Sparse Matrix Factorization from an Optimization Point of View

Quoc Tung Le Léon Zheng Elisa Riccietti Rémi Gribonval

Université de Lyon, Ecole Normale Supérieure de Lyon

INRIA, CNRS, Laboratoire de l'Informatique du Parallélisme - LIP

LIP PhD, Seminar
25nd May 2023

Joint work with

Léon Zheng

Elisa Riccietti

Rémi Gribonval

Sparse matrix factorization

Objective: Given 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

Objective: Given 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

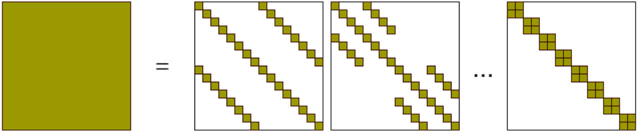
- *Fast* matrix vector products:

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

- Reduce time + memory complexity

Applications

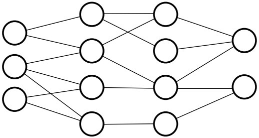
- **Fast Fourier Transform**, **Fast Hadamard Transform**, etc.



- **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**:
  - Interpretability.
  - Energy efficiency.



[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 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 and $\mathcal{E}_j$ some sets of sparse matrices, solve:

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

- Example of *sparsity patterns* $\mathcal{E}_j$: set of matrices with at most k nonzero entries per row, per column or 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 and $\mathcal{E}_j$ some sets of sparse matrices, solve:

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

- Example of *sparsity patterns* $\mathcal{E}_j$: set of matrices with at most k nonzero entries per row, per column or 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 challenging problem, how to deal with it?

Our approach: from two to multiple factors

- **Two** factors matrix factorization: the simplest nontrivial case.

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

- **Multiple** factors matrix factorization: butterfly factorization.

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, S_Y - the supports of two factors X, Y

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, S_Y - the supports of two factors X, Y

2) Optimize coefficients inside support

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

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

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

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, S_Y - the supports of two factors X, Y

2) Optimize coefficients inside support

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

Subject to:

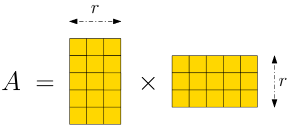
$$\text{supp}(X) \subseteq S_X$$
$$\text{supp}(Y) \subseteq S_Y$$

$\rightarrow$ Second problem: Fixed support matrix factorization (FSMF).

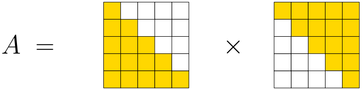
Examples of FSMF

FSMF covers:

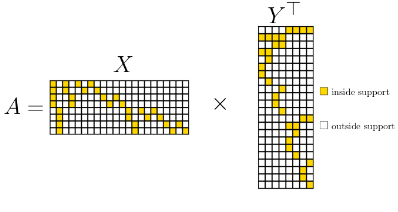
- Low rank matrix decomposition



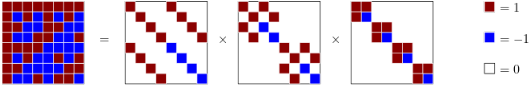
- LU decomposition



- Hierarchical $\mathcal{H}$ matrices

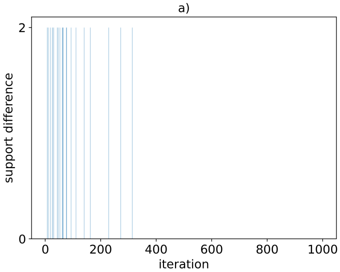


- Butterfly factorization (multiple factors)



Understand the *asymptotic behaviour* of other existing heuristics:

- Supports asymptotically don't change
- Spurious local valley in the landscape of $L(X, Y)$: certain heuristic can lead to iterates diverging to infinity



Measure of the difference between two consecutive supports

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

Main contributions on FSMF

1. Is the problem polynomially tractable?
   → We prove its *NP-hardness*.

2. Does the problem have a solution?
   → We show an instance where optimal solutions do *not* exist.

3. Are there easy instances?
   → We individuate a family of *polynomially solvable* instances and proposed an *efficient algorithm*.

