Fast learning of fast transforms, with guarantees
Quoc-Tung Le, Léon Zheng, Elisa Riccietti, Rémi Gribonval

To cite this version:
Quoc-Tung Le, Léon Zheng, Elisa Riccietti, Rémi Gribonval. Fast learning of fast transforms, with guarantees. ICASSP 2022 - IEEE International Conference on Acoustics, Speech and Signal Processing, May 2022, Singapore, Singapore. 10.1109/ICASSP43922.2022.9747791. hal-03438881v2

HAL Id: hal-03438881
https://inria.hal.science/hal-03438881v2
Submitted on 15 Feb 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
FAST LEARNING OF FAST TRANSFORMS, WITH GUARANTEES

Quoc-Tung Le∗†, Léon Zheng∗†‡, Elisa Ricciotti†, Rémi Gribonval†

† Univ Lyon, ENS de Lyon, UCBL, CNRS, Inria, LIP, F-69342, LYON Cedex 07, France
‡ valeo.ai, Paris, France

ABSTRACT

Approximating a matrix by a product of few sparse factors whose supports possess the butterfly structure, which is common to many fast transforms, is key to learn fast transforms and speed up algorithms for inverse problems. We introduce a hierarchical approach that recursively factorizes the considered matrix into two factors. Using recent advances on the well-posedness and tractability of the two-factor fixed-support sparse matrix factorization problem, the proposed algorithm is endowed with exact recovery guarantees. Experiments show that speed and accuracy of the factorization can be jointly improved by several orders of magnitude, compared to gradient-based optimization methods.

Index Terms— Matrix factorization, Sparsity, Butterfly structure, Hierarchical factorization, NP-hardness.

1. INTRODUCTION

Interestingly, SMF with fixed support and \( J = 2 \) becomes well-posed \([9,10]\) and tractable \([8]\) under certain assumptions on the prescribed supports of the factors. These analyses have led to a factorization method for this setting that exploits non-trivially the Singular Value Decomposition (SVD) on blocks and computes an optimal factorization. This contrasts with previous works on SMF \([11,12,5,4]\), which express it as an optimization problem:

\[
\text{Minimize } \| Z - X^J \cdots X^1 \|_F^2, \tag{1}
\]

where \( \| \cdot \|_F \) is the Frobenius norm, with support / sparsity constraints on \( X^\ell (\ell = 1, \ldots, J) \). Such works typically address \((1)\) with iterative first-order methods, leading to the usual difficulties related to the non-convexity of the problem: sensitivity to initialization and high dependence on hyper-parameters such as stepsizes or stopping criteria.

Our main contribution is an algorithmic framework to address Problem \((1)\) when the fixed supports of the \( J \geq 2 \) factors have the butterfly structure \([6,5]\). This structure appears ubiquitously in fast transforms such as the Discrete Fourier Transform (DFT) or the Hadamard Transform (HT), hence the algorithm allows to approximate a given matrix by a learned fast transform. The algorithm incorporates the two-factor, SVD-based, algorithm from \([8]\) into a generalized version of a hierarchical greedy sparse factorization strategy \([5,10]\).

Based on the identifiability and recovery guarantees of \([8,9,10]\), the following exact recovery guarantee is ensured: if \( Z \) admits an exact butterfly factorization (i.e., \( Z = X^J \cdots X^1 \) and \( X^J \) is a butterfly factor), then the proposed algorithm recovers such factors up to natural scaling ambiguities. Using the factorization of the HT and DFT matrices as test cases, experiments show that this method presents important advantages over iterative first-order algorithms: reliability in finding an optimal solution, ease of use (due to the absence of hyper-parameters to be tuned), improved approximation precision and running time by more than one order of magnitude.

Section \([2]\) formally introduces butterfly supports and existing theory on fixed-support SMF for \( J = 2 \). The proposed algorithmic framework is discussed in Section \([5]\), together with its exact recovery guarantees. Section \([4]\) is dedicated to the demonstration of the superiority of the proposed algorithm to factorize linear operators with a butterfly structure.

\* The first two authors contributed equally. This project was supported in part by the AllegroAssai ANR project ANR-19-CHIA-0009.
2. BUTTERFLY STRUCTURE AND TWO-LAYER FIXED-SUPPORT SPARSE FACTORIZATION

After introducing the butterfly structure, this section gives useful results on fixed-support SMF with \( J = 2 \) factors, notably on identifiability and tractability, in preparation for the proposed hierarchical algorithmic framework.

