Cours 10
Sparse matrix factorization

Elisa Riccietti   Théo Mary

LIP - ENS Lyon

December 20, 2024

Sparse matrix factorization

Given a dense matrix A, find *multiple* factors $S^{(1)}, S^{(2)}, \ldots, S^{(J)}$ such that:

$$A \approx S^{(1)} S^{(2)} \ldots S^{(J)}$$

where $S^{(i)}$ are *sparse* matrices.

Sparse matrix factorization

Given a dense matrix A, find *multiple* factors $S^{(1)}, S^{(2)}, \ldots, S^{(J)}$ such that:

$$A \approx S^{(1)} S^{(2)} \ldots S^{(J)}$$

where $S^{(i)}$ are *sparse* matrices.

Motivations

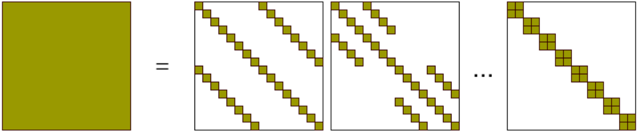
- *Fast* matrix vector products:

$$\underbrace{A}_{dense} \approx \underbrace{S^{(1)} S^{(2)} \ldots S^{(J)}}_{sparse} \quad \Rightarrow \quad Ax \approx S^{(1)}(S^{(2)}(\ldots(S^{(J)}x)))$$

- Reduce time + memory complexity

Applications

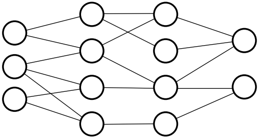
- Fast Fourier Transform, Fast Hadamard Transform, etc.



- Dictionary learning
  - $A = XY^\top$, A data, X a base (words in a dictionary), Y representation of each sample using the dictionary.

[S. Foucart, H. Rauhut, **A mathematical introduction to compressive sensing**, ANHA, 2013]

- Sparse (linear) neural networks (NN)
  - Toward interpretable NN?



[T. Dao & all. **Learning fast algorithms for linear transforms using butterfly factorizations**, PMLR, 2019]
[B. Chen & all. **Pixelated butterfly: Simple and efficient sparse training for neural network models**, PMLR, 2022]

A general formulation for sparse matrix factorization

Sparse Matrix Factorization Problem

Given a matrix A, $J \in \mathbb{N}$ and $\mathcal{E}_j$ some sets of sparse matrices, solve:

$$\min_{S^{(1)},\ldots,S^{(J)}} \|A - \prod_{j=1}^{J} S^{(j)}\|_F^2 \quad \text{subject to:} \quad S^{(j)} \in \mathcal{E}_j, \forall j \in \{1,\ldots,J\}$$

A general formulation for sparse matrix factorization

Sparse Matrix Factorization Problem

Given a matrix A, $J \in \mathbb{N}$ and $\mathcal{E}_j$ some sets of sparse matrices, solve:

$$\min_{S^{(1)},\ldots,S^{(J)}} \|A - \prod_{j=1}^{J} S^{(j)}\|_F^2 \quad \text{subject to:} \quad S^{(j)} \in \mathcal{E}_j, \forall j \in \{1, \ldots, J\}$$

- $\mathcal{E}$ = *family* of allowed supports / *sparsity patterns*:
  - $\mathcal{E}^k_{row} = \{S : |\mathrm{supp}(S_i)| \leq k\}$: at most k nonzero entries per row.
  - $\mathcal{E}^k_{col} = \{S : |\mathrm{supp}(S_i)| \leq k\}$: at most k nonzero entries per column.
  - $\mathcal{E}^k_{tot} = \{S : |\mathrm{supp}(S)| \leq k\}$: at most k nonzero entries in total.

- Known to be NP-hard (covers sparse PCA, sparse dictionary learning)

  [Malik, NP-hardness and inapproximability of sparse PCA, IPL, 2017]

  [S. Foucart, H. Rauhut, A mathematical introduction to compressive sensing, ANHA, 2013]

A general formulation for sparse matrix factorization

Sparse Matrix Factorization Problem

Given a matrix A, $J \in \mathbb{N}$ and $\mathcal{E}_j$ some sets of sparse matrices, solve:

$$\min_{S^{(1)},\ldots,S^{(J)}} \|A - \prod_{j=1}^{J} S^{(j)}\|_F^2 \quad \text{subject to:} \quad S^{(j)} \in \mathcal{E}_j, \forall j \in \{1, \ldots, J\}$$

- $\mathcal{E}$ = *family* of allowed supports / *sparsity patterns*:
  - $\mathcal{E}^k_{row} = \{S : |\mathrm{supp}(S_i)| \leq k\}$: at most k nonzero entries per row.
  - $\mathcal{E}^k_{col} = \{S : |\mathrm{supp}(S_i)| \leq k\}$: at most k nonzero entries per column.
  - $\mathcal{E}^k_{tot} = \{S : |\mathrm{supp}(S)| \leq k\}$: at most k nonzero entries in total.

- Known to be NP-hard (covers sparse PCA, sparse dictionary learning)

  [Malik, **NP-hardness and inapproximability of sparse PCA**, IPL, 2017]
  [S. Foucart, H. Rauhut, **A mathematical introduction to compressive sensing**, ANHA, 2013]

$\rightarrow$ A challenging problem, how to deal with it?

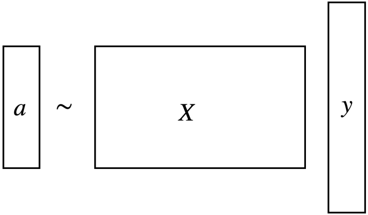
The simplest nontrivial setting

Two factors matrix factorization:

$$\text{Given } A, \min_{X,Y} \|A - XY^\top\|_F^2 \text{ subject to: } X, Y \text{ sparse matrices}$$

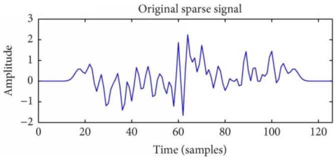
A classical related problem: sparse linear inverse problem

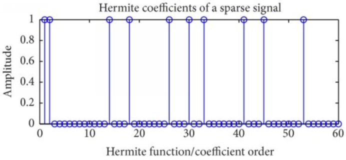
Given $a \in \mathbb{R}^m, X \in \mathbb{R}^{m \times n}, \min_{y \in \mathbb{R}^n} \|a - Xy\|_2^2$ subject to: $\|y\|_0 \leq s, s \ll n$



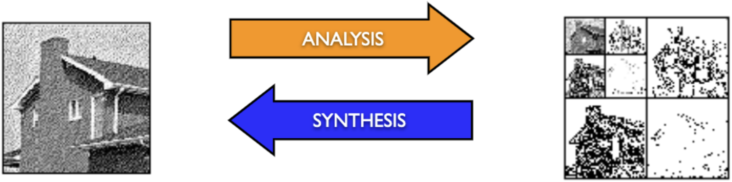
- Special case of sparse dictionary learning: compressed sensing.
- Compressed sensing is a signal processing technique for efficiently acquiring and reconstructing a signal, by finding solutions to underdetermined linear systems.
- A high-dimensional signal y ($n \geq m$) can be recovered with only a few measurements a, provided that the signal is sparse.
- Since not all signals satisfy this condition, it is crucial to find a sparse representation of that signal such as the wavelet transform

Audio signal:





- Images : wavelet transform



A classical related problem: sparse linear inverse problem

Given $a \in \mathbb{R}^m, X \in \mathbb{R}^{m \times n}, \min_{y \in \mathbb{R}^n} \|a - Xy\|_2^2$ subject to: $\|y\|_0 \leq s, s \ll n$

A classical related problem: sparse linear inverse problem

Given $a \in \mathbb{R}^m, X \in \mathbb{R}^{m \times n}, \min\limits_{y \in \mathbb{R}^n} \|a - Xy\|_2^2$ subject to: $\|y\|_0 \leq s, s \ll n$

1) Support identification

Finding a set $I \subseteq [\![n]\!]$ such that $|I| = s$.

2) Optimize coefficients inside support

$$\underset{y \in \mathbb{R}^n, \mathrm{supp}(y) \subseteq I}{\text{Minimize}} \|a - Xy\|_2^2$$

A classical related problem: sparse linear inverse problem

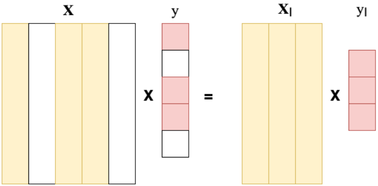
Given $a \in \mathbb{R}^m, X \in \mathbb{R}^{m \times n}, \min_{y \in \mathbb{R}^n} \|a - Xy\|_2^2$ subject to: $\|y\|_0 \leq s, s \ll n$

1) Support identification

Finding a set $I \subseteq [\![n]\!]$ such that $|I| = s$.

2) Optimize coefficients inside support

$$\text{Minimize}_{y \in \mathbb{R}^n, \text{supp}(y) \subseteq I} \|a - Xy\|_2^2 = \|a - X_I y_I\|_2^2$$



A classical related problem: sparse linear inverse problem

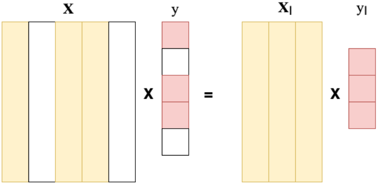
Given $a \in \mathbb{R}^m, X \in \mathbb{R}^{m \times n}, \min_{y \in \mathbb{R}^n} \|a - Xy\|_2^2$ subject to: $\|y\|_0 \leq s, s \ll n$

1) Support identification

Finding a set $I \subseteq [\![n]\!]$ such that $|I| = s$.