4. How well does gradient descent tackle the problem of FSMF?
   → We study the properties of the *landscape* of the function $L(X, Y) = \|A - XY^\top\|^2$ *under the support constraints*.

NP-hardness

Theorem (1)

Sparse matrix factorization is NP-hard even with fixed support

Proof.

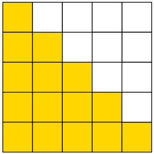
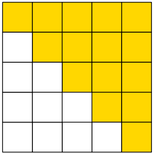
Rank-one matrix completion with noise is reducible to FSMF. □

- In line with recent results on matrix factorization:
  - non-negative matrix factorization (NMF)
  - weighted low rank
  - matrix completion

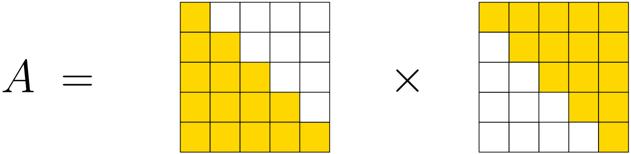
[N. Gillis, F. Glineur, Low-rank matrix approximation with weights or missing data is NP-hard. SIAM JMAA, 2010]

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

LU decomposition and non-closedness

$A =$  $\times$ 

LU decomposition and non-closedness

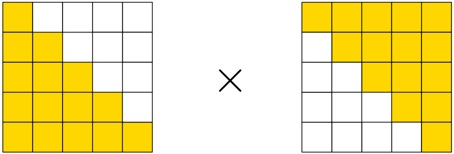


- Fact: Any square matrix is the limit of a sequence of matrices having an LU decomposition.
- But there exist square matrices that **do not have** an exact LU decomposition.

[H. Golub and C. F. Van Loan, **Matrix Computations**, **The Johns Hopkins University Press**]

LU decomposition and non-closedness

$A =$

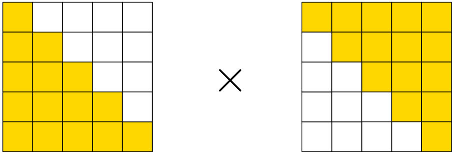


- Fact: Any square matrix is the limit of a sequence of matrices having an LU decomposition.
- But there exist square matrices that **do not have** an exact LU decomposition.

  [H. Golub and C. F. Van Loan, **Matrix Computations**, **The Johns Hopkins University Press**]

$\longrightarrow$ 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.

LU decomposition and non-closedness

$A =$ 

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

  [H. Golub and C. F. Van Loan, **Matrix Computations, The Johns Hopkins University Press**]

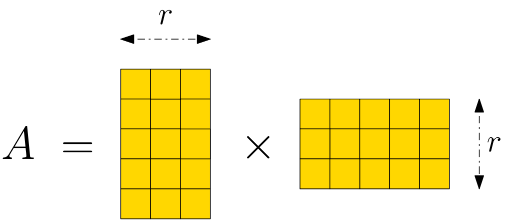
$\longrightarrow$ The set of matrices having LU decomposition is not closed
$\longrightarrow$ For certain support constraints (S_X, S_Y) and matrices A, (FSMF) does not have an optimal solution.
Question : Given (S_X, S_Y), do their FSMF instances always have optimal solutions? $\rightarrow$ This problem is *decidable* (not known to be *tractable* yet).

Polynomially solvable instances

Example (Unconstrained matrix factorization)

When there is no constraints on the support of X and Y:

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

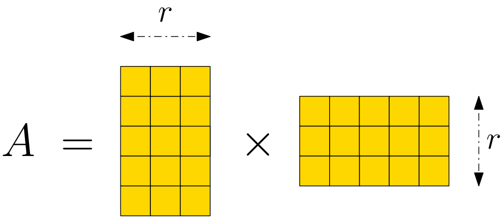


Polynomially solvable instances

Example (Unconstrained matrix factorization)

When there is no constraints on the support of X and Y:

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



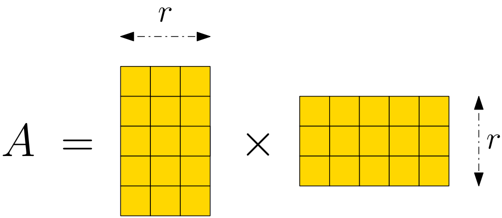
→ Solution: Use Singular Value Decomposition (SVD).