2.1. Butterfly structure

Given a matrix \( X \), we denote \( \text{supp}(X) \) the support of \( X \), and we represent \( \text{supp}(X) \) by a binary matrix of the same size as \( X \) (1 for indices in the support and 0 otherwise). Denoting the Kronecker product by \( \otimes \) and the identity matrix of size \( N \) by \( I_N \), a square matrix of size \( N = 2^J \) is said to admit the butterfly structure if it is the product of \( J \) factors whose supports are \( S^J, \ldots, S^1 \), where \( S^j := I_{N/2^j} \otimes [1 \ 1] \otimes I_{2^{j-1}}. \) These factors are sparse, as their structure permits at most 2 nonzero coefficients per row and per column.

The butterfly structure appears in many works as it is involved in the fast transform of several well-known linear transforms, such as the DFT, the HT, but also the Discrete Cosine Transform (DCT) and the Discrete Sine Transform (DST) . For instance, the Hadamard matrix \( H_2 \) of size \( N = 2 \) admits the factorization: \( H_2 = F^J \ldots F^1 \) with \( F^j := I_{N/2^j} \otimes [1 \ -1] \otimes I_{2^{j-1}} \) for \( 1 \leq j \leq J \).

2.2. Two-layer fixed-support SMF

Approximating a matrix by another admitting the butterfly structure can be viewed as an instance of SMF with fixed support. Hence, we devote the rest of this section to the introduction of several tools developed for the fixed-support SMF problem with \( J = 2 \) factors, which serves as an intermediate step for the proposed method.

Given a subset of indices \( I \subseteq \llbracket n \rrbracket \times \llbracket m \rrbracket \), where \( \llbracket p \rrbracket := \{1, \ldots, p\} \) for \( p \in \mathbb{N} \), the set of matrices with a sparsity pattern included in \( I \) will be denoted by \( \Sigma_I := \{ M \in \mathbb{C}^{n \times m} \mid \text{supp}(M) \subseteq I \} \). In the following, \( I \) will be called by abuse of terminology a matrix support. Consider a matrix \( Z \in \mathbb{C}^{n \times m} \) and a pair of matrix supports \( S^L \subseteq \llbracket n \rrbracket \times \llbracket r \rrbracket \), \( S^R \subseteq \llbracket m \rrbracket \times \llbracket r \rrbracket \), the problem of two-layer fixed-support SMF is formulated as the following optimization problem:

Minimize \( \| Z - XY^\top \|_F \), with \( \Sigma_S := \Sigma_{S^L} \times \Sigma_{S^R}. \)

For general choices of matrix \( Z \) and pair of (left and right) supports \( S := (S^L, S^R) \), Problem 2 with arbitrary \( S^L, S^R \) is shown to be NP-hard . However, there are nontrivial conditions on the supports that ensure both tractability and identifiability of Problem 2, i.e., uniqueness of its solution up to natural scaling ambiguities.

Algorithm 1 Two-layer fixed-support matrix factorization

1: procedure FSMF(\( Z, S^L, S^R \))
2: Initialize \( X = 0, Y = 0 \).
3: Denote \( (S^l)^i_{j=1} := \varphi(S^L, S^R) \).
4: Denote \( \mathcal{P} = \{ P_j \}_{j=1}^C \) the partition of \( \llbracket r \rrbracket \) into equivalence classes of indices defined by: \( i \sim j \iff S^l_i = S^l_j \).
5: for \( P := P_i \in \mathcal{P} \) do
6: \( I = \text{rowsupp}(S^k), J = \text{colsupp}(S^k), k \in P \).
7: Use the SVD of \( Z_{I,J} \) to find \( (U, V) \in \arg \min \{ \| Z_{I,J} - UV \|_F^2 \mid U \in \mathbb{C}^{\| I \| \times P}, V \in \mathbb{C}^{\| J \| \times P} \} \).
8: Assign \( X_{I,P} = U, Y_{J,P} = V \).
9: end for
10: return \( (X, Y) \)

2.3. Identifiability

Identifiability of Problem 2 can be studied when \( Z \) admits an exact factorization \( Z = X Y^\top \) with \( (X, Y) \in \Sigma_S \). Following the general framework introduced in [9], we represent a pair \( (X, Y) \) by its \( r \)-tuple of so-called rank-one contributions

\[
\varphi(X, Y) := (X_i Y_j^\top)_{i=1}^r, \tag{3}
\]

where \( M_i \) denotes the \( i \)-th column of matrix \( M \). The support constraint \( S = (S^L, S^R) \) is then represented by the \( r \)-tuple of rank-one supports \( S = (S^l)^i_{j=1} \). The matrix supports \( S^L, S^R \) are identified to their binary matrix representation. With this lifting approach, identifiability of the two factors \( (X, Y) \) is shown to be equivalent to identifiability of the rank-one contributions \( \varphi(X, Y) \) from their sum, except in trivial degenerate cases [9].