2) *Linear regression problem*

$$\underset{\tilde{y} \in \mathbb{R}^{|I|}}{\text{Minimize}} \quad \|a - X_I \tilde{y}\|_2^2$$



Two sub-problems of **two** factors matrix factorization

$$\underset{X,Y}{\text{Minimize}} \quad \|A - XY^\top\|_F^2 \quad \text{subject to: } X, Y \text{ sparse matrices}$$

Two sub-problems of **two** factors matrix factorization

$$\underset{X,Y}{\text{Minimize}} \quad \|A - XY^\top\|_F^2 \quad \text{subject to: } X, Y \text{ sparse matrices}$$

1) Support identification

Find *two* sets $S_X \subseteq [\![m]\!] \times [\![r]\!]$ and $S_Y \subseteq [\![n]\!] \times [\![r]\!]$ satisfying $\mathcal{E}$

Two sub-problems of **two** factors matrix factorization

$$\underset{X,Y}{\text{Minimize}} \quad \|A - XY^\top\|_F^2 \quad \text{subject to: } X, Y \text{ sparse matrices}$$

1) Support identification

Find *two* sets $S_X \subseteq [\![m]\!] \times [\![r]\!]$ and $S_Y \subseteq [\![n]\!] \times [\![r]\!]$ satisfying $\mathcal{E}$

2) Optimize coefficients inside support

$$\begin{aligned} \underset{X \in \mathbb{R}^{m \times r}, Y \in \mathbb{R}^{n \times r}}{\text{Minimize}} \quad & L(X, Y) = \|A - XY^\top\|_F^2 \\ \text{Subject to:} \quad & \mathrm{supp}(X) \subseteq S_X \\ & \mathrm{supp}(Y) \subseteq S_Y \end{aligned}$$

A comparison between two problems

	Linear inverse problem	*Sparse matrix factorization*
Pb	Minimize $\|a - Xy\|^2$, a, X are *known*, y is sparse	Minimize $\|A - XY^\top\|_F^2$, A is *known*, X, Y are sparse
1)	*Hard* due to exponential growth of combinations	
2)	*Easy* - Linear regression problem	??

A comparison between two problems

	Linear inverse problem	*Sparse matrix factorization*
Pb	Minimize $\|a - Xy\|^2$, a, X are *known*, y is sparse	Minimize $\|A - XY^\top\|_F^2$, A is *known*, X, Y are sparse
1)	*Hard* due to exponential growth of combinations	
2)	*Easy* - Linear regression problem	FSMF

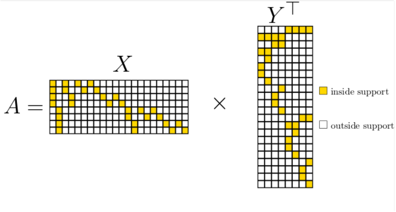
Fixed support matrix factorization

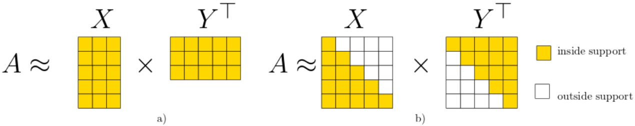
$$\begin{aligned} \underset{X\in\mathbb{R}^{m\times r}, Y\in\mathbb{R}^{n\times r}}{\text{Minimize}} \quad & L(X,Y) = \|A - XY^\top\|_F^2 \\ \text{Subject to:} \quad & \operatorname{supp}(X) \subseteq S_X \\ & \operatorname{supp}(Y) \subseteq S_Y \end{aligned} \tag{FSMF}$$

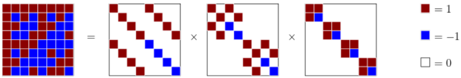
FSMF: motivation (I)

FSMF covers:

- Low rank matrix decomposition
- LU decomposition
- Hierarchical $\mathcal{H}$ and BLR matrices
- Butterfly factorization



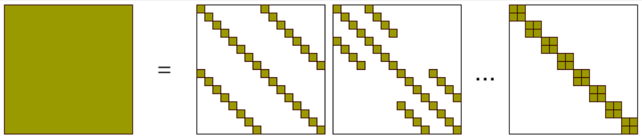




FSMF: motivation (II)

Neural network compression through butterfly structure

- It is expressive: the composition of matrices with a butterfly structure can accurately approximate any given matrix
- In neural networks faster training and inference time without harming the performance



  - $\log(N)$ factors
  - Each factor has 2 NNZ per row/column
  - From $O(N^2)$ to $O(N\log(N))$

[T. Dao & all. **Kaleidoscope: An efficient, learnable representation for all structured linear maps**, ICLR, 2020]
[B. Chen & all. **Pixelated butterfly: Simple and efficient sparse training for neural network models**, PMLR, 2022]
[T. Dao & all. **Learning Fast Algorithms for Linear Transforms Using Butterfly Factorizations** , PMLR, 2019]

What do we know about the problem?

1. The problem is *NP-hard*.

2. The problem has an essentially unique solution in the exact case

3. There is a family of *polynomially solvable* instances and an *efficient algorithm* to solve them

4. Some properties of the *landscape* of the function $L(X, Y) = \|A - XY^\top\|^2$ *under the support constraints* are known, which help to understand how well gradient descent tackles the problem of FSMF
   $\rightarrow$

[L. Le Magoarou and R. Gribonval, **Chasing butterflies: In search of efficient dictionaries**, ICASSP, 2015]

NP-hardness

FSMF can be reduced to rank-one matrix completion
$\rightarrow$ Sparse matrix factorization is NP-hard even with fixed support !

- In contrast to classical least squares
- In line with recent results on matrix factorization:
  - non-negative matrix factorization (NMF)
  - weighted low rank
  - matrix completion

[N. Gillis, F. Glineur, Low-rank matrix approximation with weights or missing data is NP-hard. SIAM JMAA, 2010]

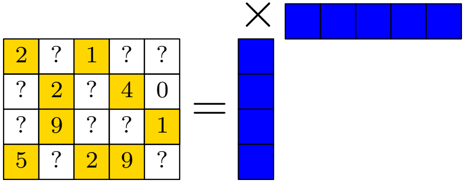
[S. A. Vavasis, On the complexity of nonnegative matrix factorization, SIOPT, 2010]

Matrix completion

Let $W \in \{0,1\}^{m \times n}$ be a binary matrix. Given $A \in \mathbb{R}^{m \times n}, s \in \mathbb{N}$, the matrix completion problem (MCP) is:

$$\underset{X \in \mathbb{R}^{m \times s}, Y \in \mathbb{R}^{n \times s}}{\text{Minimize}} \|A - XY^\top\|_W^2 = \|(A - XY^\top) \odot W\|^2. \qquad \text{(MCP)}$$

This problem is NP-hard even when $s = 1$



[N. Gillis, F. Glineur, Low-rank matrix approximation with weights or missing data is NP-hard. SIAM JMAA, 2010]
[R. Peeters, The maximum edge biclique problem is NP-complete, Discrete Appl Math, 131 (2000)].

NP-hardness of matrix completion with noise

Given a binary weighting matrix $W \in \{0,1\}^{m \times n}$ and $A \in [0,1]^{m \times n}$, the optimization problem

$$\underset{x \in \mathbb{R}^m, y \in \mathbb{R}^n}{\text{Minimize}} \|A - xy^\top\|_W^2 \tag{MCPO}$$

is called rank-one matrix completion problem (MCPO). Denote p^* the infimum of (MCPO) and let $\epsilon = 2^{-12}(mn)^{-7}$. It is NP-hard to find an approximate solution with objective function accuracy less than ϵ, i.e. with objective value $p \leq p^* + \epsilon$.

The following lemma gives a reduction from (MCPO) to (FSMF).

Lemma

For any $W \in \{0,1\}^{m \times n}$, there exist an integer r and two sets S_X and S_Y such that for all $A \in \mathbb{R}^{m \times n}$, (MCPO) and (FSMF) share the same infimum. The sets can be constructed in polynomial time. If one of the problems has a known solution that provides objective function accuracy ϵ, we can find a solution with the same accuracy for the other one in polynomial time.

Proof sketch.

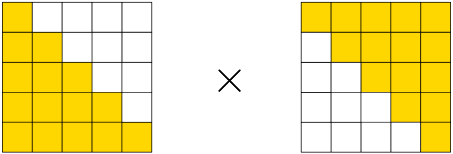
Up to a transposition, we can assume without loss of generality that $m \geq n$. Let $r = n + 1 = \min(m, n) + 1$. We define $S_X \in \{0,1\}^{m \times (n+1)}$ and $S_Y \in \{0,1\}^{n \times (n+1)}$ as follows:

$$S_{X\,i,j} = \begin{cases} 1 - W_{i,j} & \text{if } j \neq n \\ 1 & \text{if } j = n+1 \end{cases}, S_{Y\,i,j} \quad = \begin{cases} 1 & \text{if } j = i \text{ or } j = n+1 \\ 0 & \text{otherwise} \end{cases}$$

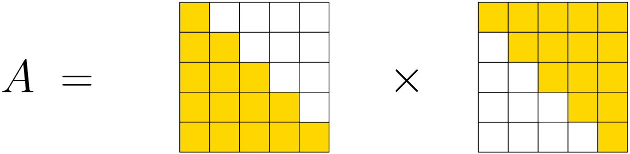
This construction can be made in polynomial time. We can then show that the two problems share the same infimum. [Q.-T. Le & all. 2023] □

- A special case of (FSMF): LU-decomposition:

$A =$



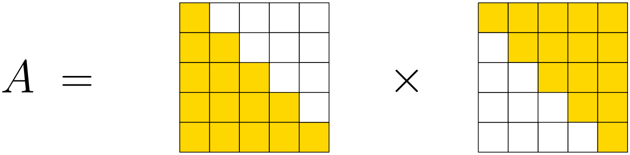
- A special case of (FSMF): LU-decomposition:



- There exist square matrices that **do not have** an exact LU decomposition.
- Any square matrix is the limit of a sequence of matrices having an LU decomposition.

