Multi-layer Sparse Matrix Factorization

Luc Le Magoarou Rémi Gribonval

Inria
Centre Inria Rennes - Bretagne Atlantique
France

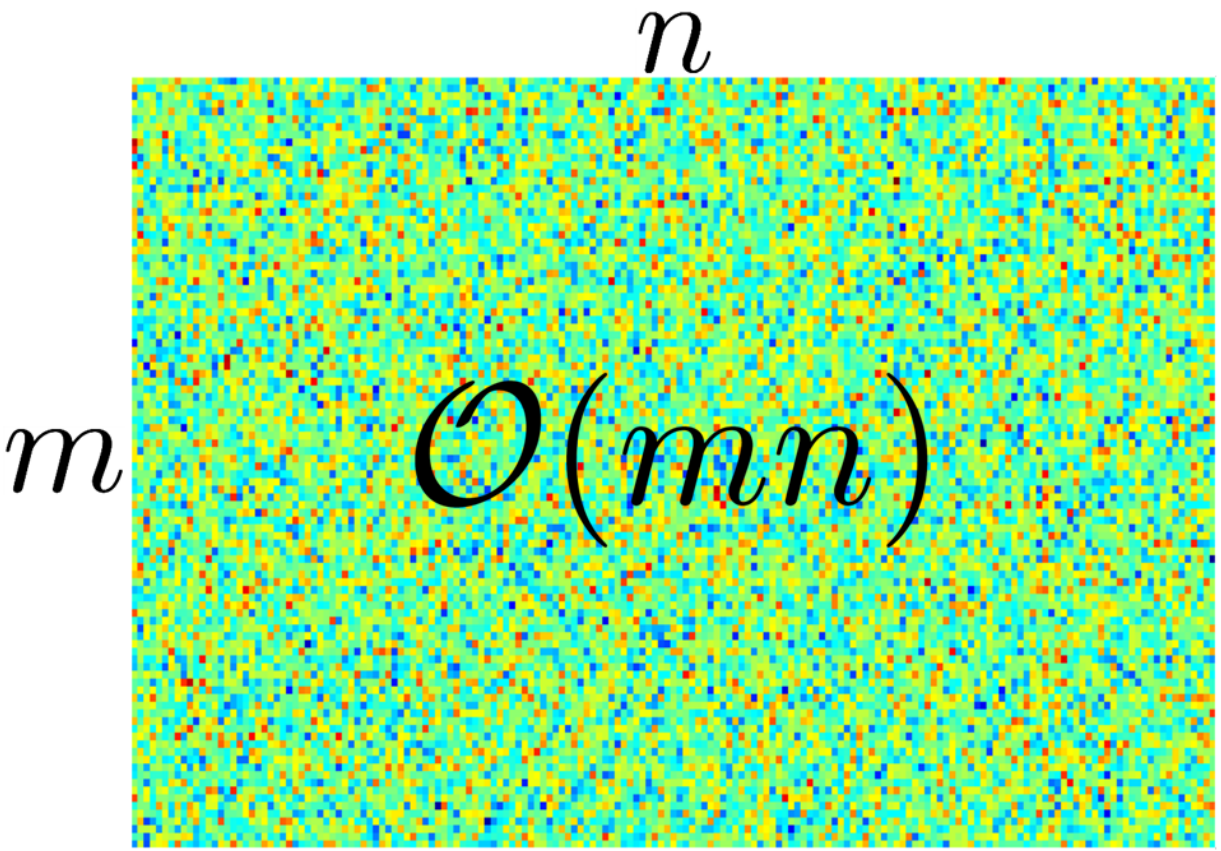
SPARS 2015

Outline

Outline

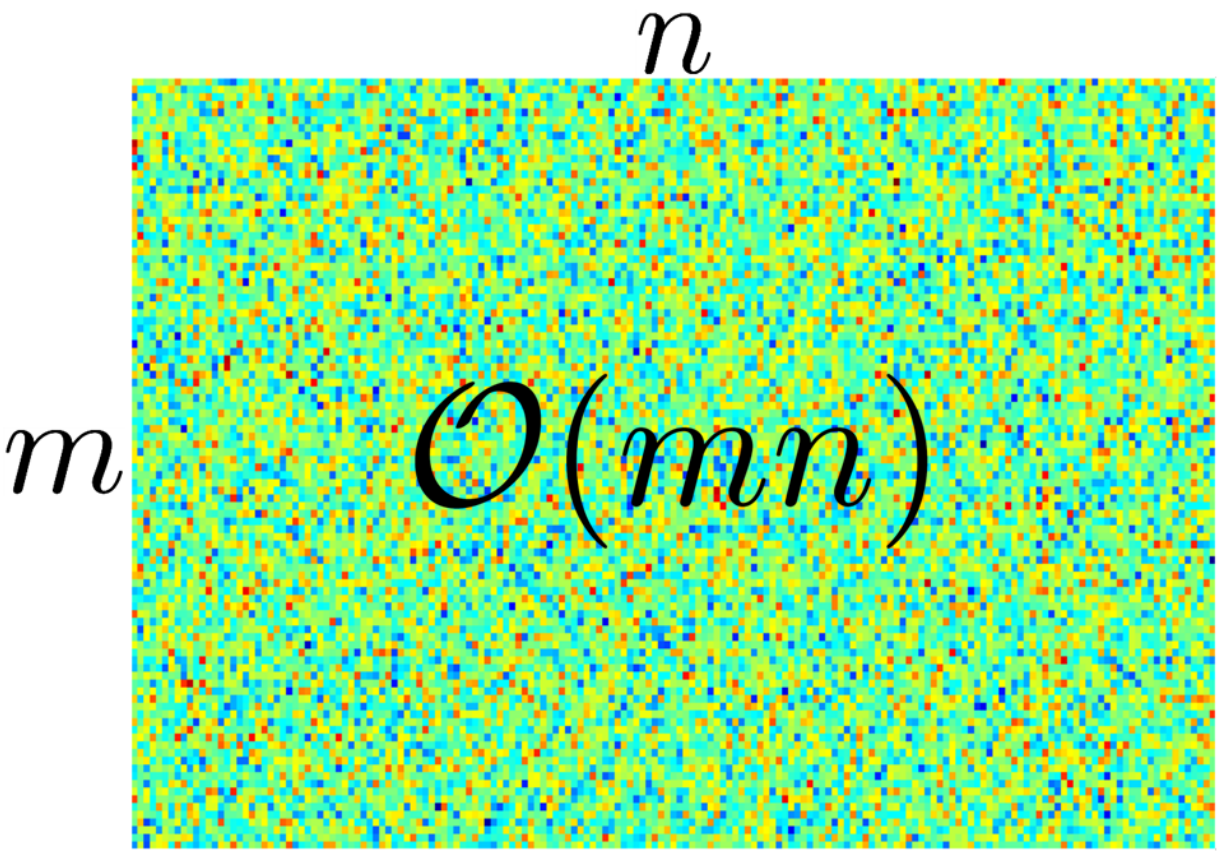
Motivation

Manipulation of dense matrices is costly in high dimension.



Motivation

Manipulation of dense matrices is costly in high dimension.

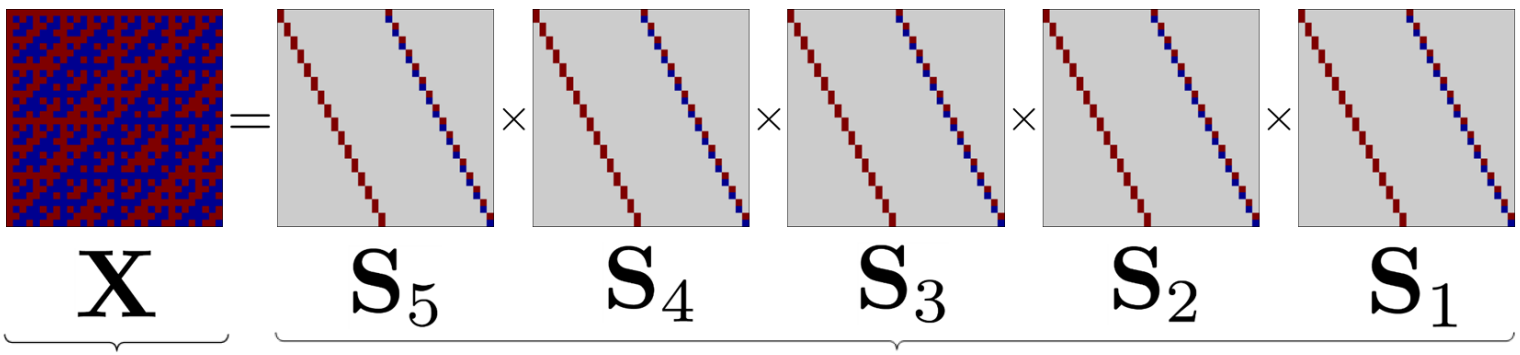


Is it possible to do better?