Assume now that \( Z \) is a sum of rank-one contributions \( C^i \):

\( Z = \Sigma_{i=1}^r C^i \), with \( \text{supp}(C^i) \subseteq S^i \), rank(\( C^i \)) \leq 1, \( i \in \llbracket r \rrbracket \).

Then, when the rank-one supports \( (S^l)^i_{j=1} \) are pairwise disjoint, the rank-one matrices \( (C^1, \ldots, C^r) \) are directly identified as the submatrices of \( Z \) restricted to \( S^i \), \( i \in \llbracket r \rrbracket \). This yields a simple sufficient condition for identifiability in exact matrix factorization with fixed support [9]. Despite its simplicity, this condition will play a crucial role to prove exact recovery guarantees of hierarchical multi-layer approaches.

2.4. Tractability and polynomial algorithm

We now present a sufficient condition for tractability of Problem 2, and a polynomial algorithm to solve it even when \( Z \) does not admit an exact factorization. To introduce it, denote \( Z_{I,J} \in \mathbb{C}^{\| I \| \times \| J \|} \) as the submatrix of \( Z \) restricted to rows and columns indexed by \( I \subseteq \llbracket n \rrbracket \) and \( J \subseteq \llbracket m \rrbracket \) respectively. Denote also \( \text{colsupp}(M) \) as the subset of indices \( i \in \llbracket r \rrbracket \) such that \( M_i \) is a nonzero column and \( \text{rowsupp}(M) \) as the subset of indices \( j \in \llbracket r \rrbracket \) such that \( M^\top_j \) is a nonzero row. The algorithm to address Problem 2 is described in Algorithm 1.
Theorem 1 ([8]). Let \( \varphi(S^L, S^R) := (S^s)^\top \). If the supports \( S^s (i \in \{1, \ldots, r\}) \) are pairwise disjoint or identical, then Algorithm \( I \) yields an optimal solution to Problem \( \mathcal{P} \).

The proof of Theorem \( I \) can be found in [8]. Its main idea is that if \( S^s (i \in \{1, \ldots, r\}) \) are pairwise disjoint or identical, Problem \( \mathcal{P} \) reduces to finding low rank approximations of submatrices of the target matrix \( Z \). A generalized algorithm with guarantees under more relaxed condition is also proposed in [8]. Interestingly, these tractable conditions also ensure nice properties to Problem \( \mathcal{P} \): if a solution is locally optimal, it is also globally optimal, despite its non-convexity [8].

3. HIERARCHICAL METHOD FOR MULTI-LAYER BUTTERFLY FACTORIZATION

Going back to the SMF problem with \( J \geq 2 \) factors, we now introduce a heuristic method using Algorithm \( I \) to rapidly approximate a matrix as a product of multiple butterfly factors through a hierarchical paradigm [5] [10].

The method recursively uses Algorithm \( I \) to factorize a matrix into two factors at each intermediate level of the hierarchy. Denote the \( J \) butterfly supports of size \( 2^J \) by \( (S^1, \ldots, S^J) \). For a given intermediate factor \( H \), at a certain level of the hierarchy characterized by some integers \( p \) and \( q \), the intermediate step in the hierarchical factorization method consists of: (a) choosing an integer \( p \leq \ell < q \); (b) applying Algorithm \( I \) to solve Problem \( \mathcal{P} \) with matrix \( H \) and support constraints \( S^L := (S^q \ldots S^{q+1}) \), \( S^R := (S^q \ldots S^{p})^\top \). The obtained factors are written \( (X, Y) := \text{FSMF}(H, S^L, S^R) \).

By Theorem 1 at each intermediate factorization, the pair \( (X, Y) := \text{FSMF}(H, S^L, S^R) \) thus yields an optimal solution of Problem \( \mathcal{P} \) with matrix \( H \).