LU decomposition and non-closedness

- A special case of (FSMF): LU-decomposition:



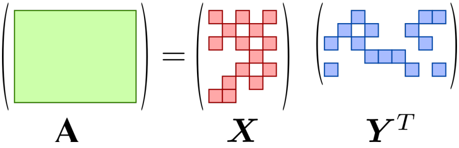
- There exist square matrices that **do not have** an exact LU decomposition.
- Any square matrix is the limit of a sequence of matrices having an LU decomposition.
- $\longrightarrow$ The set of matrices having LU decomposition is not closed
- $\longrightarrow$ For certain support constraints (S_X, S_Y) and matrices A, (FSMF) does not have an optimal solution.

Open problem : characterize the instances that admit a solution

Essential uniqueness

We consider the *exact case*:
Given a matrix A and a couple of feasible sets $S = (S_X, S_Y)$, our problem is:

$$\text{find } (X, Y) \text{ such that } A = XY^\top \text{ and } \operatorname{supp}(X, Y) \subseteq S \qquad \text{(EMF)}$$



Essential uniqueness

The solution (X, Y) to (EMF) is *essentially unique*[a] if any other solution $(\bar{X}, \bar{Y})$ is equivalent to (X, Y), i.e., it exists D invertible diagonal matrix such that $(\bar{X}, \bar{Y}) = (XD, YD^{-1})$. We write in this case $(\bar{X}, \bar{Y}) \sim (X, Y)$

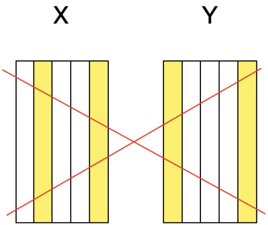
[a] we do not consider permutation ambiguities

Characterization of essential uniqueness

$S = (S_X, S_Y)$, $\mathcal{U}(S)$ the set of couples of essentially unique factors.

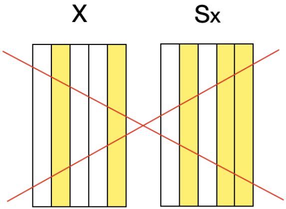
Lemma: necessary condition for identifiability

For any pair of supports S, we have: $\mathcal{U}(S) \subseteq IC_S \cap MC_S$.



$IC_S \rightarrow \texttt{colsupp}(\text{X}) = \texttt{colsupp}(\text{Y})$

$i \in \texttt{colsupp}(\text{X}) \implies X_i \neq 0$



$MC_S \rightarrow \texttt{colsupp}(\text{X}) = \texttt{colsupp}(S_X)$,

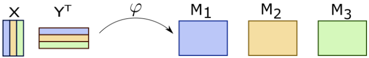
$\texttt{colsupp}(Y) = \texttt{colsupp}(S_Y)\}$,

The lifting procedure

$$A = XY^T, X \in \mathbb{R}^{n \times r}, Y \in \mathbb{R}^{m \times r}$$

$$\Downarrow$$

$$A = \sum_{i=1}^{r} x_i y_i^T = \sum_{i=1}^{r} \underbrace{M_i}_{\text{rank-one}}$$



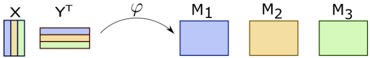
[L. Le Magoarou, **Matrices efficientes pour le traitement du signal et l'apprentissage automatique**, PhD thesis, 2016]

The lifting procedure

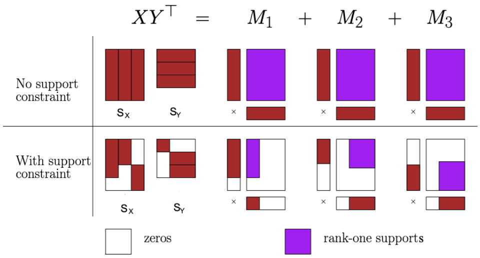
$$A = XY^T, X \in \mathbb{R}^{n\times r}, Y \in \mathbb{R}^{m\times r}$$

$$\Downarrow$$

$$A = \sum_{i=1}^{r} x_i y_i^T = \sum_{i=1}^{r} \underbrace{M_i}_{\text{rank-one}}$$



[L. Le Magoarou, Matrices efficientes pour le traitement du signal et l'apprentissage automatique, PhD thesis, 2016]

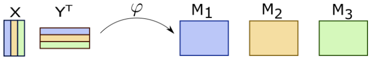


Identifiability of (X, Y) in $S = (S_X, S_Y) \leftrightarrow$ identifiability of $(M_i)_{i=1}^r$ in $\varphi(S) = (S_1, \ldots, S_r)$

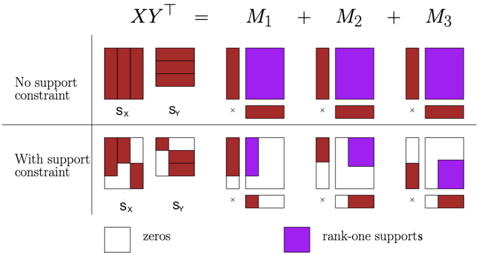
$$A = XY^T, X \in \mathbb{R}^{n \times r}, Y \in \mathbb{R}^{m \times r}$$

$$\Downarrow$$

$$A = \sum_{i=1}^{r} x_i y_i^T = \sum_{i=1}^{r} \underbrace{M_i}_{\text{rank-one}}$$



[L. Le Magoarou, **Matrices efficientes pour le traitement du signal et l'apprentissage automatique**, PhD thesis, 2016]



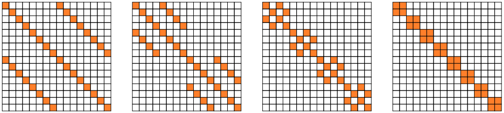
Identifiability of (X, Y) in $S = (S_X, S_Y) \leftrightarrow$ identifiability of $(M_i)_{i=1}^r$ in $\varphi(S) = (S_1, \ldots, S_r)$

Proposition

$\mathcal{U}(S) = IC_S \cap MC_S \leftrightarrow$ the rank-one supports $(\mathcal{S}_i)_{i=1}^r$ are pairwise disjoint

Examples

- Butterfly supports: $S_B^{(\ell)} := I_{2^{\ell-1}} \otimes \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \otimes I_{N/2^\ell},\ 1 \leq \ell \leq J, N = 2^J$



Butterfly supports: block diagonal + 2-sparse by row and by column.

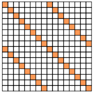
Application: common sparsity pattern for DCT, DST, DFT, Hadamard

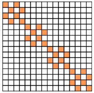
- Hierarchically off-diagonal low-rank (HODLR) matrices

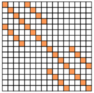
The butterfly factorization

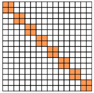
Product of $J \geq 2$ butterfly factors. Example:

$\mathsf{A} := \mathsf{X}^{(4)}\mathsf{X}^{(3)}\mathsf{X}^{(2)}\mathsf{X}^{(1)}$ such that:

$\mathrm{supp}(\mathbf{X}^{(4)}) \subseteq$ 

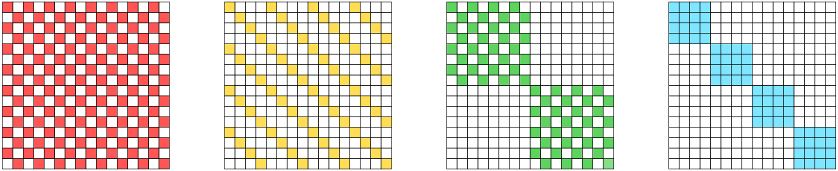
$\mathrm{supp}(\mathbf{X}^{(3)}) \subseteq$ 

$\mathrm{supp}(\mathbf{X}^{(2)}) \subseteq$ 

$\mathrm{supp}(\mathbf{X}^{(1)}) \subseteq$ 

Properties of the butterfly supports

- If $\mathrm{supp}(X^{(j)}) \subseteq S_B^{(j)}$, then $\mathrm{supp}(X^{(p)} \dots X^{(q)}) \subseteq S_B^{(p:q)} := S_B^{(p)} ... S_B^{(q)}$



(a) $S_B^{(1:3)}$ (b) $S_B^{(1:2)}$ (c) $S_B^{(2:3)}$ (d) $S_B^{(3:4)}$

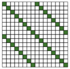
- The rank-one contributions of $(S_B^{(p:\ell)}, S_B^{(\ell+1:q)})$ have disjoint rank-one supports.

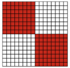
Two-layer fixed-support sparse matrix factorization

$$\min_{\mathbf{A},\mathbf{B}} \|\mathbf{Z} - \mathbf{A}\mathbf{B}\|_F^2, \text{ s.t. } \mathrm{supp}(\mathbf{A}) \subseteq \mathbf{S}_{\mathtt{bf}}^{(4)}, \ \mathrm{supp}(\mathbf{B}) \subseteq \mathbf{S}_{\mathtt{bf}}^{(3)}\mathbf{S}_{\mathtt{bf}}^{(2)}\mathbf{S}_{\mathtt{bf}}^{(1)}$$

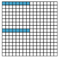
Fact: $\mathbf{A}\mathbf{B} = \sum_{i=1}^{N} \mathbf{A}_{\bullet,i}\mathbf{B}_{i,\bullet}$.