Polynomially solvable instances

Example (Unconstrained matrix factorization)

When there is 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).

→ Question: Can Singular Value Decomposition (SVD) still work in the *constrained* case?

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:

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

2) Finding the SVD:

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

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:

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

$\to$ SVD is a greedy algorithm in disguise

Algorithm 1 Algorithm for *unconstrained* matrix factorization

1: **for** $i \in \{1, \dots, r\}$ **do**
2: $M_i :=$ 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

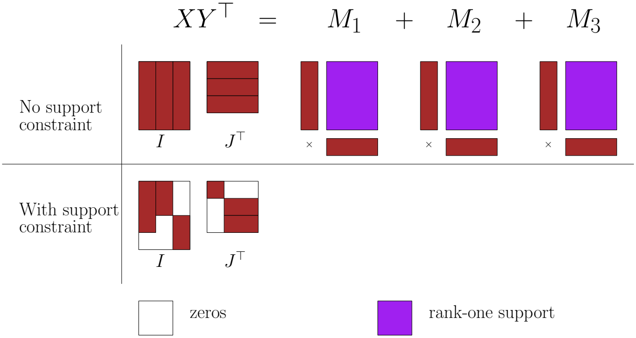
- How to generalize the greedy algorithm?
- Decompose $XY^\top$:

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

SVD as a greedy algorithm: the *constrained* case

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

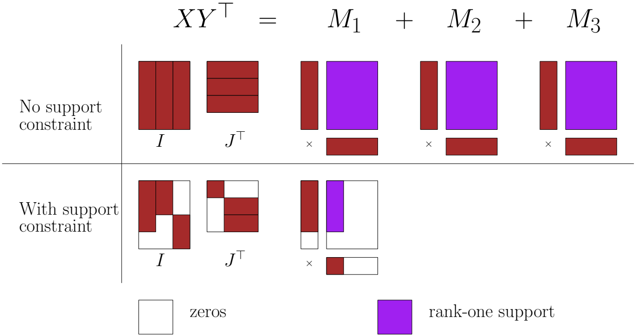
$$XY^\top = \sum_{i=1}^{r} x_i y_i^\top = \sum_{i=1}^{r} \underbrace{M_i}_{\text{rank-one}} \qquad (M_i := x_i y_i^\top)$$



SVD as a greedy algorithm: the *constrained* case

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

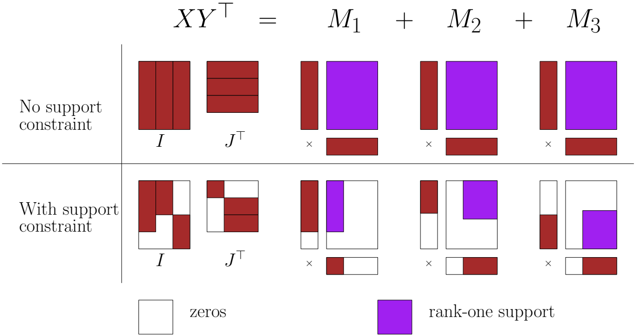
$$XY^\top = \sum_{i=1}^{r} x_i y_i^\top = \sum_{i=1}^{r} \underbrace{M_i}_{\text{rank-one}} \qquad (M_i := x_i y_i^\top)$$



SVD as a greedy algorithm: the *constrained* case

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

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

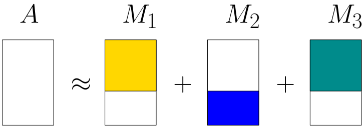


- Finding optimal solution (X, Y) $\rightleftarrows$ Finding optimal the rank-one constrained supports.