There are many ways to recursively apply Algorithm \( I \) to factorize a matrix \( Z \) into \( J \) butterfly factors. For example, at the first level of the hierarchy, one can approximate \( Z \) by \( X Y^\top (X, Y) = \text{FSMF}(Z, S^J, (S^{J-1} \ldots S^1)^\top) \). Then, \( Y \) is similarly factorized into the products of \( J - 1 \) factors with supports \( S^{J-1}, \ldots, S^1 \). Another way is to approximate at the first level the matrix \( Z \) by \( X Y^\top \) with \( (X, Y) = \text{FSMF}(Z, (S^J \ldots S^{J/2}], (S^{J/2} \ldots S^1)^\top) \), where \( \lfloor \cdot \rfloor \) denotes the floor function. After that, \( X \) (resp. \( Y \)) will be factorized into \( J - \lfloor J/2 \rfloor \) factors constrained by the supports \( S^J \), \( S^{J/2}, \ldots, S^{[J/2] + 1} \) (resp. \( S^{[J/2]}, \ldots, S^1 \)). In fact, each way of performing a hierarchical factorization corresponds to a tree. The two trees corresponding to the two schemes mentioned above are illustrated in Figure 1 and will be respectively referred to as unbalanced and balanced tree.

4. EXPERIMENTS

We now show experimentally the advantages of the proposed hierarchical factorization. All methods are implemented in Python and available in open source for reproducible research [18]. An implementation of the algorithm in C++ via Python and Matlab wrappers is also provided by the FAµST 3.25 toolbox at https://faust.inria.fr.

4.1. Outperforming iterative optimization methods

In [4], an iterative optimization technique is introduced to approximate a matrix by a product \( BP \), where \( B \) is the product of butterfly factors and \( P \) is a permutation matrix, a model that covers both the DFT and HT matrix. The approach consists of two steps: first, a joint optimization of the butterfly

![Figure 1: Two types of tree for hierarchical factorization, with \( J = 4 \) factors. A factorization into two factors is performed at each non-leaf node of the tree.](https://faust.inria.fr/)

The authors thank Hakim Hadj-Djilani for this implementation.
FACTORIZATION OF DFT MATRICES

Fig. 2: Approximation error (in Frobenius norm) of the DFT matrix of size $512$ with 9 factors: iterative method [4] vs. hierarchical method. Cumulated running times at each iteration are shown for the former, total running time for the latter.

Fig. 3: Approximation error (in Frobenius norm) of a noisy DFT matrix of size $2^J$: iterative method [4] vs. hierarchical method. The added noise matrix has i.i.d. complex centered Gaussian entries with variance $\sigma^2 = 0.005$ for both the real and imaginary part. Markers show the average running time / error. Error bars (almost invisible for the hierarchical approaches) show the extreme values over 10 different realizations of the noise matrix. As a baseline, horizontal dashed lines show the average of noise matrix norms.

For the noiseless DFT matrix, Figure 2 shows that the hierarchical approach, whether balanced or unbalanced, gives both better precision and much shorter running time, with gains of several orders of magnitude. Similar results were obtained with the HT matrix (not shown here).

We also compare the robustness of the approaches to noise. For the noisy DFT matrix, Figure 3 shows the instability of the iterative method [4], as it can give poor approximation of the noisy DFT matrix over several factorization experiments. In contrast, the proposed method is not only faster, but also finds more reliably a good approximation of the noisy DFT matrix, with an approximation error of the same order of magnitude as the norm of the noise matrix.

4.2. Balanced hierarchical factorization is faster

Finally, we compare the computational time of the balanced and unbalanced hierarchical method and show that both can learn other fast transforms than the DFT or HT, even in the noisy setting. The methods are run on a matrix $Z$ of size $2^J$ generated as a noisy version of a product $B$ of $J$ random butterfly factors. For $J \geq 11$, Figure 4 shows that an order of magnitude in running time is gained with the balanced method compared to the unbalanced one. Moreover, for the parameters used in Figure 4 to generate $Z$ and $B$, we checked that the balanced method yields an approximation error similar to the norm of the noise matrix $\|Z-B\|_F$, while the unbalanced one yields an error twice larger than $\|Z-B\|_F$.

5. CONCLUSION

This paper proposes a new approach for sparse matrix factorization with butterfly supports. Besides being free from hyper-parameters, the method has exact recovery guarantees and significantly improves over existing first-order methods to recover classical linear operators, both in terms of speed and accuracy, even in the noisy setting. It can serve as building block to “project” any matrix onto the butterfly structure, with expected applications to many problems where it is desirable to replace dense matrices with learned fast transforms, e.g., in dictionary learning or neural network training.
6. REFERENCES