Fast transforms as sparse factorizations

Analytic transforms (Fourier, wavelets, Hadamard, DCT...) lead to fast algorithms because of their factorizable structure[1] :

$$\mathbf{X} = \prod_{j=1}^{J} \mathbf{S}_j$$



[1] J. Morgenstern, **The Linear Complexity of Computation.** *J. ACM*, 1975

Objective

Our goal is to find multi-layer sparse approximations:

$$\underset{\text{known}}{\mathbf{X}} \approx \underbrace{\prod_{j=1}^{J} \mathbf{S}_j}_{\text{unknown}},$$

and get Flexible Approximate MUlti-layer Sparse Transforms (FAμST) associated to matrices $\mathbf{X}$ of interest:

- Dictionaries
- Forward operators of inverse problems
- ...

Multi-layer sparse benefits

FAμSTs have several advantages over dense matrices:

- Lower *storage cost*
- Higher *speed of multiplication*
- Improved *statistical significance*

Relative complexity

Gains are related to the Relative Complexity (RC) defined as:

$$\text{RC} \triangleq \frac{\sum_{j=1}^{J} \|\mathbf{S}_j\|_0}{\|\mathbf{X}\|_0}$$

Outline

Optimization problem

- Input: matrix $\mathbf{X} \in \mathbb{R}^{m \times n}$
- Goal: find J sparse matrices $\mathbf{S}_j$ such that $\mathbf{X} \approx \mathbf{S}_J \dots \mathbf{S}_1$
- Approach:

$$\underset{\lambda, \mathbf{S}_1, \dots, \mathbf{S}_J}{\text{Minimize}} \quad \underbrace{\frac{1}{2} \left\| \mathbf{X} - \lambda \prod_{j=1}^{J} \mathbf{S}_j \right\|_F^2}_{\text{Data fitting}} + \underbrace{\sum_{j=1}^{J} \delta_{\mathcal{E}_j}(\mathbf{S}_j)}_{\text{Sparsity enforcing}}$$

Sparsity enforcing term: indicator functions of sets of the form, e.g. $\mathcal{E}_j = \{\mathbf{A} \in \mathbb{R}^{a_j \times a_{j+1}} : \|\mathbf{A}\|_0 \leq p_j, \|\mathbf{A}\|_F = 1\}$.

This optimization problem is highly non-convex and non-smooth.

PALM for Multi-layer Sparse Approximations

The Proximal Alternating Linearized Minimization (PALM)[2] algorithm can be used with:

$$H(\lambda, \mathbf{S}_1, \ldots, \mathbf{S}_J) \triangleq \tfrac{1}{2}\left\|\mathbf{X} - \lambda \prod_{j=1}^{J} \mathbf{S}_j\right\|_F^2,$$

and:

$$\mathcal{E}_j \triangleq \{\mathbf{A} \in \mathbb{R}^{a_j \times a_{j+1}} : \|\mathbf{A}\|_0 \leq p_j, \|\mathbf{A}\|_F = 1\}.$$

Algorithm 1 `palm4MSA` iteration

1: **for** $j = 1$ to J **do**

2: $\quad \mathbf{S}_j^{i+1} \leftarrow P_{\mathcal{E}_j}\left(\mathbf{S}_j^i - \frac{1}{c_j^i}\nabla_{\mathbf{S}_j} H\left(\lambda^i, \mathbf{S}_1^{i+1}, \ldots, \mathbf{S}_j^i, \ldots, \mathbf{S}_J^i\right)\right)$

3: **end for**

Proposition. Each bounded sequence generated by `palm4MSA` converges to a stationary point of the objective.

[2] J. Bolte et al., **Proximal alternating linearized minimization for nonconvex and nonsmooth problems.** *Math. Program.*, 2013.

Hierarchical strategy

In order to initialize the factors in a good region, we adopt a hierarchical factorization strategy, reminiscent of layerwise training of neural networks[3]:

This hierarchical factorization is surprisingly effective and the attained local minima are very good.

[3] G. Hinton and R. Salakhutdinov, **Reducing the dimensionality of data with neural networks**, *Science*, vol. 313, no. 5786, 2006.

Hierarchical strategy

In order to initialize the factors in a good region, we adopt a hierarchical factorization strategy, reminiscent of layerwise training of neural networks[3]:

$$\mathbf{X} \approx \mathbf{S}_1 \mathbf{R}_1$$

This hierarchical factorization is surprisingly effective and the attained local minima are very good.

[3] G. Hinton and R. Salakhutdinov, **Reducing the dimensionality of data with neural networks**, *Science*, vol. 313, no. 5786, 2006.

Hierarchical strategy

In order to initialize the factors in a good region, we adopt a hierarchical factorization strategy, reminiscent of layerwise training of neural networks[3]:

$$\mathbf{X} \approx \mathbf{S}_1 \mathbf{R}_1 \approx \mathbf{S}_2 \mathbf{R}_2$$

This hierarchical factorization is surprisingly effective and the attained local minima are very good.

[3] G. Hinton and R. Salakhutdinov, **Reducing the dimensionality of data with neural networks**, *Science*, vol. 313, no. 5786, 2006.

Hierarchical strategy

In order to initialize the factors in a good region, we adopt a hierarchical factorization strategy, reminiscent of layerwise training of neural networks[3]:

$$\mathbf{X} \approx \mathbf{S}_1 \mathbf{R}_1$$
$$\approx \mathbf{S}_2 \mathbf{R}_2$$
$$\approx \mathbf{S}_3 \ddots \mathbf{S}_{J-1} \mathbf{S}_J$$

This hierarchical factorization is surprisingly effective and the attained local minima are very good.

[3] G. Hinton and R. Salakhutdinov, **Reducing the dimensionality of data with neural networks**, *Science*, vol. 313, no. 5786, 2006.

Outline

Inverse problems (joint work with A. Gramfort )

Data $\mathbf{y}$ and parameters $\boldsymbol{\gamma}$ are linked through the operator $\mathbf{M}$:

$$\mathbf{y} \approx \mathbf{M}\boldsymbol{\gamma}$$

Recovery methods are often iterative algorithms relying on applications of the operator $\mathbf{M}$, which can be costly in high dimension.

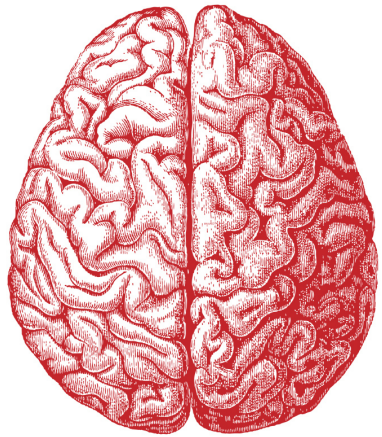
Inverse problems (joint work with A. Gramfort )

Data $\mathbf{y}$ and parameters $\boldsymbol{\gamma}$ are linked through the operator $\mathbf{M}$:

$$\mathbf{y} \approx \underbrace{\mathbf{M}\boldsymbol{\gamma}}_{J} \prod_{j=1} \mathbf{S}_j$$

