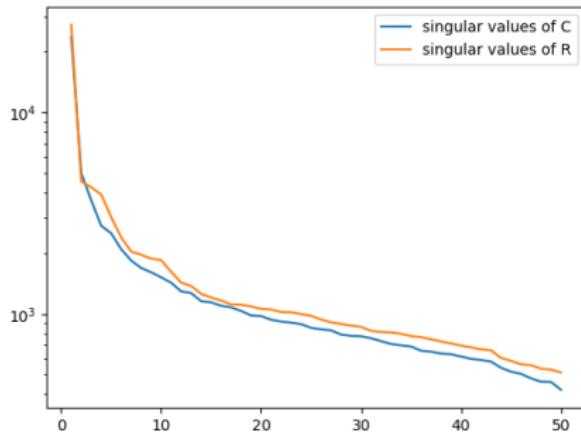


Figure – Singular values of C and R for $k = 50$

Tensors

Definition (order of a tensor)

The order of a tensor is its number of dimensions.

Definition (norm of a tensor)

$X \in \mathbb{R}^{I_1 \times \dots \times I_N}$. Its norm is defined by :

$$\|X\| = \sqrt{\sum_{i_1=1}^{I_1} \dots \sum_{i_N=1}^{I_N} x_{i_1 \dots i_N}^2}$$

Definition (n -rank of a tensor)

The n -rank of $\mathbf{X} \in \mathbb{R}^{I_1 \times \dots \times I_N}$, denoted $\text{rank}_n(\mathbf{X})$, is the column rank of $\mathbf{X}_{(n)}$.
⇒ If we denote $R_n = \text{rank}_n(\mathbf{X})$ for $n = 1, \dots, N$, then \mathbf{X} is rank- (R_1, \dots, R_N) .

Matricization of a tensor

Definition (mode- n matricization)

Let $X \in \mathbb{R}^{I_1 \times \dots \times I_N}$, $n \in \mathbb{N}$, $1 \leq n \leq N$. The element $x_{i_1 i_2 \dots i_N}$ will become $x_{i_n j}$ where :

$$j = 1 + \sum_{\substack{k=1 \\ k \neq n}}^N (i_k - 1) J_k, \text{ with } J_k = \prod_{\substack{m=1 \\ m \neq n}}^{k-1} I_m$$

Example : $X \in \mathbb{R}^{5 \times 3 \times 2}$ given by :

$$X_1 = \begin{bmatrix} 1 & 6 & 11 \\ 2 & 7 & 12 \\ 3 & 8 & 13 \\ 4 & 9 & 14 \\ 5 & 10 & 15 \end{bmatrix} \text{ and } X_2 = \begin{bmatrix} 16 & 21 & 26 \\ 17 & 22 & 27 \\ 18 & 23 & 28 \\ 19 & 24 & 29 \\ 20 & 25 & 30 \end{bmatrix}$$

$$\Rightarrow X_{(1)} = \begin{bmatrix} 1 & 6 & 11 & 16 & 21 & 26 \\ 2 & 7 & 12 & 17 & 22 & 27 \\ 3 & 8 & 13 & 18 & 23 & 28 \\ 4 & 9 & 14 & 19 & 24 & 29 \\ 5 & 10 & 15 & 20 & 25 & 30 \end{bmatrix}, X_{(2)} = \begin{bmatrix} 1 & 2 & 3 & \dots & 19 & 20 \\ 6 & 7 & 8 & \dots & 24 & 25 \\ 11 & 12 & 13 & \dots & 29 & 30 \end{bmatrix}, X_{(3)} = \begin{bmatrix} 1 & 2 & 3 & \dots & 15 \\ 16 & 17 & 18 & \dots & 30 \end{bmatrix}$$

Tensor multiplication

Definition (mode- n product)

Let $X \in \mathbb{R}^{I_1 \times \dots \times I_N}$ and $U \in \mathbb{R}^{J \times I_n}$. The n -mode product of X and U , denoted by $X \times_n U$, is a tensor of size $I_1 \times \dots \times I_{n-1} \times J \times I_{n+1} \times \dots \times I_N$, whose elements are :

$$(X \times_n U)_{i_1 \dots i_{n-1} j i_{n+1} \dots i_N} = \sum_{i_n=1}^{I_n} x_{i_1 i_2 \dots I_N} u_{j i_n}$$

Example : $U = \begin{bmatrix} 1 & 3 & 5 & 7 & 9 \\ 2 & 4 & 6 & 8 & 10 \end{bmatrix}$. The product $Y = X \times_1 U \in \mathbb{R}^{2 \times 3 \times 2}$ is given by :

$$Y_{(1)} = \begin{bmatrix} 95 & 200 & 345 \\ 110 & 260 & 410 \end{bmatrix} \text{ and } Y_{(2)} = \begin{bmatrix} 470 & 595 & 720 \\ 560 & 710 & 860 \end{bmatrix}$$