Algorithm 2 Algorithm for fixed-support matrix factorization

for $i \in \{1, \dots, r\}$ **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

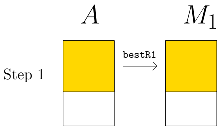
Example:



Algorithm 2 Algorithm for fixed-support matrix factorization

```
for i ∈ {1, ..., r} do
    S_i ← i-th rank-one support
    M_i := best rank-one approximation of (A − Σ_{k=1}^{i−1} M_k) ⊙ S_i
end for
```

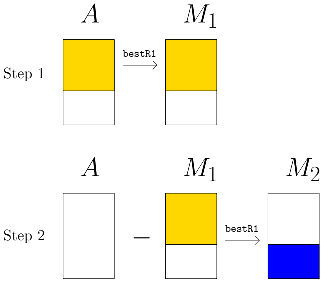
Example:



Algorithm 2 Algorithm for fixed-support matrix factorization

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

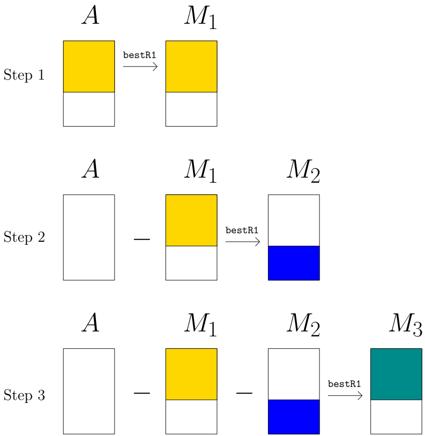
Example:



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

Example:



Polynomial solvability characterized by rank-one supports

- The output of the greedy algorithm will always *satisfy the constraints*
- Is the output an optimal solution? Not always

Theorem (2)

If the rank-one supports are pairwise disjoint or identical, then the greedy algorithm gives an optimal solution.

For the same result with a weaker assumption:

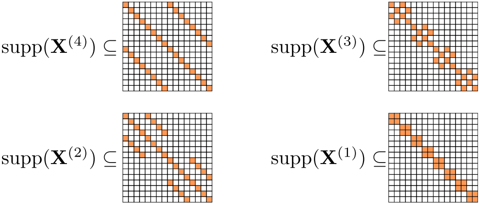
[QT. Le, E. Riccietti, R. Gribonval, **Spurious Valleys, NP-hardness, and Tractability of Sparse Matrix Factorization With Fixed Support**, SIAM Journal of Matrix Analysis and Applications, In press, 2022.]

Multiple-factors cases: the butterfly factorization

A special case: the butterfly factorization

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

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

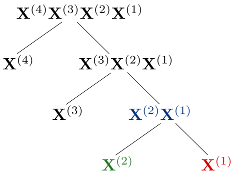


- It is expressive: the composition of matrices with a butterfly structure can accurately approximate any given matrix
- In neural networks faster training and inference time without harming the performance

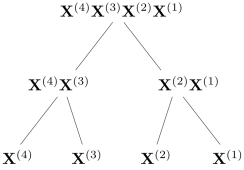
[T. Dao & all. Kaleidoscope: An efficient, learnable representation for all structured linear maps, ICLR, 2020]
[B. Chen & all. Pixelated butterfly: Simple and efficient sparse training for neural network models, PMLR, 2022]

Butterfly factorization: theoretical guarantees

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



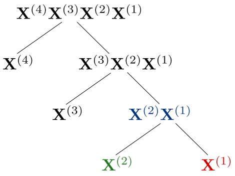
Left-to-right tree



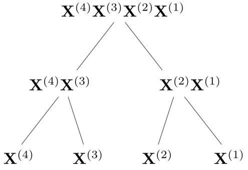
Balanced tree

Butterfly factorization: theoretical guarantees

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



Left-to-right tree



Balanced tree

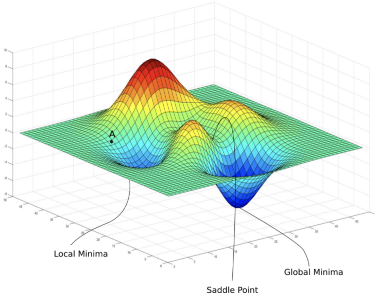
Theorem (3)

If E^* is best error approximation of $A \in \mathbb{R}^{N\times N}$, the solution of the hierarchical algorithm yields a distance/error E such that:

- *Any tree:* $E \leq (N/2 - 1) \times E^*$.
- *Left-to-right (or right-to-left) tree:* $E \leq N^c \times E^*$ *where* $c = \log_4 3 < 1$.