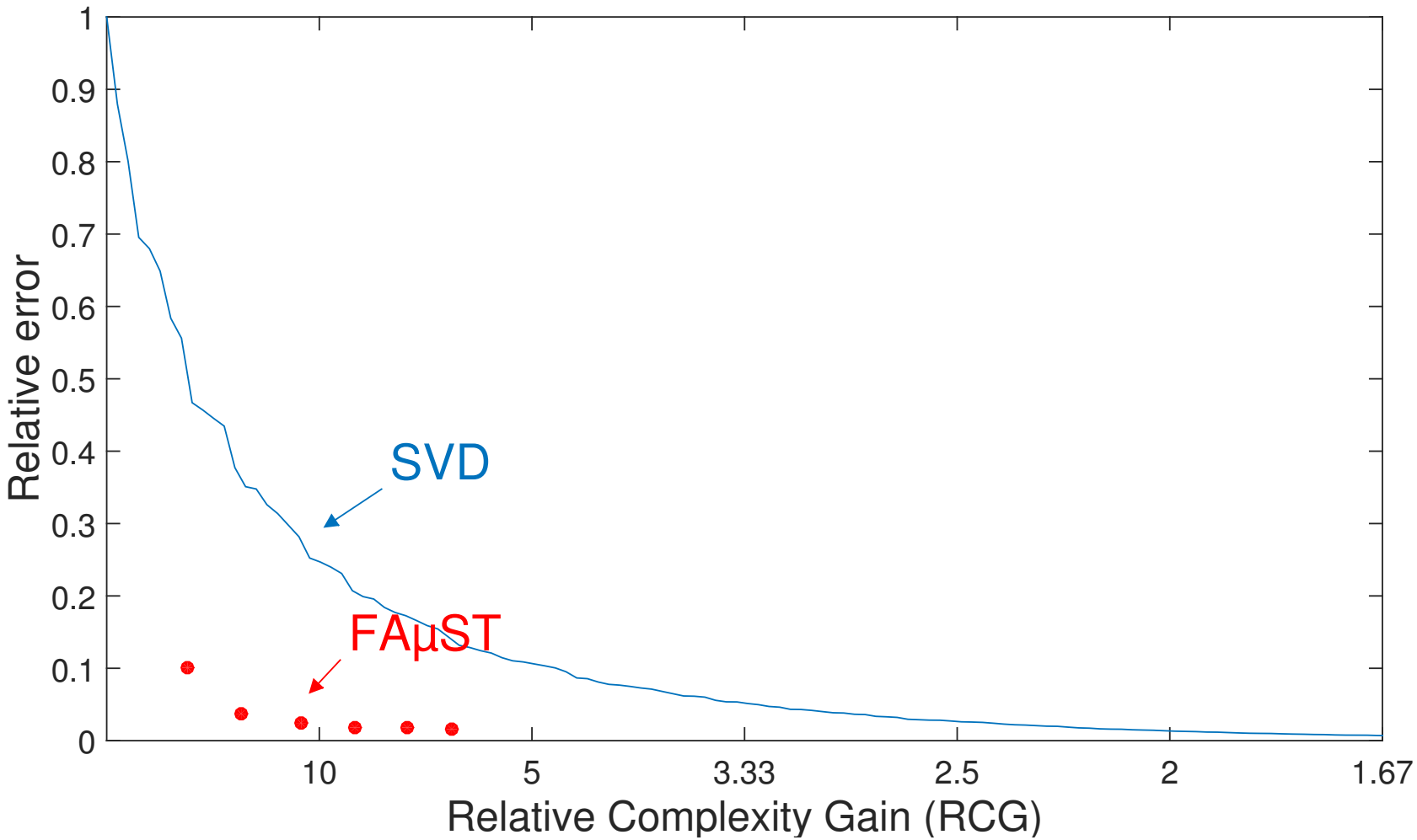
Recovery methods are often iterative algorithms relying on applications of the operator $\mathbf{M}$, which can be costly in high dimension.

Inverse problems: MEG imaging



- $\boldsymbol{\gamma} \in \mathbb{R}^{8193}$ represents electric sources at different locations.
- $\mathbf{y} \in \mathbb{R}^{204}$ is the signal intensity measured by electrodes.
- $\mathbf{M} \in \mathbb{R}^{204 \times 8193}$ models the physics of the propagation (Maxwell's equations).

Inverse problems: Factorization of $\mathbf{M}$



Inverse problems: Source localization experiment

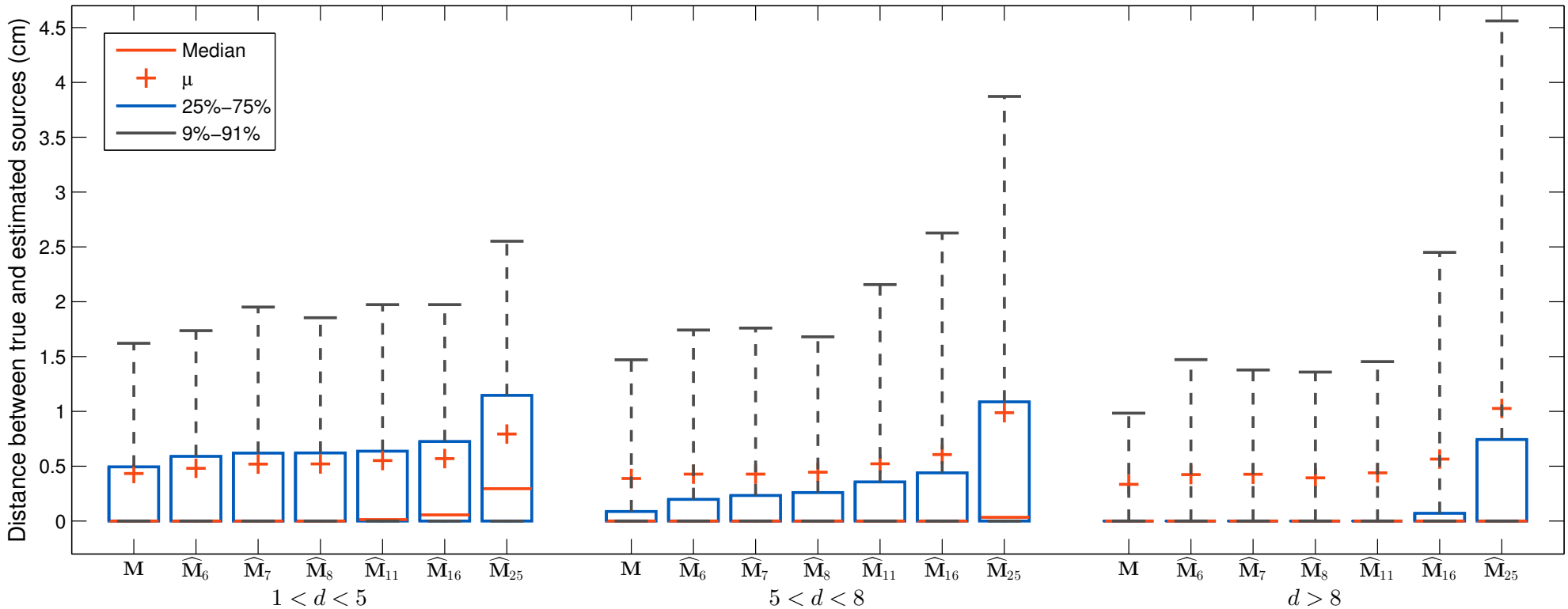
Experiment: The objective is to retrieve the location of 2 brain sources chosen uniformly at random, activated with gaussian random weights, giving a 2-sparse vector $\boldsymbol{\gamma} \in \mathbb{R}^{8193}$.

Resolution method: Orthogonal Matching Pursuit (OMP), choosing 2 atoms.