Tucker decomposition

Definition

Let $\mathcal{G} \in \mathbb{R}^{J_1 \times \dots \times J_1}$ and $A^{(i)} \in \mathbb{R}^{I_i \times J_i}$, for $i = 1, \dots, N$. Then the Tucker decomposition of $X \in \mathbb{R}^{I_1 \times \dots \times I_N}$ is :

$$X \approx \underbrace{\mathcal{G}}_{\text{core tensor}} \times_1 A^{(1)} \times_2 A^{(2)} \times_3 \dots \times_N A^{(N)} = [\![\mathcal{G}; A^{(1)}, \dots, A^{(N)}]\!]$$

Quasi-optimality and HOOI

Quasi-optimality

The method is *quasi-optimal* if

$$||X - \hat{X}||_F \leq d ||X - X_\bullet||_F$$

Where X_\bullet is the best rank- r Tucker approximation.

Algorithm HOOI

Require: $\mathcal{X}, (R_1, \dots, R_N)$

initialize $A^{(N)}$ for $n = 1, \dots, N$ using HOSVD

repeat

for $n = 1, \dots, N$ **do**

$\mathcal{Y} \leftarrow \mathcal{X} \times_1 A^{(1)T} \times_2 \dots \times_{n-1} A^{(n-1)T} \times_{n+1} A^{(n+1)T} \times_{n+2} \dots \times_N A^{(N)T}$

$A^{(n)} \leftarrow R_n$ leading singular vectors of $\mathcal{Y}_{(n)}$

end for

until fit ceases to improve or maximum iterations exhausted

$\mathcal{G} \leftarrow \mathcal{X} \times_1 A^{(1)T} \times_2 \dots \times_N A^{(N)T}$

return $\mathcal{G}, A^{(1)}, \dots, A^{(N)}$.

Error Metrics

Relative Error

Given $\mathcal{X} \in \mathbb{R}^{I_1 \times \dots \times I_N}$ and $\hat{\mathcal{X}}$, its approximation :

$$\text{Relative}(\mathcal{X}, \hat{\mathcal{X}}) = \frac{\|\mathcal{X} - \hat{\mathcal{X}}\|_F}{\|\mathcal{X}\|_F}$$

Not all tensors are Low-rank.

Regret metric

We consider the **HOOI** as the best approximation, then

$$\text{Regret}(\mathcal{X}, \hat{\mathcal{X}}) = \frac{\|\mathcal{X} - \hat{\mathcal{X}}\|_F - \|\mathcal{X} - \mathcal{X}_{HOOI}\|_F}{\|\mathcal{X}\|_F}$$

Algorithm HOSVD

Require: $X, (R_1, \dots, R_N)$
for $n = 1, \dots, N$ **do**
 $A^{(n)} \leftarrow R_n$ leading left singular vectors of the randomised SVD of $X_{(n)}$
end for
 $\mathcal{G} \leftarrow X \times_1 A^{(1)T} \times_2 \dots \times_N A^{(N)T}$
return $\mathcal{G}, A^{(1)}, \dots, A^{(N)}$.

Reduce the computational burden of the HOSVD \Rightarrow ST-HOSVD.

Algorithm ST-HOSVD

Require: X and target rank $r = (r_1, \dots, r_n)$
 $\mathcal{G} = X$
for $n = 1, \dots, N$ **do**
 $U_n, \Sigma_n, V_n = \text{TruncatedSVD}(\mathcal{G}^{(n)}, r_n)$ randomized SVD
 $\mathcal{G}^{(n)} \leftarrow \Sigma_n V_n^*$
end for
return $X_{ST-HOSVD} = [\mathcal{G}; U_1, \dots, U_N]$

Numerical results



(a) Original tensor of size (425,638,3)



(b) HOSVD with $R = (20, 20, 3)$



(d) HOSVD with $R = (100, 100, 3)$



(c) HOSVD with $R = (50, 50, 3)$



(e) HOSVD with $R = (200, 200, 3)$

Figure – Results for different target ranks after using the HOSVD

Numerical results

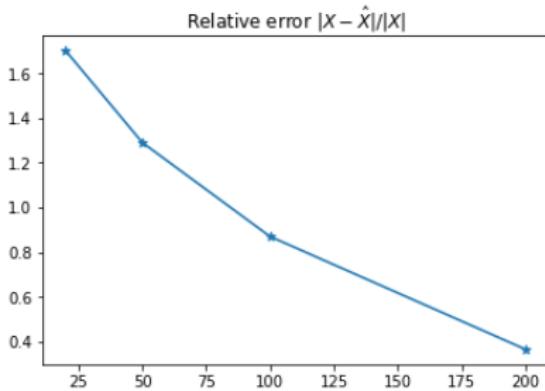


Figure – Relative errors $\frac{\|X - \hat{X}\|}{\|X\|}$ for each target rank k

Tools

Dimension Reduction maps (DRMs)