Study of the landscape of the loss function

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



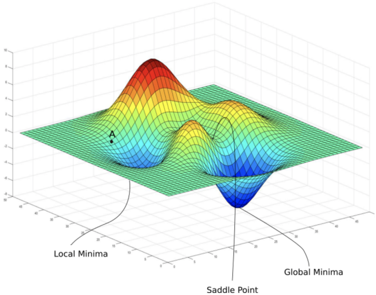
Has been studied for:

- 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]

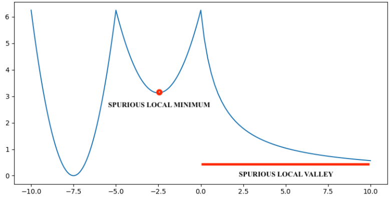
Never with support constraints!

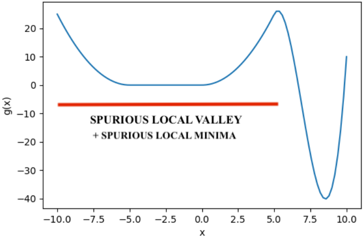
Undesirable spurious objects

Example of undesirable spurious objects :

- Spurious local minima: local minima (but not global minima)
- Spurious local valleys: less known but equally troublesome

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





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

Landscape of full support matrix factorization

- With unconstrained (full support) matrix factorization:

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

- 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 full support matrix factorization

- With unconstrained (full support) matrix factorization:

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

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

→ Are there other instances similar to full support matrix factorization ?

[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

Reminder : Fixed support matrix factorization

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

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

Theorem (4)

If (S_X, S_Y) satisfy the condition of polynomial solvability in Theorem (2), then for all A, the landscape of $L(X, Y)$ does not contain any spurious local minimum and spurious local valley.

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) Multiple factors can be dealt with by a hierarchical approach.

On-going works/perspectives

- Study the closedness of the set of solutions to FSMF.
- Can we enlarge the family of tractable instances of FSMF?
- Can we improve the approximate factor in the case of *butterfly factorization*?

Available: an implementation of the algorithm in C++ via Python and Matlab wrappers FAμST toolbox at `https://faust.inria.fr/`.

To know more:

- Q.-T. Le, E. Riccietti, and R. Gribonval (2022), Spurious Valleys, Spurious Minima and NP-hardness of Sparse Matrix Factorization With Fixed Support, *arXiv preprint*, arXiv:2112.00386.
- L. Zheng, E. Riccietti, and R. Gribonval (2022), Efficient Identification of Butterfly Sparse Matrix Factorizations, *arXiv preprint*, arXiv:2110.01235.
- Q.-T. Le, L. Zheng, E. Riccietti, and R. Gribonval (2022), Fast learning of fast transforms, with guarantees, *ICASSP 2022*

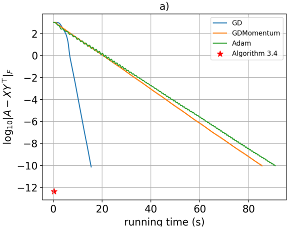
Definition: spurious local valleys

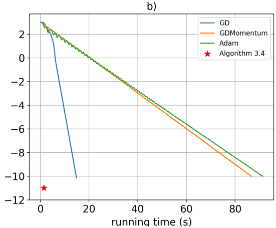
Definition (Spurious local valley - Informal)

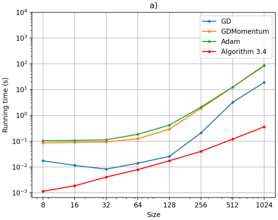
$S \in \mathbb{R}^d$ is a spurious local valley if for all $x \in S$, there does not exist any *continuous path* connecting x and a global minimum x^* *without increasing* the loss function f.

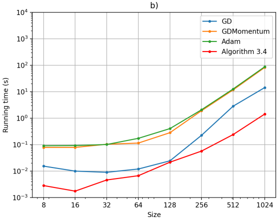
Numerical results: 2 factors

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





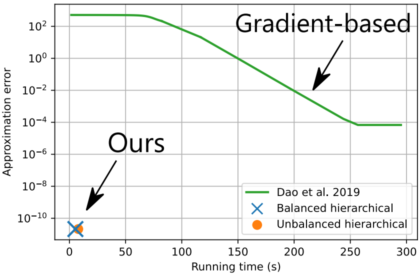




Numerical results: J factors

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