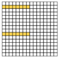
Constraint on the pair of factors

$\mathrm{supp}(\mathbf{A}) \subseteq$  $= \mathbf{S}_{\mathtt{bf}}^{(4)}$

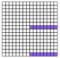
$\mathrm{supp}(\mathbf{B}) \subseteq$  $= \mathbf{S}_{\mathtt{bf}}^{(3)}\mathbf{S}_{\mathtt{bf}}^{(2)}\mathbf{S}_{\mathtt{bf}}^{(1)}$

Constraint on the rank-one matrices

$\mathrm{supp}(\mathbf{A}_{\bullet,1}\mathbf{B}_{1,\bullet}) \subseteq$  $= \mathcal{S}_1$

$\mathrm{supp}(\mathbf{A}_{\bullet,2}\mathbf{B}_{2,\bullet}) \subseteq$  $= \mathcal{S}_2$

$\vdots$

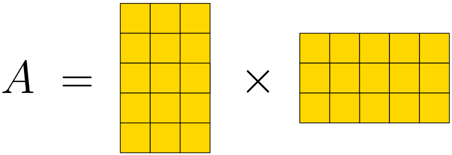
$\mathrm{supp}(\mathbf{A}_{\bullet,N}\mathbf{B}_{N,\bullet}) \subseteq$  $= \mathcal{S}_N$

Polynomially solvable instances (2 factors)

Example (Unconstrained matrix factorization)

If $S_X = [\![m]\!] \times [\![r]\!]$, $S_Y = [\![n]\!] \times [\![r]\!]$, i.e no constraints on the support of X and Y:

$$\underset{X \in \mathbb{R}^{m \times r}, Y \in \mathbb{R}^{n \times r}}{\text{Minimize}} L(X, Y) = \|A - XY^\top\|_F^2$$

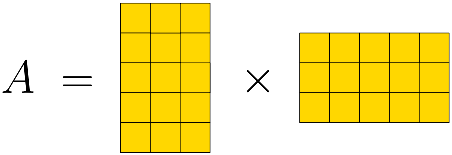


Polynomially solvable instances (2 factors)

Example (Unconstrained matrix factorization)

If $S_X = [\![m]\!] \times [\![r]\!]$, $S_Y = [\![n]\!] \times [\![r]\!]$, i.e no constraints on the support of X and Y:

$$\underset{X \in \mathbb{R}^{m \times r}, Y \in \mathbb{R}^{n \times r}}{\text{Minimize}} L(X, Y) = \|A - XY^\top\|_F^2$$



→ Solution: Use Singular Value Decomposition (SVD).

SVD as a greedy algorithm

1) Decompose the problem:

$$A - XY^\top = A - \sum_{i=1}^{r} x_i y_i^\top = A - \sum_{i=1}^{r} \underbrace{M_i}_{\text{rank-one}} \quad (M_i := x_i y_i^\top)$$

SVD as a greedy algorithm

1) Decompose the problem:

$$A - XY^\top = A - \sum_{i=1}^{r} x_i y_i^\top = A - \sum_{i=1}^{r} \underbrace{M_i}_{\text{rank-one}} \quad (M_i := x_i y_i^\top)$$

2) Finding the SVD:

$$\begin{aligned}
&\texttt{bestRankOneApprox}(A) && \to M_1 \\
&\texttt{bestRankOneApprox}(A - M_1) && \to M_2 \\
&\dots \\
&\texttt{bestRankOneApprox}(A - M_1 \ldots - M_{r-1}) && \to M_r
\end{aligned}$$

SVD as a greedy algorithm

1) Decompose the problem:

$$A - XY^\top = A - \sum_{i=1}^{r} x_i y_i^\top = A - \sum_{i=1}^{r} \underbrace{M_i}_{\text{rank-one}} \qquad (M_i := x_i y_i^\top)$$

2) Finding the SVD:

$$\begin{aligned}
&\texttt{bestRankOneApprox}(A) && \to M_1 \\
&\texttt{bestRankOneApprox}(A - M_1) && \to M_2 \\
&\dots && \\
&\texttt{bestRankOneApprox}(A - M_1 \dots - M_{r-1}) && \to M_r
\end{aligned}$$

$\to$ SVD is a greedy algorithm in disguise

Algorithm 1 Algorithm for unconstrained matrix factorization

1: **for** $i \in \{1, \dots, r\}$ **do**
2: $\quad M_i :=$ best rank-one approximation of $A - \sum_{k=1}^{i-1} M_k$.
3: **end for**

SVD as a greedy algorithm: the constrained case

- How to generalize the greedy algorithm?

SVD as a greedy algorithm: the constrained case

- How to generalize the greedy algorithm?
- Decompose $XY^\top$:

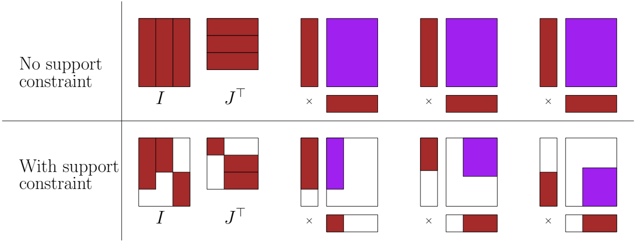
$$XY^\top = \sum_{i=1}^{r} x_i y_i^\top = \sum_{i=1}^{r} \underbrace{M_i}_{\text{rank-one}} \quad (M_i := x_i y_i^\top)$$

SVD as a greedy algorithm: the constrained case

- How to generalize the greedy algorithm?
- Decompose $XY^\top$:

$$XY^\top = \sum_{i=1}^{r} x_i y_i^\top = \sum_{i=1}^{r} \underbrace{M_i}_{\text{rank-one}} \qquad (M_i := x_i y_i^\top)$$

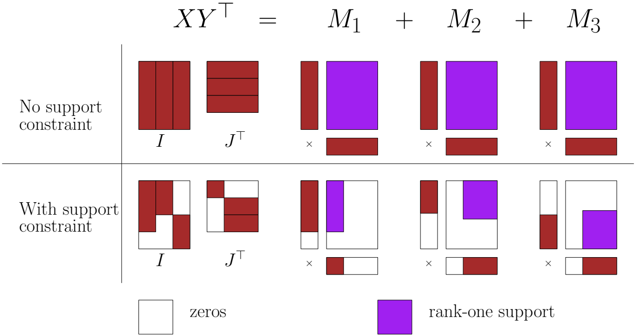
$$XY^\top = M_1 + M_2 + M_3$$



SVD as a greedy algorithm: the constrained case

- How to generalize the greedy algorithm?
- Decompose $XY^\top$:

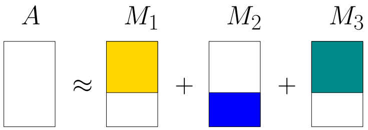
$$XY^\top = \sum_{i=1}^{r} x_i y_i^\top = \sum_{i=1}^{r} \underbrace{M_i}_{\text{rank-one}} \qquad (M_i := x_i y_i^\top)$$



- Finding optimal solution $(X, Y) \rightleftarrows$ Finding optimal entries in the rank-one supports.

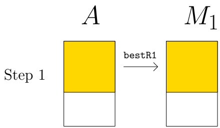
Algorithm 2 Algorithm for fixed-support matrix factorization

for $i \in \{1, \ldots, r\}$ **do**
  $S_i \leftarrow i$-th rank-one support
  $M_i :=$ best rank-one approximation of $(A - \sum_{k=1}^{i-1} M_k) \odot S_i$
end for



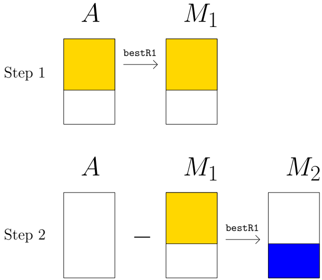
Algorithm 2 Algorithm for fixed-support matrix factorization

for $i \in \{1, \ldots, r\}$ **do**
    $S_i \leftarrow i$-th rank-one support
    $M_i :=$ best rank-one approximation of $(A - \sum_{k=1}^{i-1} M_k) \odot S_i$
end for



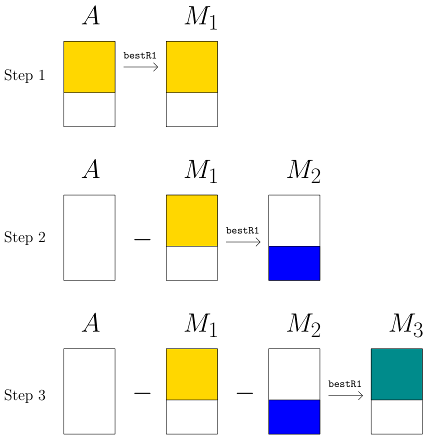
Algorithm 2 Algorithm for fixed-support matrix factorization

for $i \in \{1, \ldots, r\}$ **do**
  $S_i \leftarrow i$-th rank-one support
  $M_i :=$ best rank-one approximation of $(A - \sum_{k=1}^{i-1} M_k) \odot S_i$
end for



Algorithm 2 Algorithm for fixed-support matrix factorization

for $i \in \{1, \ldots, r\}$ **do**
    $S_i \leftarrow i$-th rank-one support
    $M_i :=$ best rank-one approximation of $(A - \sum_{k=1}^{i-1} M_k) \odot S_i$
end for



Polynomial solvability characterized by rank-one supports

- The solution will always *satisfy the constraints*
- It may NOT be *optimal*

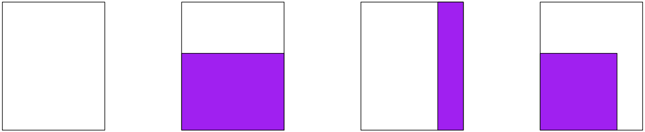
Theorem (Sufficient condition for tractability)

If the rank-one supports are pairwise disjoint or identical the greedy algorithm gives an optimal solution, even in the non-exact case

An even more general result exists

- A more general condition for tractability is introduced in our paper that allows for partial overlapping

$$A \approx M_1 + M_2 + M_3$$



[QT. Le, E. Riccietti, R. Gribonval, **Spurious Valleys, NP-hardness, and Tractability of Sparse Matrix Factorization With Fixed Support**, arxiv preprint, 2022.]