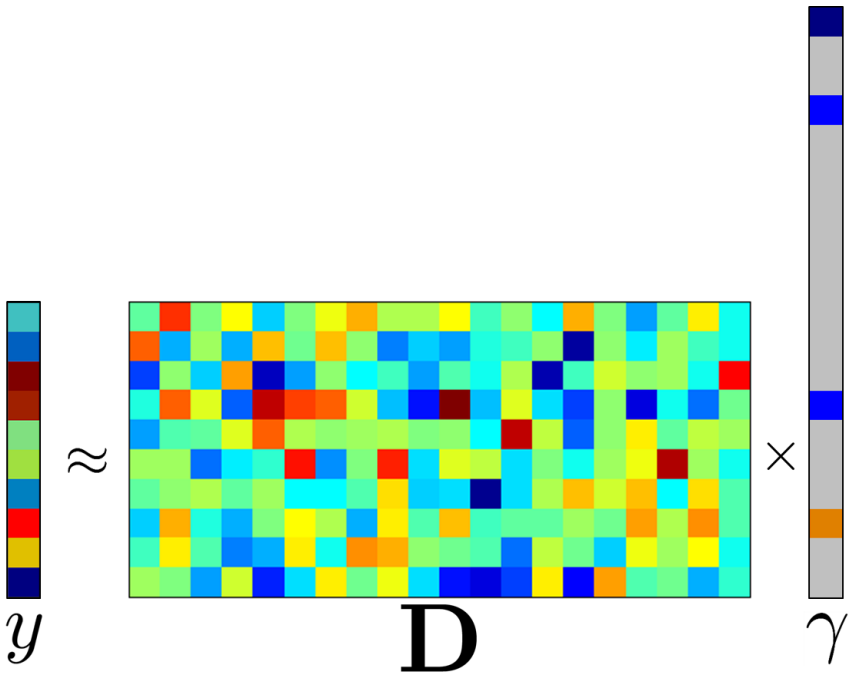
Matrix used:

- The actual matrix $\mathbf{M}$.
- The FAμSTs $\widehat{\mathbf{M}}_{25}$, $\widehat{\mathbf{M}}_{16}$, $\widehat{\mathbf{M}}_{11}$, $\widehat{\mathbf{M}}_{8}$, $\widehat{\mathbf{M}}_{7}$, $\widehat{\mathbf{M}}_{6}$, where the subscript indicates the achieved RCG (rounded to the closest integer).

Inverse problems: Source localization results

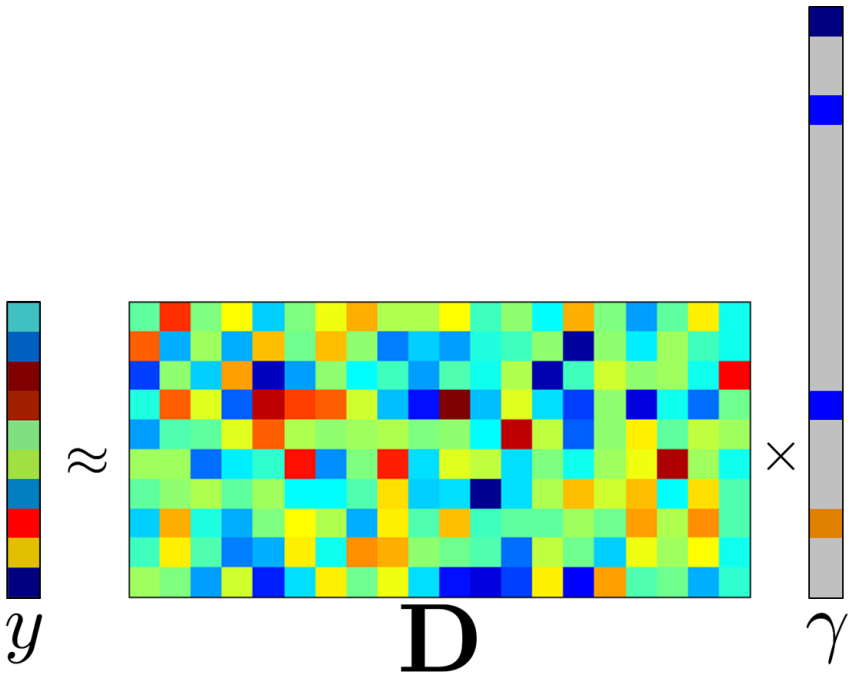


Dictionary learning



$$\mathbf{Y} \approx \mathbf{D}\Gamma$$

Dictionary learning



$$\mathbf{Y} \approx \underbrace{\mathbf{D}\boldsymbol{\Gamma}}_{J} \prod_{j=1} \mathbf{S}_j$$

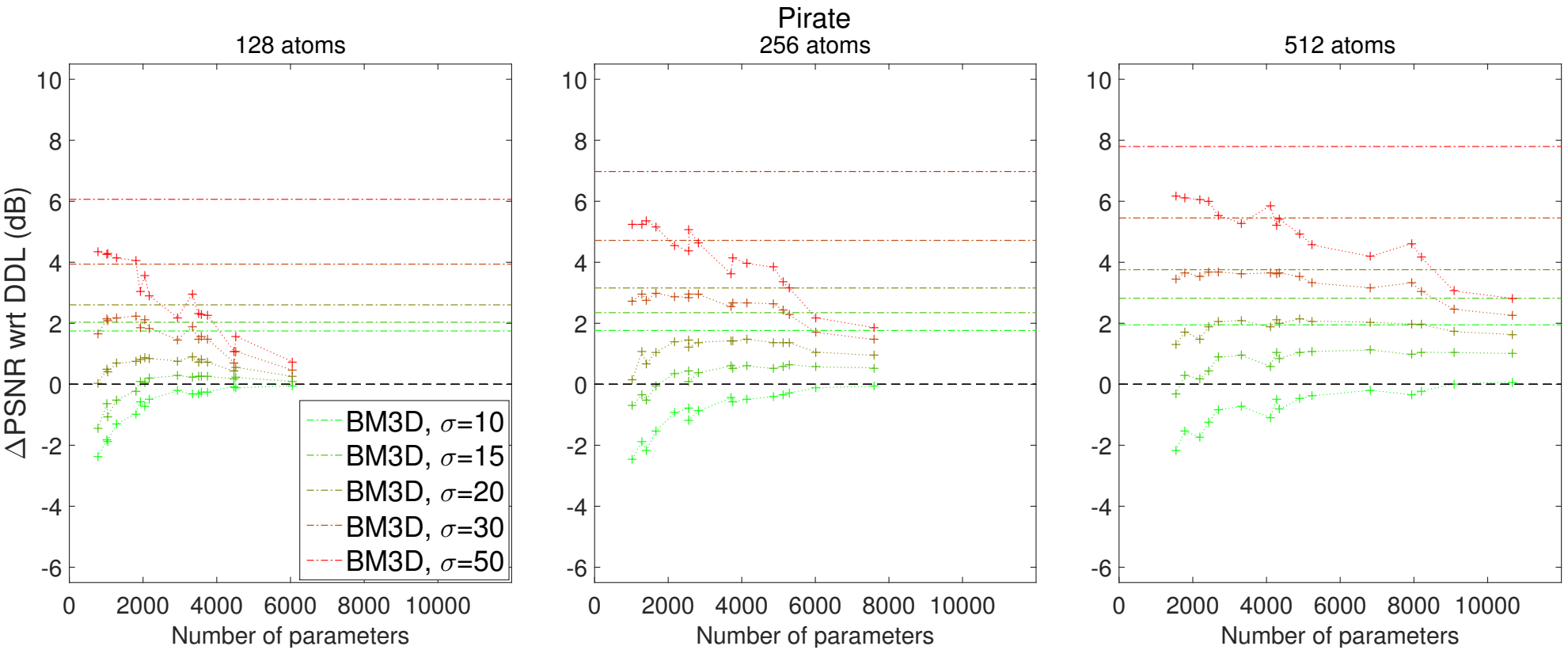
Dictionary learning: Experimental settings

Experiment: 8×8 noisy image patches are gathered in $\mathbf{Y} \in \mathbb{R}^{64 \times 10000}$, on which a dictionary $\mathbf{D}$ is learned: $\mathbf{Y} \approx \mathbf{D}\mathbf{\Gamma}$, the coefficient matrix $\mathbf{\Gamma}$ having 5 non-zero entries per column. The learned dictionary is then used to denoise the whole image using OMP.