Let $k < N$, the **DRMs** is a matrix $\mathcal{M} \in \mathbb{R}^{k \times N}$

$$\mathbb{E}||\mathcal{M}u||_2^2 = \text{const.} ||u||_2^2 \quad \text{for all } u \in \mathbb{R}^k$$

Tensor Random Projection (TRP)

Given $S_i \in \mathbb{K}^{d \times m_i}$ for $i = 1, \dots, k$ independent DRMs, the TRP is defined as

$$\mathbf{S} := S_1 \odot S_2 \odot \dots \odot S_k \in \mathbb{K}^{d \times n}$$

Where $n = \prod_{i=1}^k m_i$.

Sketching

Tensor compression

Given $X \in \mathbb{R}^{I_1 \times \dots \times I_N}$ and a target rank $r = (r_1, \dots, r_N)$ we set the sketch size parameters :

- ⇒ $k = (k_1, \dots, k_N) \geq r$
- ⇒ $s = (s_1, \dots, s_N) \geq k$

Then draw independent, random DRMs :

$$\Omega_1, \dots, \Omega_N ; \Phi_1, \dots, \Phi_N$$

with $\Omega \in \mathbb{R}^{I_{(-n)} \times k_n}$ and $\Phi \in \mathbb{R}^{I_n \times s_n}$ for $n \in [N]$. We use these DRMs to compute :

$$V_n = X^{(n)} \Omega_n \in \mathbb{R}^{I_n \times k_n}, n \in [N]$$

$$\mathcal{H} = X \times_1 \Phi_1^T \cdots \times_N \Phi_N^T \in \mathbb{R}^{s_1 \times \dots \times s_N}$$

Tucker Sketch

Algorithm Tucker Sketch

Require: RDRM a function that generates a random RMD and a tensor $X \in \mathbb{R}^{I_1 \times \dots \times I_N}$

```
1: function Tucker_Sketch( $X, k, s$ )
2: for  $, \in 1, \dots, N$  do
3:    $\Omega_n = RDRM(I_{(-n)}, k_n)$ 
4:    $\Phi_n = RDRM(I_n, s_n)$ 
5:    $V_n = X^{(n)} \Omega_n$ 
6: end for
7:  $\mathcal{H} = X \times_1 \Phi_1^T \dots \times_N \Phi_N^T$ 
8: return  $(\mathcal{H}, V_1, \dots, V_N, \{\Phi_n, \Omega_n\}_n)$ 
```

Reducing storage for the Ω_n

Generate DRMs $A_n \in \mathbb{R}^{I_n \times k} \forall n \in \{1, \dots, N\}$ and then set

$$\Omega_n = A_1 \odot \dots \odot A_{n-1} \odot A_{n+1} \odot \dots \odot A_N \quad \forall n \in \{1, \dots, N\}.$$

Two Pass

The Two-Pass algorithm uses only the factors sketches V_n , we set then $s = 0$

Algorithm Two-Pass Sketch

Require: tensor X and sketch parameters k

```
( $\mathcal{H}, V_1, \dots, V_N, \{\Phi_n, \Omega_n\}_n$ ) = TuckerSketch( $X, k, 0$ )
for  $n \in \{1, \dots, N\}$  do
     $(Q_n, \_) = QR(V_n)$  (QR factorization)
end for
 $\mathcal{W} = X \times_1 Q_1^T \cdots \times_N Q_N^T \in \mathbb{R}^{k_1 \times \cdots \times k_N}$ 
return  $\hat{X} = [\![\mathcal{W}; Q_1, \dots, Q_N]\!]$ 
```

Remark :

$$[\![\mathcal{W}; Q_1, \dots, Q_N]\!] = \mathcal{W} \times_1 Q_1 \times_2 \cdots \times_N Q_N$$

Test



(a) Shape (1200, 768, 3)



(b) $k = (100, 50, 3)$



(d) $k = (600, 300, 3)$



(c) $k = (300, 100, 3)$



(e) $k = (900, 500, 3)$

Figure – Results for different target ranks after using the Two-Pass

One Pass

For the One-Pass algorithm, we need the core sketch \mathcal{H} defined earlier.