Multiple-factors case: a greedy hierarchical algorithm

Extension

Use our algorithm as a building block to approximate a matrix by a product of $J \geq 2$ sparse factors

$$\mathbf{X}^{(4)}\mathbf{X}^{(3)}\mathbf{X}^{(2)}\mathbf{X}^{(1)}$$

$$\mathbf{X}^{(4)} \qquad \mathbf{X}^{(3)}\mathbf{X}^{(2)}\mathbf{X}^{(1)}$$

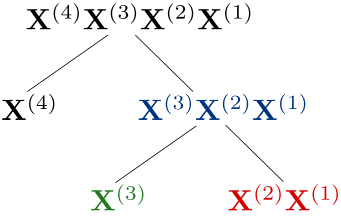
A greedy procedure

Use our algorithm to recover the partial factors: solve a sequence of two factors problems, if the supports are known

Multiple-factors case: a greedy hierarchical algorithm

Extension

Use our algorithm as a building block to approximate a matrix by a product of $J \geq 2$ sparse factors



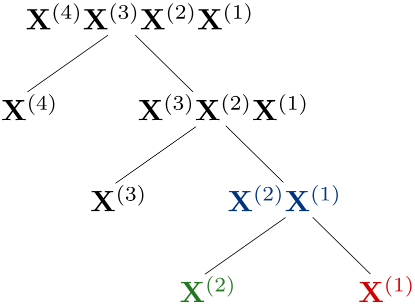
A greedy procedure

Use our algorithm to recover the partial factors: solve a sequence of two factors problems, if the supports are known

Multiple-factors case: a greedy hierarchical algorithm

Extension

Use our algorithm as a building block to approximate a matrix by a product of $J \geq 2$ sparse factors



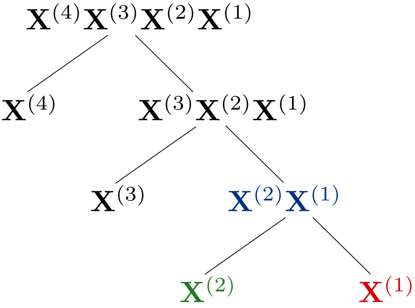
A greedy procedure

Use our algorithm to recover the partial factors: solve a sequence of two factors problems, if the supports are known

Multiple-factors case: a greedy hierarchical algorithm

Extension

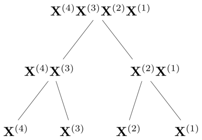
Use our algorithm as a building block to approximate a matrix by a product of $J \geq 2$ sparse factors



A greedy procedure

Use our algorithm to recover the partial factors: solve a sequence of two factors problems, if the supports are known

Works also with different kind of trees



An important application: the butterfly factorization

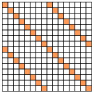
Theoretical guarantees?

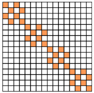
In general we cannot guarantee optimality of the solution.

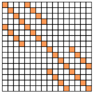
A special case: the butterfly factorization

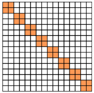
Approximate **any** matrix by a product of $J \geq 2$ butterfly factors

Let $\mathsf{A} := \mathsf{X}^{(4)}\mathsf{X}^{(3)}\mathsf{X}^{(2)}\mathsf{X}^{(1)}$ such that:

$\mathrm{supp}(\mathbf{X}^{(4)}) \subseteq$ 

$\mathrm{supp}(\mathbf{X}^{(3)}) \subseteq$ 

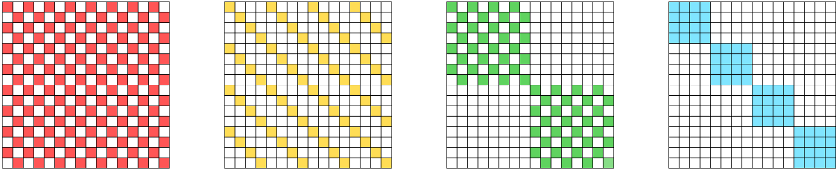
$\mathrm{supp}(\mathbf{X}^{(2)}) \subseteq$ 

$\mathrm{supp}(\mathbf{X}^{(1)}) \subseteq$ 

Properties of the butterfly supports

Why the butterfly structure? Tractability conditions hold at each step

- If $\text{supp}(X^{(j)}) \subseteq S_B^{(j)}$, then $\text{supp}(X^{(p)} \ldots X^{(q)}) \subseteq S_B^{(p:q)} := S_B^{(p)} \ldots S_B^{(q)}$



(a) $S_B^{(1:3)}$ (b) $S_B^{(1:2)}$ (c) $S_B^{(2:3)}$ (d) $S_B^{(3:4)}$

- The rank-one contributions of $(S_B^{(p:\ell)}, S_B^{(\ell+1:q)})$ have disjoint rank-one supports.

First step of the hierarchical factorization algorithm

$$\min \|Z - X^{(4)}X^{(3)}X^{(2)}X^{(1)}\|_F^2$$

$\mathbf{X}^{(4)}\mathbf{X}^{(3)}\mathbf{X}^{(2)}\mathbf{X}^{(1)}$

$\mathbf{X}^{(4)}$ $\quad$ $\mathbf{X}^{(3)}\mathbf{X}^{(2)}\mathbf{X}^{(1)}$

Two-layer fixed-support sparse matrix factorization

$$\min_{\mathbf{A},\mathbf{B}} \|\mathbf{Z} - \mathbf{A}\mathbf{B}\|_F^2, \ \text{s.t.}\ \text{supp}(\mathbf{A}) \subseteq \mathbf{S}_{\text{bf}}^{(4)},\ \text{supp}(\mathbf{B}) \subseteq \mathbf{S}_{\text{bf}}^{(3)}\mathbf{S}_{\text{bf}}^{(2)}\mathbf{S}_{\text{bf}}^{(1)}$$

Fact: $\mathbf{AB} = \sum_{i=1}^{N} \mathbf{A}_{\bullet,i}\mathbf{B}_{i,\bullet}$.

Constraint on the pair of factors

$\text{supp}(\mathbf{A}) \subseteq$  $= \mathbf{S}_{\text{bf}}^{(4)}$

$\text{supp}(\mathbf{B}) \subseteq$  $= \mathbf{S}_{\text{bf}}^{(3)}\mathbf{S}_{\text{bf}}^{(2)}\mathbf{S}_{\text{bf}}^{(1)}$

Constraint on the rank-one matrices

$\text{supp}(\mathbf{A}_{\bullet,1}\mathbf{B}_{1,\bullet}) \subseteq$  $= \mathcal{S}_1$

$\text{supp}(\mathbf{A}_{\bullet,2}\mathbf{B}_{2,\bullet}) \subseteq$  $= \mathcal{S}_2$

$\vdots$

$\text{supp}(\mathbf{A}_{\bullet,N}\mathbf{B}_{N,\bullet}) \subseteq$  $= \mathcal{S}_N$

Proposition

The rank-one matrices have pairwise disjoint supports. Consequently, (FSMF) is polynomially solvable and admits an essentially unique solution.

Butterfly factorization: theoretical guarantees

Exact setting

- The exact factorization $A = X^{(J)} \dots X^{(1)}$ into J butterfly factors is essentially unique
- These factors can be recovered by our algorithm.

Noisy setting

- Our algorithm can approximate any matrix by a matrix having the butterfly structure
- Global optimality of the multi-layer factorization is *not guaranteed*

Practical advantages

Bounded complexity

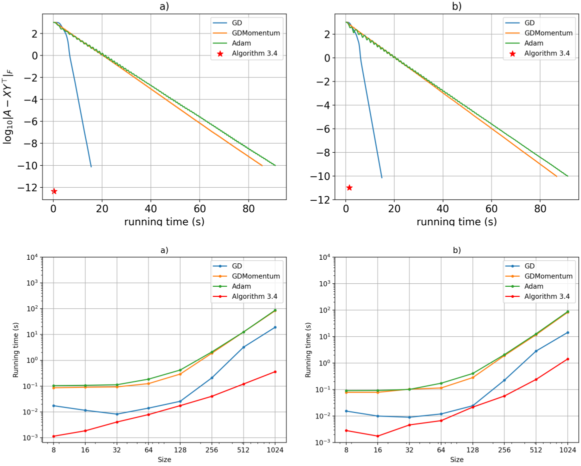
- A controlled number of truncated SVDs
- Complexity of the factorization algorihtm: $O(N^2)$, $N = 2^J \rightarrow$ the cost of few dense matrix-vector products
- The factorization allows fast matrix-vector products in $O(N\log(N))$

A direct algorithm

- No hyper parameters tuning (learning rate or stopping criterion)
- No sensitivity to initialization

Numerical results: 2 factors

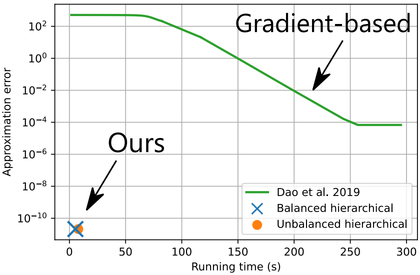
A the Hadamard matrix of size $2^J \times 2^J$, $J = 10$, two different supports



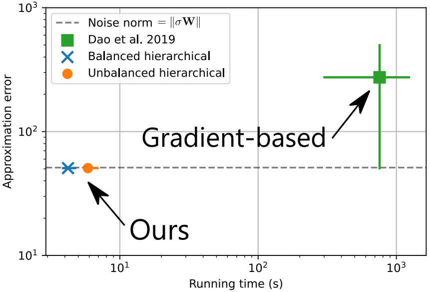
Numerical results: J factors

Approximation of the DFT matrix by a product of $J = 9$ butterfly factors.

Faster and more accurate in the noiseless setting

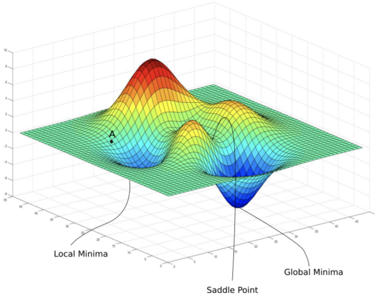


Also more robust in the noisy setting



Study of the landscape of the loss function

$$L(X, Y) = \|A - XY^T\|_F^2$$



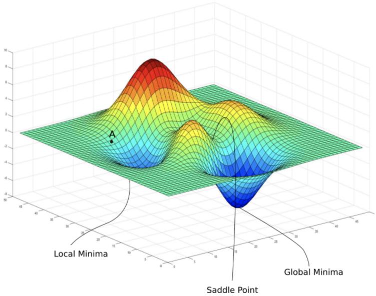
Has been studied for:

- linear and shallows neural networks
- matrix sensing, phase retrieval, matrix completion ...