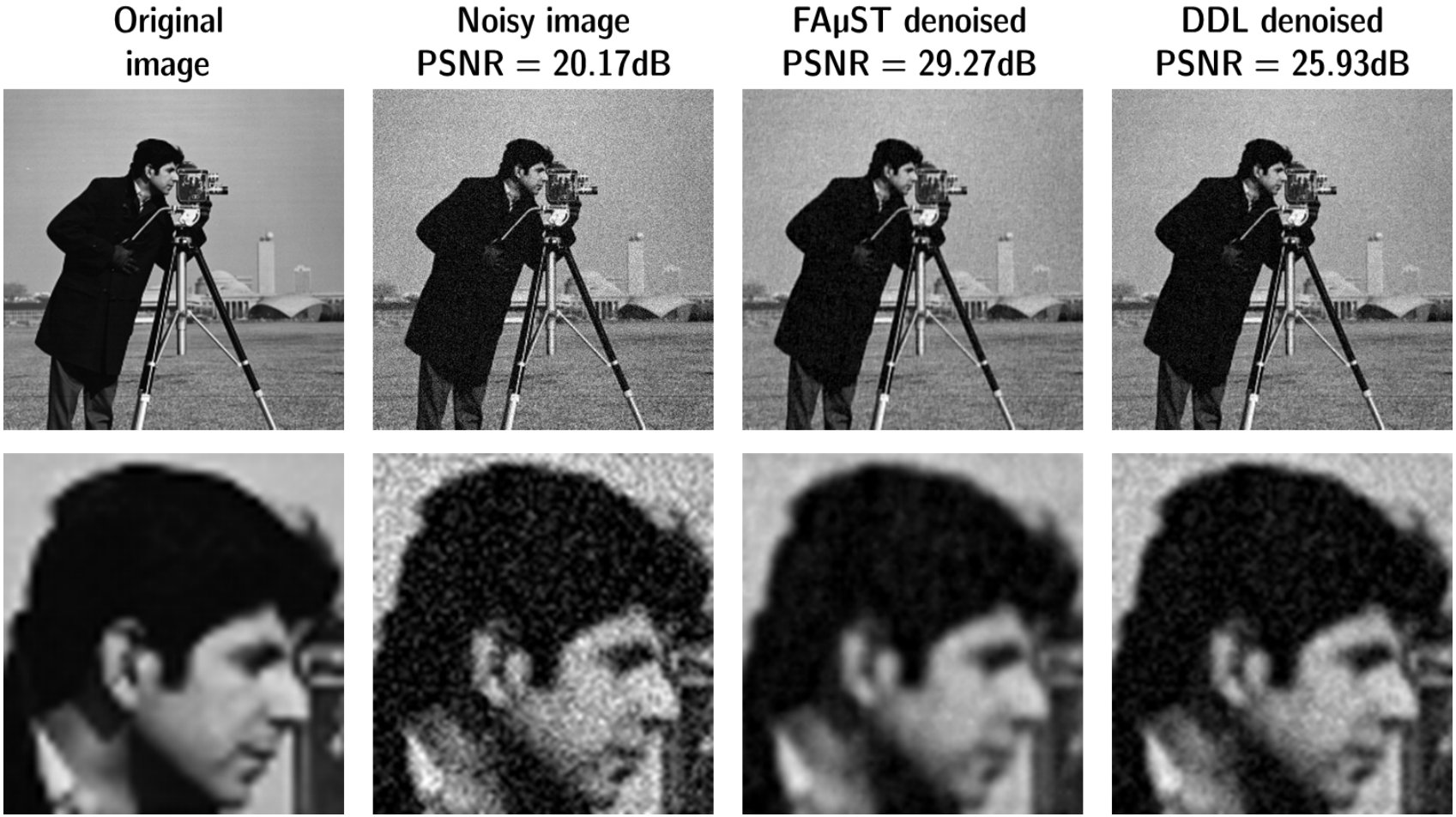
Dictionary learning methods:

- FAμST: $\mathbf{D} = \prod_{j=1}^{J} \mathbf{S}_j$
- Dense Dictionary Learning (DDL): $\mathbf{D}$ is unconstrained

Dictionary learning: Image denoising results

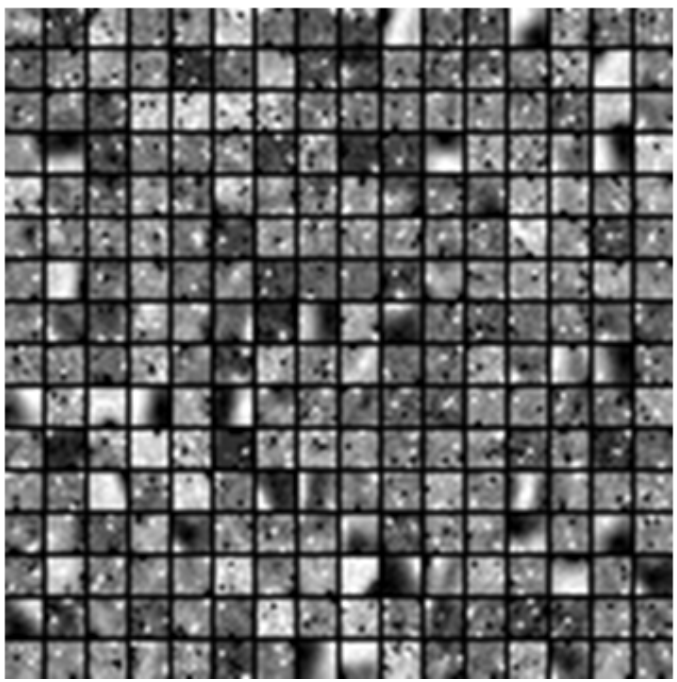


Dictionary learning: Image denoising example

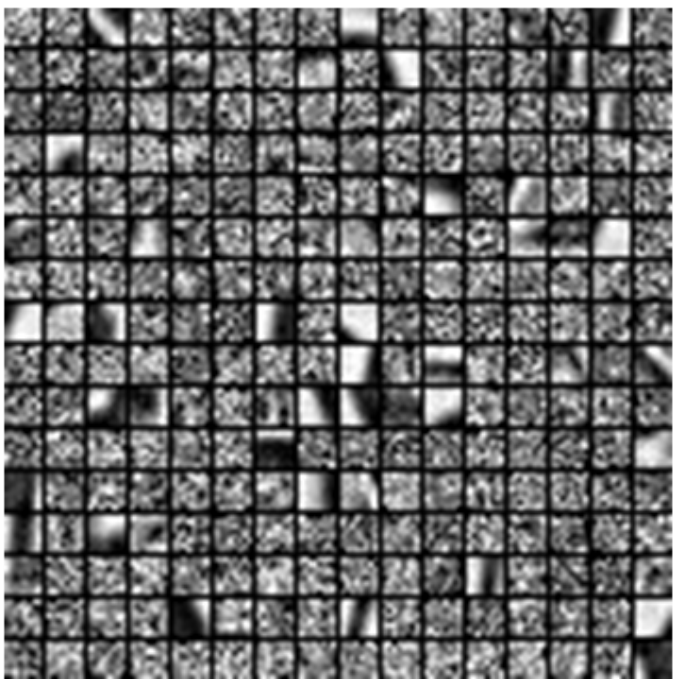


Dictionary learning: Image denoising example

FAμST Dictionary (RC=0,13)



DDL Dictionary (RC=1)



Dictionary learning: Generalization bound

General result applicable to various dictionary classes, distributions and penalties[4]:

$$\sup_{\mathbf{D}\in\mathfrak{D}} |F_{\mathbf{X}}(\mathbf{D}) - \mathbb{E}_{\mathbf{x}\sim\mathbb{P}} f_{\mathbf{x}}(\mathbf{D})| \leq \eta_n(g, \mathfrak{D}, \mathfrak{P}),$$

with $\eta_n \propto \sqrt{d(\mathfrak{D})}$. For multi-layer sparse dictionaries, we have:

$$d(\mathfrak{D}) = \sum_{j=1}^{J} \|\mathbf{S}_j\|_0 \,.$$

This gives $\eta_n \propto \mathcal{O}\left(\sqrt{\sum_{j=1}^{J} \|\mathbf{S}_j\|_0}\right)$ instead of $\mathcal{O}\left(\sqrt{\|\mathbf{D}\|_0}\right)$ for classical dense dictionaries.

[4] R. Gribonval et al., **Sample Complexity of Dictionary Learning and other Matrix Factorizations.** *IEEE Trans. Inf. Theory.* 2015.

Outline

Conclusion

Summary:

- A new matrix factorization method with complexity constraints.
- An improved computational efficiency with good adaptation to the training data.

Ongoing and future work:

- Task-driven dictionary learning.
- Signal processing on graphs.
- Theoretical analysis of multi-layer sparse approximations.

Questions?

Le Magoarou, L. & Gribonval, R., **Flexible Multi-layer Sparse Approximations of Matrices and Applications**, arXiv:1506.07300.