Algorithm One-Pass Sketch

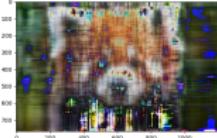
Require: tensor X and sketches parameters k and s

- 1: $(\mathcal{H}, V_1, \dots, V_N, \{\Phi_n, \Omega_n\}_n) = \text{Tucker_Sketch}(X, k, s)$
 - 2: **for** $n \in \{1, \dots, N\}$ **do**
 - 3: $(Q_n, _) = QR(V_n)$
 - 4: **end for**
 - 5: $\mathcal{M} = \mathcal{H} \times_1 (\Phi_1^T Q_1)^{-1} \dots \times_N (\Phi_N^T Q_N)^{-1}$
 - 6: **return** $\hat{X} = [\|\mathcal{M}, Q_1 \dots Q_n\|]$
-

Remark :

We set $s = 2k + 1$ for the test.

Test



(a) $k = (50, 20, 3)$



(c) $k = (300, 200, 3)$



(b) $k = (100, 80, 3)$



(d) $k = (600, 300, 3)$

Figure – Results for different target ranks after using the One-Pass

Test

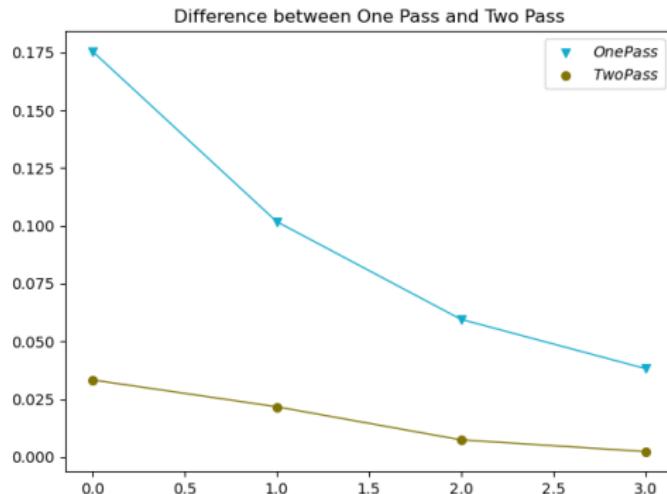


Figure – Approximation error with Two-Pass and One-Pass with the regret metric

Conclusion

Other methods possible :

- ⇒ Fixed-rank approximations ;
- ⇒ SRFT for dense matrix to compute the RSVD ;
- ⇒ Streaming methods : decomposing a tensor into : $A = H_1 + H_2 + \dots$

References

-  Bhatt, V., Kumar, S., and Saini, S. (2021).
 Tucker decomposition and applications.
materialstoday : proceedings, 46 :10787–10792.
-  Erichson, N. B., Voronin, S., Brunton, S. L., and Kutz, J. N. (May 2019).
 Randomized Matrix Decomposition Using R.
Journal of Statistical Software, 89(11).
-  Gundersen, G. (1-17 2019).
 Randomized Singular Value Decomposition.
<https://gregorygundersen.com/>.
-  Halko, N., Martinsson, P.-G., and Tropp, J. A. (2011).
 Finding structure with randomness : Probabilistic algorithms for constructing approximate matrix decompositions.
SIAM Review, 53(2) :217–288.
-  J. A. Tropp, A. Yurtsever, M. U. and Cevher, V. (2019).
 streaming low-rank matrix approximation.
SIAM Journal on Scientific Computing (SISC).
-  Kolda, T. G. (2006).
 Multilinear operators for higher-order decompositions.
 Technical Report SAND2006-2081, Sandia National Laboratories.
-  Kolda, T. G. and Bader, B. W. (2009).
 Tensor Decompositions and Applications.
SIAM Review, 51(3) :455–500.
-  Martinsson, P.-G. (October 2017).
 Randomized methods for matrix computations.
 Technical report.
-  Martinsson, P.-G. and Tropp, J. A. (2020).
 Randomized numerical linear algebra : Foundations and algorithms.
Acta Numerica, pages 403–572.
-  Sun, Y., Guo, Y., Luo, C., Tropp, J., and Udell, M. (2020).
 Low-Rank Approximation of a Tensor From Streaming Data.
SIAM Journal on Mathematics of Data Science, pages 1123–1150.
-  Wang, M. (12-5 2017).
 Vectorization, Kronecker Product, and Khatri-Rao Product.
<https://research.wmz.ninja/>.
-  Woodruff, D. P. (2014).
 Sketching as a Tool for Numerical Linear Algebra.
Foundations and Trends in Theoretical Computer Science, 10(1-2) :1–157.