[Q. Li, Z. Zhu, G. Tang, **The non-convex geometry of low-rank matrix optimization**, Information and Inference, 2018]
[Z. Zhu & all. **The global optimization geometry of shallow linear neural networks**, JMIV, 2019]
[L. Venturi, A. S. Bandeira, J. Bruna, **Spurious valleys in one-hidden-layer neural network optimization landscapes**, JMLR, 2019]

Study of the landscape of the loss function

$$L(X, Y) = \|A - XY^T\|_F^2$$



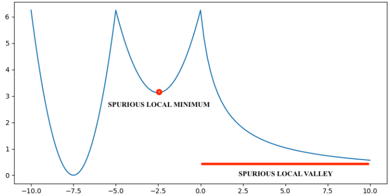
Has been studied for:

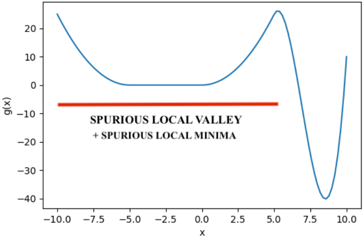
- linear and shallows neural networks
- matrix sensing, phase retrieval, matrix completion ...

[Q. Li, Z. Zhu, G. Tang, **The non-convex geometry of low-rank matrix optimization**, Information and Inference, 2018]
[Z. Zhu & all. **The global optimization geometry of shallow linear neural networks**, JMIV, 2019]
[L. Venturi, A. S. Bandeira, J. Bruna, **Spurious valleys in one-hidden-layer neural network optimization landscapes**, JMLR, 2019]

Never with support constraints!

Undesirable spurious objects





Example of spurious local minimum and spurious local valley. Two undesirable objects: may make the convergence of iterative methods difficult

Definition (Spurious local valley - Informal)

$S \in \mathbb{R}^d$ is a spurious local valley if for all $x \in S$, there does not exist any *continuous path* connecting x and a global minimum x^* *without increasing* the loss function f.

Landscape of full support matrix factorization

- What makes the low rank matrix approximation special?

$$\underset{X\in\mathbb{R}^{m\times r}, Y\in\mathbb{R}^{n\times r}}{\text{Minimize}} L(X,Y) = \|A - XY^\top\|^2$$

- The landscape of $L(X,Y)$ is *benign*:
  - No spurious local minima.[1]
  - No spurious local valleys [2]

[1] [Z. Zhu & all. **The global optimization geometry of shallow linear neural networks**, JMIV, 2019]

[2] [L. Venturi, A. S. Bandeira, J. Bruna, **Spurious valleys in one-hidden-layer neural network optimization landscapes**, JMLR, 2019]

Landscape of $L(X, Y)$ under sparsity constraints

Fixed support matrix factorization

$$\underset{X \in \mathbb{R}^{m \times r}, Y \in \mathbb{R}^{n \times r}}{\text{Minimize}} \quad L(X, Y) = \|A - XY^\top\|^2$$

$$\text{Subject to:} \quad \text{supp}(X) \subseteq S_X$$

$$\text{supp}(Y) \subseteq S_Y$$

Theorem (Spurious local minima and valley)

If (S_X, S_Y) satisfy the condition of polynomial solvability, for all A, the landscape of $L(X, Y)$ does not contain any spurious local minimum and spurious local valley.

A tempting conjecture?

Conjecture (?)

FSMF is polynomially solvable iff its landscape has no spurious objects.

A FALSE conjecture?

Conjecture (?)

FSMF is polynomially solvable iff its landscape has no spurious objects.

$\rightarrow$ This conjecture is not *true*. There is a counter-example.

A FALSE conjecture?

Conjecture (?)

FSMF is polynomially solvable iff its landscape has no spurious objects.

$\rightarrow$ This conjecture is not *true*. There is a counter-example.

Example

Take $n = m = r$ and consider the LU decomposition of the matrix A:

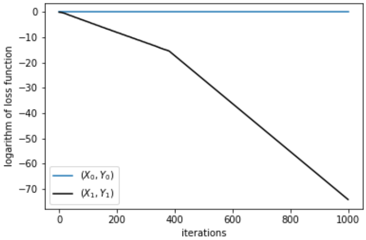
$$A = \begin{pmatrix} A' & 0 \\ 0 & 0 \end{pmatrix}, \quad A' = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \in \mathbb{R}^{2\times 2}$$

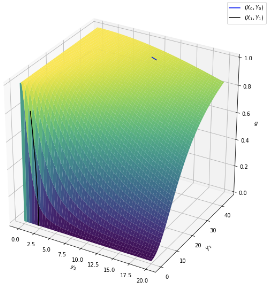
For this A, $L(X, Y)$ has a spurious local valley but it exists a polynomial algorithm to solve FSMF

Is the behaviour of GD good in a benign landscape?

Choice of the initial guess

- *Speed of convergence* of GD is deeply affected by the choice of the initial guess, even in absence of spurious objects in the landscape.
- In case of spurious valleys, GD *is not ensured to stay out* of them, even with a good initialisation





Conclusions

Take home message

For Fixed support matrix factorization (FSMF), we have:

1) It is NP-hard to solve
2) Easy instances with effective direct algorithm exists, competitive with gradient descent
3) Those easy instances have benign landscape
4) Any matrix having the butterfly structure admits an essentially unique factorization. The factors can be recovered by a hierarchical factorization method

Quantization of a sparse factorization

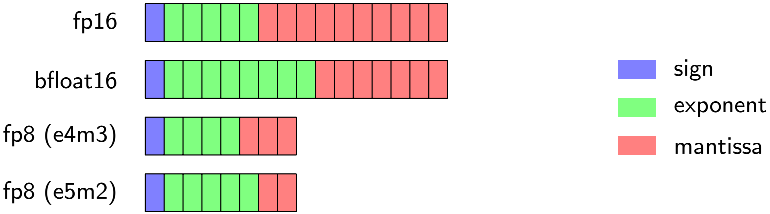
Aim: approximate dense A by product of quantized sparse factors:

$$A \sim \hat{S}_1 \dots \hat{S}_L$$

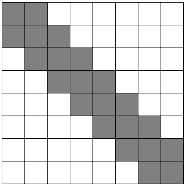
Motivation

Growing size of models and datasets $\rightarrow$ approximate computing

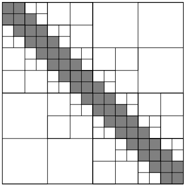
- Quantization to low precision floating-point arithmetic



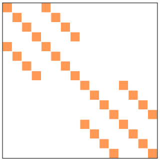
- Low-rank, structured, data sparse matrices



BLR matrix

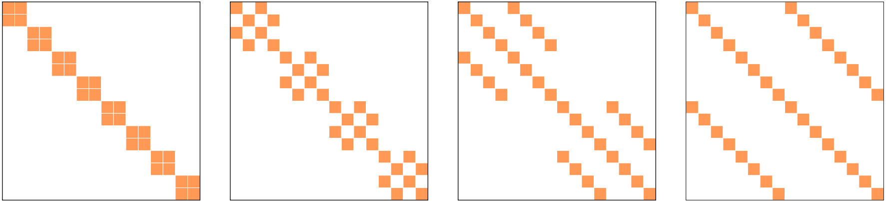


$\mathcal{H}$-matrix



Butterfly matrix

Let us focus on butterfly matrices



- Butterfly matrices are extremely sparse yet highly expressive, they appear in many fast linear transforms
- Butterfly factorization: decompose dense $n \times n$ matrix as $B_1 \ldots B_L$, with $L = \log_2 n \Rightarrow O(n \log n)$ complexity

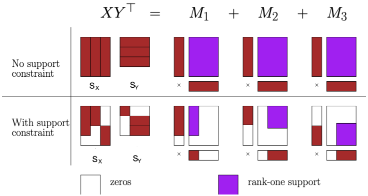
Optimal two-factor quantization

- Remember our key property: for any partial product XY^T of consecutive factors

$$B_1 \dots B_j \underbrace{B_{j+1} \dots B_k}_{X} \underbrace{B_{k+1} \dots B_\ell}_{Y^T} B_{\ell+1} \dots B_L$$

$$XY^T = \sum_{i=1}^{n} x_i y_i^T$$

where the rank-one matrices $x_i y_i^T$ have disjoint support.



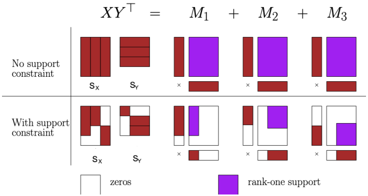
Optimal two-factor quantization

- Remember our key property: for any partial product XY^T of consecutive factors

$$B_1 \dots B_j \underbrace{B_{j+1} \dots B_k}_{X} \underbrace{B_{k+1} \dots B_\ell}_{Y^T} B_{\ell+1} \dots B_L$$

$$XY^T = \sum_{i=1}^{n} x_i y_i^T$$

where the rank-one matrices $x_i y_i^T$ have disjoint support.



- We can optimally quantize two factors X and Y by quantizing each $x_i y_i^T$ optimally and independently

Quantization of rank-one matrices

Goal: quantize the rank-one matrix

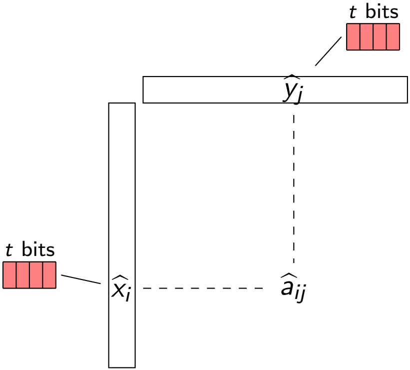
$$xy^T \to \widehat{x}\widehat{y}^T \qquad (x \in \mathbb{R}^m,\ y \in \mathbb{R}^n)$$

where the coefficients of $\widehat{x}$, $\widehat{y}$ have t bits of mantissa

- The standard approach uses round-to-nearest (RTN) and leads to an error of order $u = 2^{-t}$: if $\widehat{x} = \text{round}(x)$, $\widehat{y} = \text{round}(y)$ then

$$\begin{aligned} \|\widehat{x} - x\| &\le u\|x\| \\ \|\widehat{y} - y\| &\le u\|y\| \\ \Rightarrow \|\widehat{x}\widehat{y}^T - xy^T\| &\le (2u + u^2)\|x\|\|y\| \end{aligned}$$

- We will show this is far from optimal!