PALM convergence conditions

The following conditions are sufficient (not necessary) to ensure that each bounded sequence generated by PALM converges to a stationary point of its objective:

1. H is smooth.
2. The $\mathcal{E}_j$s are semi-algebraic sets.
3. $\nabla_{\mathbf{x}_j} H$ is globally Lipschitz for all j, with Lipschitz moduli $L_j(\mathbf{x}_1 \ldots \mathbf{x}_{j-1}, \mathbf{x}_{j+1} \ldots \mathbf{x}_N)$.
4. $\forall i,\ c_j^i > L_j(\mathbf{x}_1^{i+1} \ldots \mathbf{x}_{j-1}^{i+1}, \mathbf{x}_{j+1}^i \ldots \mathbf{x}_N^i)$ (the inequality need not be strict for convex f_j).

The palm4MSA algorithm

Algorithm 2 PALM for Multi-layer Sparse Approximations

Input: Matrix $\mathbf{X}$, desired number of factors J, constraint sets $\mathcal{E}_j$, $j \in \{1 \dots J\}$ and a stopping criterion.

1: **for** $i = 0$ to $N_{iter} - 1$ **do**
2: **for** $j = 1$ to J **do**
3: Set $c_j^i > (\lambda^i)^2 \left\|\mathbf{R}\right\|_2^2 . \left\|\mathbf{L}\right\|_2^2$
4: $\mathbf{S}_j^{i+1} \leftarrow P_{\mathcal{E}_j}\left(\mathbf{S}_j^i - \frac{1}{c_j^i}\lambda \mathbf{L}^T(\lambda \mathbf{L}\mathbf{S}_j^i\mathbf{R} - \mathbf{X})\mathbf{R}^T\right)$
5: **end for**
6: $\lambda^{i+1} \leftarrow \frac{\mathrm{Tr}(\mathbf{X}^T\hat{\mathbf{X}})}{\mathrm{Tr}(\hat{\mathbf{X}}^T\hat{\mathbf{X}})}$
7: **end for**

Output: $\lambda^{N_{iter}}, \{\mathbf{S}_k^{N_{iter}}\}_{k=1}^J = \texttt{palm4MSA}(\mathbf{X}, J, \{\mathcal{E}_j\}_{j=1}^J)$

Proposition. Each bounded sequence generated by palm4MSA converges to a stationary point of the objective.

Hierarchical factorization algorithm

Algorithm 3 Hierarchical factorization

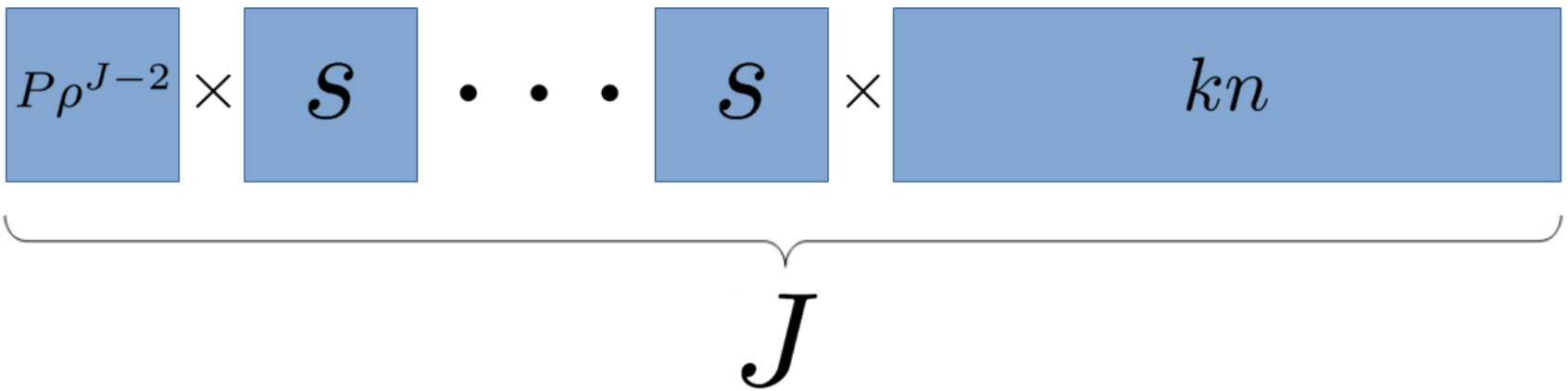
Input: Matrix $\mathbf{X}$, desired number of factors J and the constraint sets $\mathcal{E}_k,\ k \in \{1 \dots J-1\}$ and $\tilde{\mathcal{E}}_k,\ k \in \{1 \dots J-1\}$.

1: $\mathbf{R} \leftarrow \mathbf{X}$
2: **for** $k = 1$ to $J-1$ **do**
3: $\quad \lambda', \{\mathbf{T}_1, \mathbf{T}_2\} = \texttt{palm4MSA}(\mathbf{R},\ 2,\ \{\mathcal{E}_k, \tilde{\mathcal{E}}_k\})$
4: $\quad \mathbf{S}_k \leftarrow \lambda' \mathbf{T}_1$ and $\mathbf{R} \leftarrow \mathbf{T}_2$
5: $\quad \lambda, \left\{\{\mathbf{S}_j\}_{j=1}^{k}, \mathbf{R}\right\} = \texttt{palm4MSA}(\mathbf{X},\ k+1,\ \left\{\{\mathcal{E}_j\}_{j=1}^{k}, \tilde{\mathcal{E}}_k\right\})$
6: **end for**
7: $\mathbf{S}_J \leftarrow \mathbf{R}$

Output: $\lambda, \{\mathbf{S}_k\}_{k=1}^{J}$.

Inverse problems: Factorization of $\mathbf{M}$

Objective: Factorize $\mathbf{M}$ in order to make complexity savings.



What complexity/accuracy trade-offs are achievable?

Dictionary learning: Algorithm

Algorithm 4 Hierarchical factorization for dictionary learning

Input: Data matrix $\mathbf{Y}$; Dictionary $\mathbf{D}$; Coefficients $\mathbf{\Gamma}$; desired number of factors J; constraint sets $\mathcal{E}_k$ and $\tilde{\mathcal{E}}_k$, $k \in \{1 \dots J-1\}$.

1: $\mathbf{T}_0 \leftarrow \mathbf{D}$
2: **for** $k = 1$ to $J-1$ **do**
3: Factorize the residual $\mathbf{T}_{k-1}$ into 2 factors:
$\lambda', \{\mathbf{F}_2, \mathbf{F}_1\} = \texttt{palm4MSA}(\mathbf{T}_{k-1},\ 2,\ \{\tilde{\mathcal{E}}_k, \mathcal{E}_k\},\ \dots)$
4: $\mathbf{T}_k \leftarrow \lambda' \mathbf{F}_2$ and $\mathbf{S}_k \leftarrow \mathbf{F}_1$
5: Global optimization using `palm4MSA`
6: Coefficients update:
$\mathbf{\Gamma} = \texttt{sparseCoding}(\mathbf{Y},\ \mathbf{T}_k \prod_{j=1}^{k} \mathbf{S}_j)$
7: **end for**
8: $\mathbf{S}_J \leftarrow \mathbf{T}_{J-1}$

Output: The estimated factorization: $\lambda, \{\mathbf{S}_j\}_{j=1}^{J}$, $\mathbf{\Gamma}$.

Dictionary learning: Algorithm

Algorithm 5 Hierarchical factorization for dictionary learning

Input: Data matrix $\mathbf{Y}$; Dictionary $\mathbf{D}$; Coefficients $\mathbf{\Gamma}$; desired number of factors J; constraint sets $\mathcal{E}_k$ and $\tilde{\mathcal{E}}_k$, $k \in \{1 \dots J-1\}$.

1: $\mathbf{T}_0 \leftarrow \mathbf{D}$
2: **for** $k = 1$ to $J-1$ **do**
3: Factorize the residual $\mathbf{T}_{k-1}$ into 2 factors:
$\lambda', \{\mathbf{F}_2, \mathbf{F}_1\} = \texttt{palm4MSA}(\mathbf{T}_{k-1}, 2, \{\tilde{\mathcal{E}}_k, \mathcal{E}_k\}, \dots)$
4: $\mathbf{T}_k \leftarrow \lambda' \mathbf{F}_2$ and $\mathbf{S}_k \leftarrow \mathbf{F}_1$
5: Global optimization using `palm4MSA`
6: Coefficients update:
$\mathbf{\Gamma} = \texttt{sparseCoding}(\mathbf{Y}, \mathbf{T}_k \prod_{j=1}^{k} \mathbf{S}_j)$
7: **end for**
8: $\mathbf{S}_J \leftarrow \mathbf{T}_{J-1}$

Output: The estimated factorization: $\lambda, \{\mathbf{S}_j\}_{j=1}^{J}$, $\mathbf{\Gamma}$.