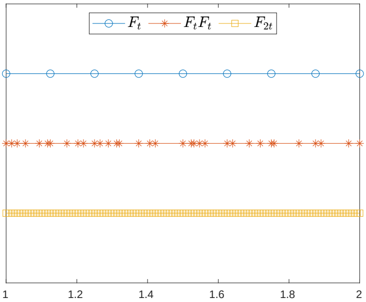


- What we really care about is the accuracy of $\widehat{a}_{ij} = \widehat{x}_i \widehat{y}_j$, not of the two separately
- What about $a = xy$? (Which $\widehat{x}$,$\widehat{y}$ yields the best approximation $\widehat{x}\widehat{y}$?)

The simplest case: $m = n = 1$

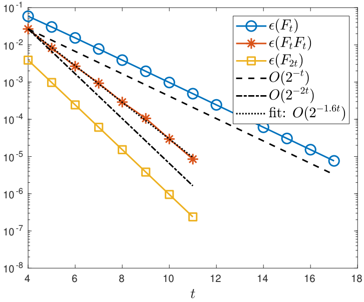
- Let $\mathbb{F}_t$ be the set of t-bit floating-point numbers. We are interested in the set

$$\mathbb{F}_t\mathbb{F}_t = \{a = xy,\ x \in \mathbb{F}_t,\ y \in \mathbb{F}_t\}$$



- No closed form expression of its elements, but we can simply enumerate all of them for small t

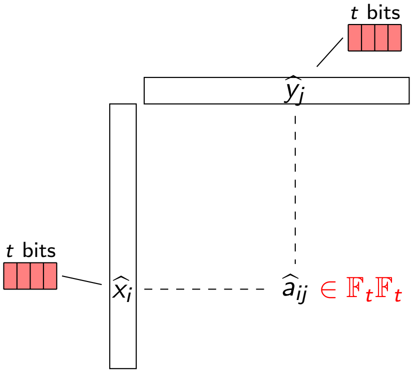
The simplest case: $m = n = 1$



- $\epsilon(S) = \sup_{z \neq 0} \frac{d(z,S)}{|z|}$: the worst-case relative error of quantizing an element $z \in \mathbb{R}$ on S
- $\epsilon(\mathbb{F}_t) = \frac{2^{-t}}{1+2^{-t}}$

$\Rightarrow$ $\epsilon(\mathbb{F}_t \mathbb{F}_t)$ error of order $2^{-1.6t}$

A constrained combinatorial problem



We don't just have one scalar, but a rank-one matrix $\Rightarrow$ two issues:

- We have constraints: $\widehat{x}_i$ must be the same in $\widehat{a}_{ij} = \widehat{x}_i\widehat{y}_j$ and $\widehat{a}_{ik} = \widehat{x}_i\widehat{y}_k$
- How can we find the optimal quantization? Combinatorial problem!

$$\min_{\widehat{x}\in\mathbb{F}_t^m, \widehat{y}\in\mathbb{F}_t^n} \|xy^T - \widehat{x}\widehat{y}^T\|$$

Intuition

In exact arithmetic:

$$xy^T = (\lambda x)(\frac{1}{\lambda}y)^T$$

In floating point arithmetic

$$\text{round}(xy^T) \neq \text{round}(\lambda x)\,\text{round}(\frac{1}{\lambda}y)^T$$

- Can we find the optimal scaling λ^*?
- Can we reduce the problem to a scalar problem?

Characterization of the optimum

Theorem

$$\min_{\widehat{x}\in\mathbb{F}_t^m, \widehat{y}\in\mathbb{F}_t^n} \|xy^T - \widehat{x}\widehat{y}^T\| = \min_{\lambda\in\mathbb{R}} \|xy^T - \mathrm{round}(\lambda x)\,\mathrm{round}(\mu(\lambda)y)^T\|$$

The optimal quantization $\widehat{x}\widehat{y}^T$ is given by

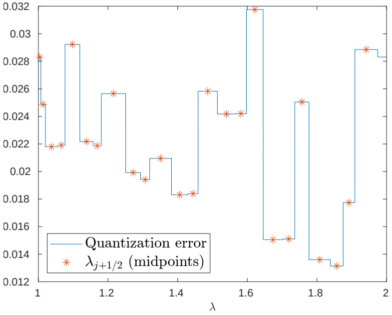
$$\widehat{x} = \mathrm{round}(\lambda x)$$
$$\widehat{y} = \mathrm{round}(\mu(\lambda)y^T)$$

where $\lambda \in \mathbb{R}$ and $\mu(\lambda) = \frac{x^T\widehat{x}}{\|\widehat{x}\|^2}$.

- It suffices to find the optimal λ to find the optimal $\widehat{x}\widehat{y}^T$!

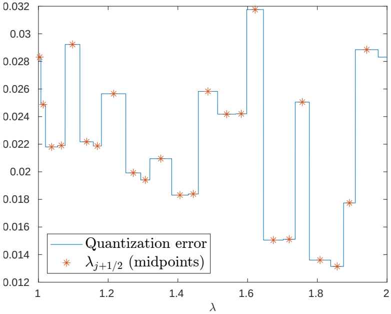
Finding λ

- How do we find the optimal $\lambda \in \mathbb{R}$?
- The optimum is stable under sign flip and multiplication by powers of two $\rightarrow$ restrict the search to $\lambda \in [1, 2]$
- Only a finite number of values of λ change the value of round(λx). Denoting these "breakpoints" as λ_j, we can enumerate the midpoints $\lambda_{j+1/2} = (\lambda_j + \lambda_{j+1})/2$



Finding λ

- How do we find the optimal $\lambda \in \mathbb{R}$?
- The optimum is stable under sign flip and multiplication by powers of two $\rightarrow$ restrict the search to $\lambda \in [1, 2]$
- Only a finite number of values of λ change the value of round(λx). Denoting these "breakpoints" as λ_j, we can enumerate the midpoints $\lambda_{j+1/2} = (\lambda_j + \lambda_{j+1})/2$



Algorithm:

- Build the set of midpoints
- For each midpoint $\lambda_{j+1/2}$:
  - Build $\widehat{x} = \text{round}(\lambda_{j+1/2} x)$
  - Compute $\mu(\widehat{x}) = x^T \widehat{x} / \|\widehat{x}\|^2$
  - Build $\widehat{y} = \text{round}(\mu y)$
  - Test the accuracy of $\widehat{x}\widehat{y}^T$

Complexity of the algorithm

$O(mn2^t)$ complexity $\Rightarrow$ tractable for large matrices and low precisions

Alternative: Approximation of the optimum via (1D) derivative free optimization

$$\min_{\lambda \in \mathbb{R}} \|xy^T - \mathrm{round}(\lambda x)\,\mathrm{round}(\mu(\lambda)y)^T\|$$

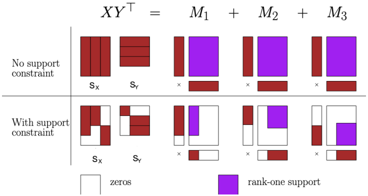
Back to butterfly quantization: optimal two-factor quantization

- We use the key property: for any partial product XY^T of consecutive factors

$$B_1 \dots B_j \underbrace{B_{j+1} \dots B_k}_{X} \underbrace{B_{k+1} \dots B_\ell}_{Y^T} B_{\ell+1} \dots B_L$$

$$XY^T = \sum_{i=1}^{n} x_i y_i^T$$

where the rank-one matrices $x_i y_i^T$ have disjoint support.



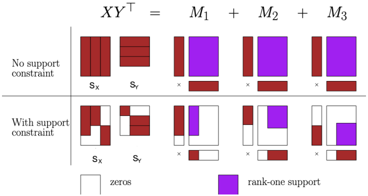
Back to butterfly quantization: optimal two-factor quantization

- We use the key property: for any partial product XY^T of consecutive factors

$$B_1 \dots B_j \underbrace{B_{j+1} \dots B_k}_{X} \underbrace{B_{k+1} \dots B_\ell}_{Y^T} B_{\ell+1} \dots B_L$$

$$XY^T = \sum_{i=1}^{n} x_i y_i^T$$

where the rank-one matrices $x_i y_i^T$ have disjoint support.