Dictionary learning: Algorithm

Algorithm 6 Hierarchical factorization for dictionary learning

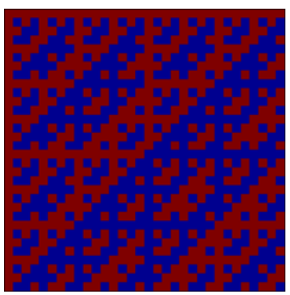
Input: Data matrix $\mathbf{Y}$; Dictionary $\mathbf{D}$; Coefficients $\mathbf{\Gamma}$; desired number of factors J; constraint sets $\mathcal{E}_k$ and $\tilde{\mathcal{E}}_k$, $k \in \{1 \dots J-1\}$.

1: $\mathbf{T}_0 \leftarrow \mathbf{D}$
2: **for** $k = 1$ to $J-1$ **do**
3: Factorize the residual $\mathbf{T}_{k-1}$ into 2 factors:
$\lambda', \{\mathbf{F}_2, \mathbf{F}_1\} = \texttt{palm4MSA}(\mathbf{T}_{k-1},\ 2,\ \{\tilde{\mathcal{E}}_k, \mathcal{E}_k\},\ \dots)$
4: $\mathbf{T}_k \leftarrow \lambda' \mathbf{F}_2$ and $\mathbf{S}_k \leftarrow \mathbf{F}_1$
5: Global optimization using `palm4MSA`
6: Coefficients update:
$\mathbf{\Gamma} = \texttt{sparseCoding}(\mathbf{Y},\ \mathbf{T}_k \prod_{j=1}^{k} \mathbf{S}_j)$
7: **end for**
8: $\mathbf{S}_J \leftarrow \mathbf{T}_{J-1}$

Output: The estimated factorization: $\lambda, \{\mathbf{S}_j\}_{j=1}^{J}$, $\mathbf{\Gamma}$.

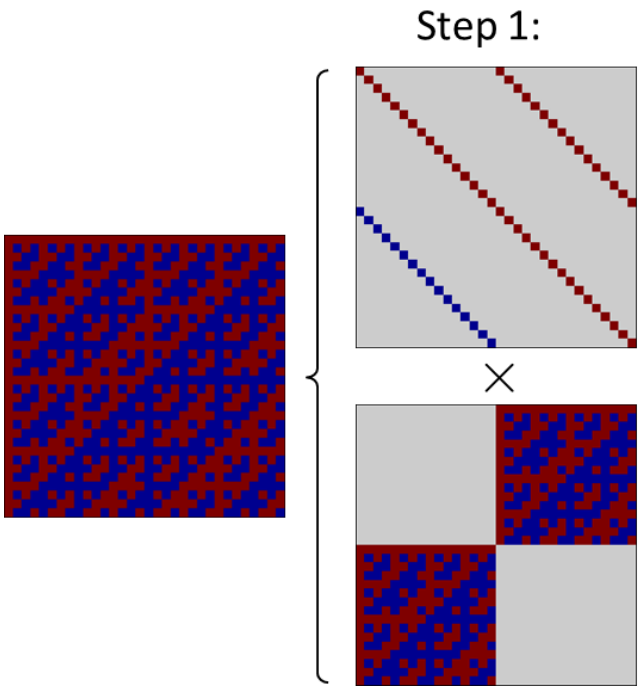
Fast transform retrieval example: Hadamard transform

The hierarchical factorization allows to retrieve the fast implementation of the Hadamard transform of size n, in running time $\mathcal{O}(n^2)$:



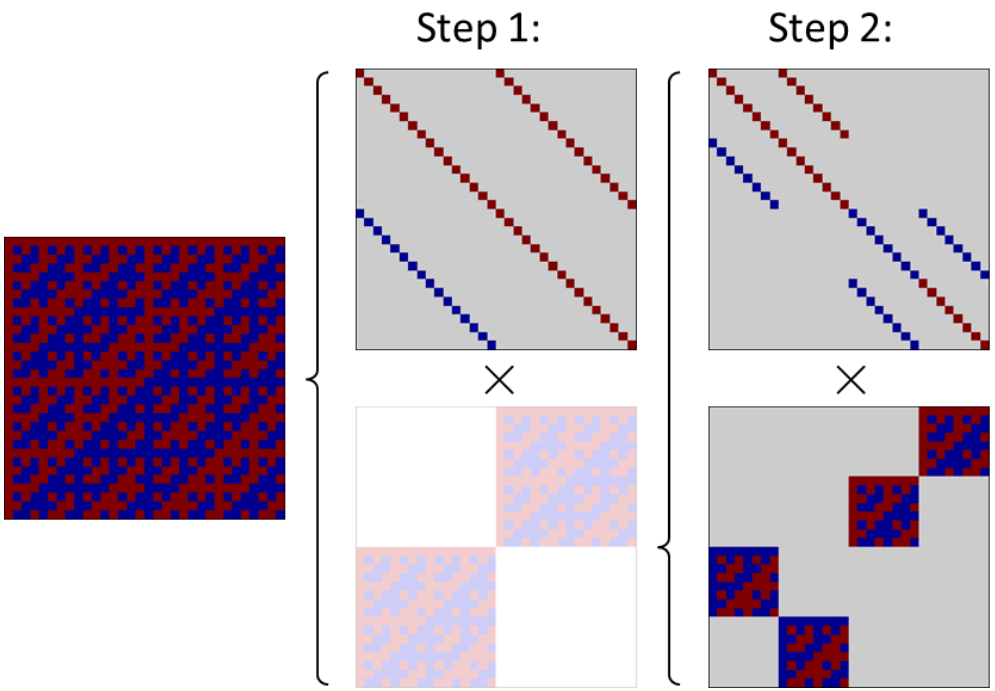
Fast transform retrieval example: Hadamard transform

The hierarchical factorization allows to retrieve the fast implementation of the Hadamard transform of size n, in running time $\mathcal{O}(n^2)$:



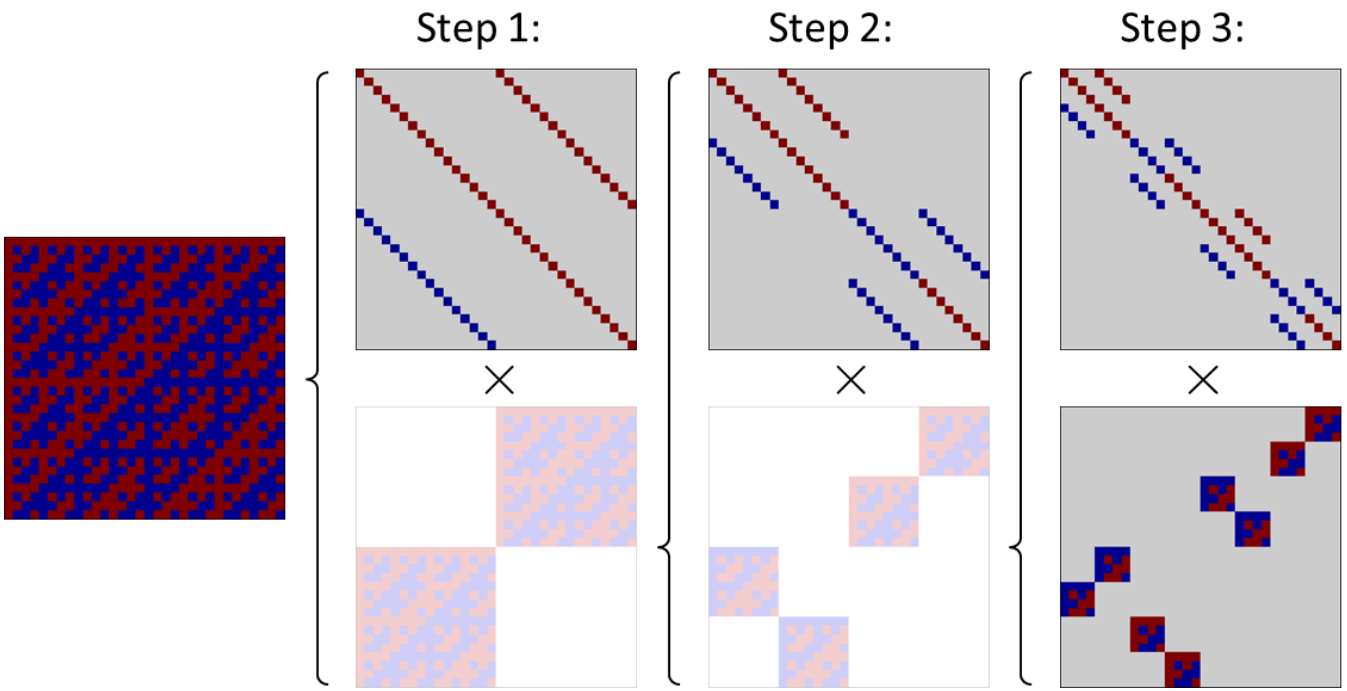
Fast transform retrieval example: Hadamard transform