- We can optimally quantize two factors X and Y by quantizing each $x_i y_i^T$ optimally and independently: $\widehat{x_i} = \text{round}(\lambda_i x_i)$, $\widehat{y_i} = \text{round}(\mu_i y_i)$ yields

$$\widehat{X} = \text{round}(X\Lambda), \quad \Lambda = \text{diag}(\lambda_i)$$

$$\widehat{Y} = \text{round}(YM), \quad M = \text{diag}(\mu_i)$$

Heuristics for the L-factor butterfly quantization

When $L > 2$, need heuristics to decide how to order/group the factors

- Pairwise heuristic:

$$B_1 \, B_2 \, B_3 \, B_4 \ldots B_L$$

Heuristics for the L-factor butterfly quantization

When $L > 2$, need heuristics to decide how to order/group the factors

- Pairwise heuristic:

$$\underbrace{B_1 \, B_2}_{XY^T} \, B_3 \, B_4 \ldots B_L$$

Heuristics for the L-factor butterfly quantization

When $L > 2$, need heuristics to decide how to order/group the factors

- Pairwise heuristic:

$$\underbrace{\widehat{B}_1 \, \widehat{B}_2}_{XY^T} \, B_3 \, B_4 \ldots B_L$$

Heuristics for the L-factor butterfly quantization

When $L > 2$, need heuristics to decide how to order/group the factors

- Pairwise heuristic:

$$\underbrace{\widehat{B}_1 \, \widehat{B}_2}_{XY^T} \underbrace{B_3 \, B_4}_{XY^T} \dots B_L$$

Heuristics for the L-factor butterfly quantization

When $L > 2$, need heuristics to decide how to order/group the factors

- Pairwise heuristic:

$$\underbrace{\widehat{B}_1 \, \widehat{B}_2}_{XY^T} \underbrace{\widehat{B}_3 \, \widehat{B}_4}_{XY^T} \dots B_L$$

Heuristics for the L-factor butterfly quantization

When $L > 2$, need heuristics to decide how to order/group the factors

- Pairwise heuristic:

$$\underbrace{\widehat{B}_1 \, \widehat{B}_2}_{XY^T} \underbrace{\widehat{B}_3 \, \widehat{B}_4}_{XY^T} \ldots \underbrace{\widehat{B}_L}_{RTN}$$

Heuristics for the L-factor butterfly quantization

When $L > 2$, need heuristics to decide how to order/group the factors

- Pairwise heuristic:

$$\underbrace{\widehat{B}_1 \, \widehat{B}_2}_{XY^T} \underbrace{\widehat{B}_3 \, \widehat{B}_4}_{XY^T} \dots \underbrace{\widehat{B}_L}_{RTN}$$

- Left-to-right heuristic:

$$B_1 \, B_2 \, B_3 \dots B_{L-1} B_L$$

Heuristics for the L-factor butterfly quantization

When $L > 2$, need heuristics to decide how to order/group the factors

- Pairwise heuristic:

$$\underbrace{\widehat{B}_1\,\widehat{B}_2}_{XY^T}\,\underbrace{\widehat{B}_3\,\widehat{B}_4}_{XY^T}\ldots\underbrace{\widehat{B}_L}_{RTN}$$

- Left-to-right heuristic:

$$\underbrace{B_1}_{X}\,\underbrace{B_2\,B_3\ldots B_{L-1}B_L}_{Y^T}$$

Heuristics for the L-factor butterfly quantization

When $L > 2$, need heuristics to decide how to order/group the factors

- Pairwise heuristic:

$$\underbrace{\widehat{B}_1 \, \widehat{B}_2}_{XY^T} \underbrace{\widehat{B}_3 \, \widehat{B}_4}_{XY^T} \ldots \underbrace{\widehat{B}_L}_{RTN}$$

- Left-to-right heuristic:

$$\underbrace{\widehat{B}_1}_{X} \underbrace{{\color{red}M_2} B_2 \, B_3 \ldots B_{L-1} B_L}_{Y^T}$$

Heuristics for the L-factor butterfly quantization

When $L > 2$, need heuristics to decide how to order/group the factors

- Pairwise heuristic:

$$\underbrace{\widehat{B}_1 \, \widehat{B}_2}_{XY^T} \underbrace{\widehat{B}_3 \, \widehat{B}_4}_{XY^T} \ldots \underbrace{\widehat{B}_L}_{RTN}$$

- Left-to-right heuristic:

$$\underbrace{\widehat{B}_1}_{X} \; \underbrace{\underbrace{{\color{red}M_2} B_2}_{X} \underbrace{B_3 \ldots B_{L-1} B_L}_{Y^T}}_{Y^T}$$

Heuristics for the L-factor butterfly quantization

When $L > 2$, need heuristics to decide how to order/group the factors

- Pairwise heuristic:

$$\underbrace{\widehat{B}_1 \widehat{B}_2}_{XY^T} \underbrace{\widehat{B}_3 \widehat{B}_4}_{XY^T} \dots \underbrace{\widehat{B}_L}_{RTN}$$

- Left-to-right heuristic:

$$\underbrace{\widehat{B}_1}_{X} \underbrace{\underbrace{\widehat{B}_2}_{X} \underbrace{\color{red}{M_3} B_3 \dots B_{L-1} B_L}_{Y^T}}_{Y^T}$$

Heuristics for the L-factor butterfly quantization

When $L > 2$, need heuristics to decide how to order/group the factors

- Pairwise heuristic:

$$\underbrace{\widehat{B}_1 \widehat{B}_2}_{XY^T} \underbrace{\widehat{B}_3 \widehat{B}_4}_{XY^T} \dots \underbrace{\widehat{B}_L}_{RTN}$$

- Left-to-right heuristic:

$$\underbrace{\widehat{B}_1}_{X} \underbrace{\underbrace{\widehat{B}_2}_{X} \underbrace{\widehat{B}_3 \dots \underbrace{\color{red}{M_{L-1}}B_{L-1}}_{X} \underbrace{B_L}_{Y^T}}_{Y^T}}_{Y^T}$$

Heuristics for the L-factor butterfly quantization

When $L > 2$, need heuristics to decide how to order/group the factors

- Pairwise heuristic:

$$\underbrace{\widehat{B}_1 \, \widehat{B}_2}_{XY^T} \underbrace{\widehat{B}_3 \, \widehat{B}_4}_{XY^T} \dots \underbrace{\widehat{B}_L}_{RTN}$$

- Left-to-right heuristic:

$$\underbrace{\widehat{B}_1}_{X} \; \underbrace{\underbrace{\widehat{B}_2}_{X} \; \underbrace{\widehat{B}_3 \dots \underbrace{\widehat{B}_{L-1}}_{X} \, \underbrace{\widehat{B}_L}_{Y^T}}_{Y^T}}_{Y^T}$$

Heuristics for the L-factor butterfly quantization

When $L > 2$, need heuristics to decide how to order/group the factors

- Pairwise heuristic:

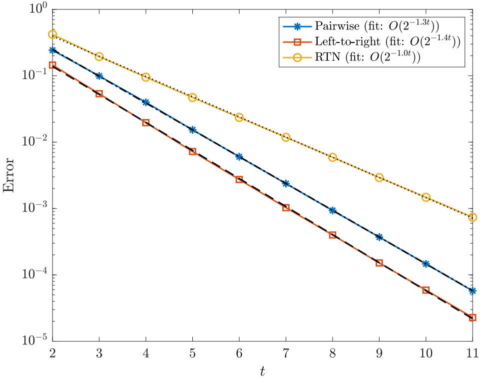
$$\underbrace{\widehat{B}_1 \widehat{B}_2}_{XY^T} \underbrace{\widehat{B}_3 \widehat{B}_4}_{XY^T} \dots \underbrace{\widehat{B}_L}_{RTN}$$

- Left-to-right heuristic:

$$\underbrace{\widehat{B}_1}_{X} \underbrace{\underbrace{\widehat{B}_2}_{X} \underbrace{\widehat{B}_3 \dots \underbrace{\widehat{B}_{L-1}}_{X} \underbrace{\widehat{B}_L}_{Y^T}}_{Y^T}}_{Y^T}$$

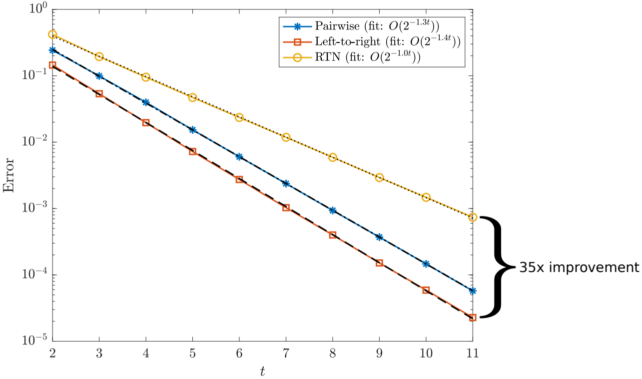
- L2R more expensive because it densifies the factors

Experimental results



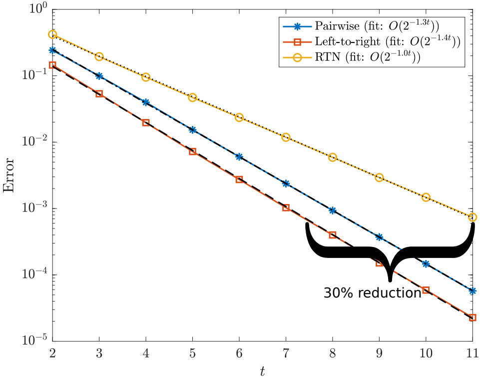
- Randomly generated butterfly factors

Experimental results



- Randomly generated butterfly factors
- Significant accuracy improvement...

Experimental results



- Randomly generated butterfly factors
- Significant accuracy improvement. . .
- . . . or, equivalently, can reduce storage by about 30% with no loss of accuracy

Conclusion

Key results:

- Characterized optimal quantization of xy^T as $\text{round}(\lambda x)\,\text{round}(\mu y)^T$
- Proposed algorithm to find the optimal λ in $O(mn2^t)$ complexity
- Proposed two heuristics to apply method to butterfly factorization and obtained storage reductions of 30% with no loss of accuracy

Butterfly matrices are only one possible application, many other perspectives: rank-r matrices, tensors, DNNs, ...

In practice: the FAμST library

FAμST library: an implementation of the hierarchical algorithm, fast GPU matrix vector multiplication of butterfly matrices, quantization algorithm in C++ via Python and Matlab wrappers FAμST 3.25 toolbox at `https://faust.inria.fr/`.

To know more:

- R. Gribonval, T. Mary, E. Riccietti (2024), Optimal quantization of rank-one matrices in floating point arithmetic - with applications to butterfly factorizations, in revision for SISC.
- Q.-T. Le, E. Riccietti, and R. Gribonval (2023), Spurious Valleys, Spurious Minima and NP-hardness of Sparse Matrix Factorization With Fixed Support, SIMAX.
- L. Zheng, E. Riccietti, and R. Gribonval (2023), Efficient Identification of Butterfly Sparse Matrix Factorizations, SIMODS.
- Q.-T. Le, L. Zheng, E. Riccietti, and R. Gribonval (2022), Fast learning of fast transforms, with guarantees, ICASSP 2022