The hierarchical factorization allows to retrieve the fast implementation of the Hadamard transform of size n, in running time $\mathcal{O}(n^2)$:



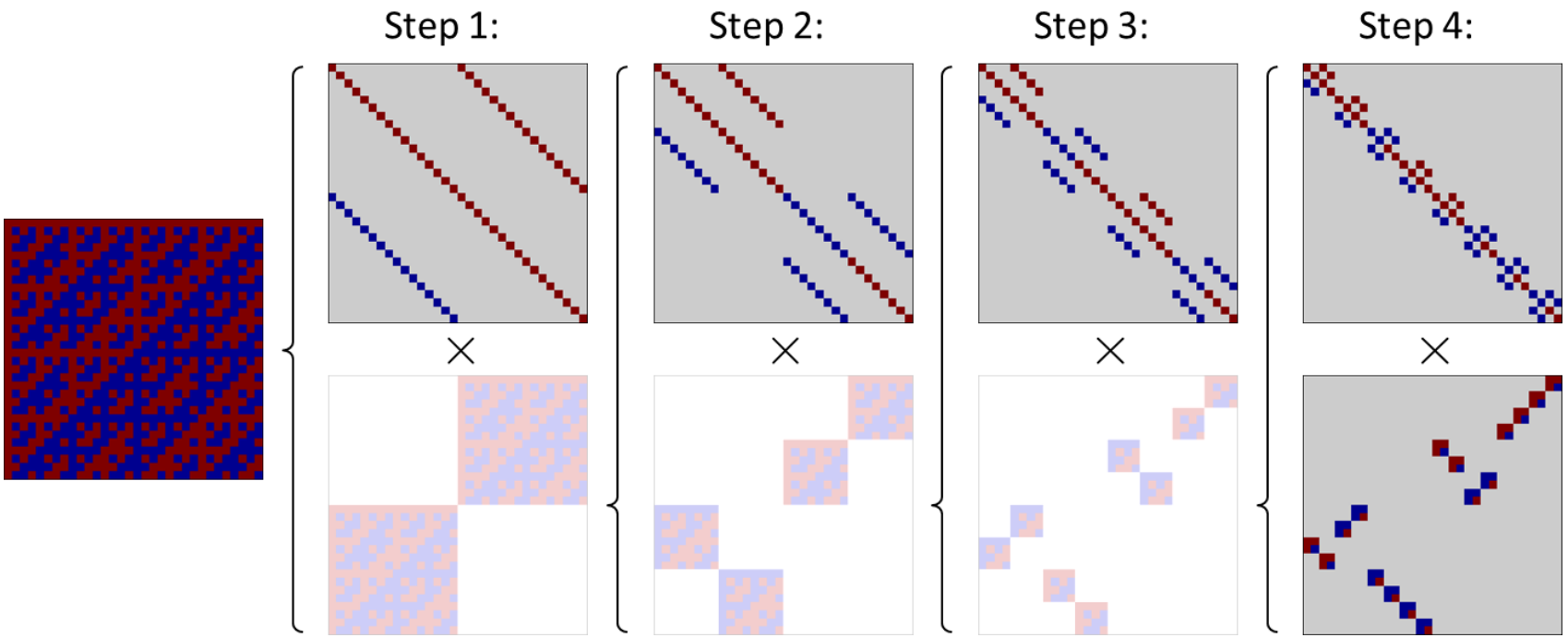
Fast transform retrieval example: Hadamard transform

The hierarchical factorization allows to retrieve the fast implementation of the Hadamard transform of size n, in running time $\mathcal{O}(n^2)$:



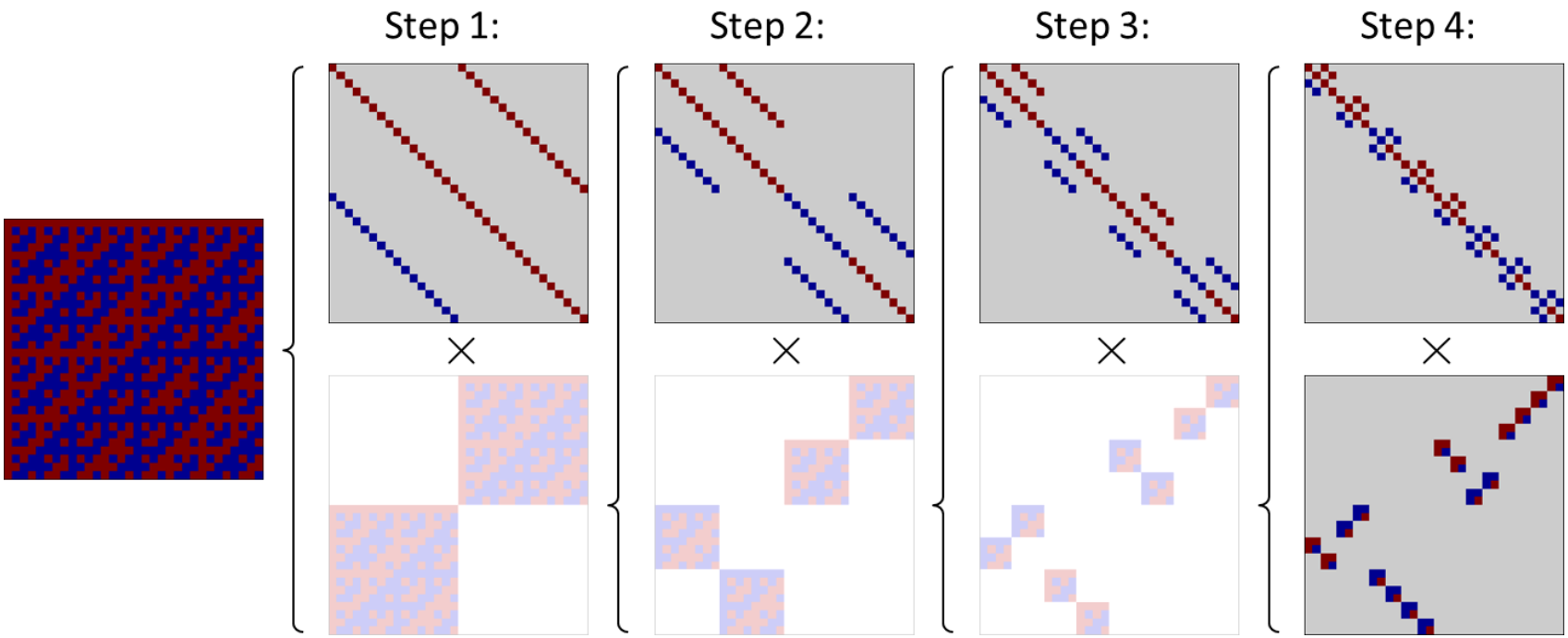
Fast transform retrieval example: Hadamard transform

The hierarchical factorization allows to retrieve the fast implementation of the Hadamard transform of size n, in running time $\mathcal{O}(n^2)$:



Fast transform retrieval example: Hadamard transform

The hierarchical factorization allows to retrieve the fast implementation of the Hadamard transform of size n, in running time $\mathcal{O}(n^2)$:



This factorization is as good as the reference.