Toward Fast Transform Learning

Olivier Chabiron, Francois Malgouyres, Jean-Yves Tourneret, Nicolas Dobigeon

Toward Fast Transform Learning

O; Chabiron*[2], F. Malgouyres†[1], J.Y. Tourneret[2], and N. Dobigeon[2]

[1]Institut de Mathématiques de Toulouse, IMT-CNRS UMR 5219, Université de Toulouse, Toulouse, France

[1]Institut de Recherche en Informatique de Toulouse, IRIT- CNRS UMR 5505, ENSEEIHT, Toulouse, France

September 17, 2013

Abstract

The dictionary learning problem aims at finding a dictionary of atoms that best represents an image according to a given objective. The most usual objective consists of representing an image or a class of images sparsely. Most algorithms performing dictionary learning iteratively estimate the dictionary and a sparse representation of images using this dictionary. Dictionary learning has led to many state of the art algorithms in image processing. However, its numerical complexity restricts its use to atoms with a small support since the computations using the constructed dictionaries require too much resources to be deployed for large scale applications.

In order to alleviate these issues, this paper introduces a new strategy to learn dictionaries composed of atoms obtained as a composition of K convolutions with S-sparse kernels. The dictionary update step associated with this strategy is a non-convex optimization problem. We reformulate the problem in order to reduce the number of its irrelevant stationary points and introduce a Gauss-Seidel type algorithm, referred to as Alternative Least Square Algorithm, for its resolution. The search space of the considered optimization problem is of dimension KS, which is typically smaller than the size of the target atom and is much smaller than the size of the image. The complexity of the algorithm is linear with regard to the size of the image.

Our experiments show that we are able to approximate with a very high accuracy many atoms such as modified DCT, curvelets, sinc functions or cosines when K is large (say $K = 10$). We also argue empirically that, maybe surprisingly, the algorithm generally converges to a global minimum for large values of K and S.

1 Introduction

1.1 Problem Formulation

We consider $d \in \mathbb{N}$ and a d-dimensional signal living in a domain $\mathcal{P} \subset \mathbb{Z}^d$ (i.e., $d = 1$ for 1D signals, $d = 2$ for 2D images,...). Typically, $\mathcal{P} = \{1,\ldots,N\}^d$, where $N \in \mathbb{N}$ is the number of "pixels" along each axis. We assume that we have observed $u \in \mathbb{R}^{\mathcal{P}}$ constructed by weighted translations of a target atom $H \in \mathbb{R}^{\mathcal{P}}$ contaminated by additive noise. More precisely, we are interested in measurements defined by

$$u = \alpha * H + b, \tag{1}$$

*Olivier Chabiron is funded by the CIMI Excellence Laboratory.

†This work was performed during the Thematic Trimester on image processing of the CIMI Excellence Laboratory which was held in Toulouse, France, during the period May-June-July 2013.

where $\alpha \in \mathbb{R}^{\mathcal{P}}$ is a code of known coefficients (the knowledge of α will be motivated more carefully later), $b \in \mathbb{R}^{\mathcal{P}}$ is a random noise and $*$ stands for the circular discrete convolution[1] in dimension d. The typical frameworks we have in mind include situations where α is a sparse code, and where α contains coefficients that have been estimated by strategies such as those described in Section 1.2. Note that we do not need any constraint about the code vector α even though the performance of the proposed algorithm will of course depend on the conditioning of the convolution with respect to the value of α.

The problem addressed in this paper consists of estimating the unknown atom H and to express it as a composition of convolutions with sparse kernels. As an example, we mention the decomposition of a curvelet atom H that will receive a specific attention in our experiments (see Section 4.3.1). More precisely, we consider an integer $K \geq 2$ and K convolutions with sparse kernels $(h^k)_{1\leq k\leq K} \in (\mathbb{R}^{\mathcal{P}})^K$. We assume that all these kernels have less than a fixed number S of non-zero elements, i.e., that they are at most S-sparse. Furthermore, we assume that the locations of the non-zero elements in $\mathcal{P}$ are known or pre-set. In order to manipulate the supports of these kernels, we define, for all $k \in \{1,\ldots,K\}$, an injective mapping

$$S^k : \{1,\ldots,S\} \longrightarrow \mathcal{P},$$

denoted $S^k \in \mathcal{P}^S$, indicating which elements of h^k can differ from zero. More precisely, by denoting as $\text{supp}\left(h^k\right)$ the support of the kth kernel (i.e., the locations of the non zero elements of h^k), we assume the following constraint

$$\text{supp}\left(h^k\right) \subset \text{rg}\left(S^k\right) \qquad \forall k \in \{1,\ldots,K\} \tag{2}$$

where $\text{rg}\left(S^k\right) = \{S^k(1),\ldots,S^k(S)\}$ contains all the possible locations of the non-zero elements of h^k. Examples of simple mappings include $S^k(s) = ks$, $\forall s \in \{1,\ldots,S\}$, for 1D signals. A similar support is displayed in Fig. 1 for 2D images. In addition to the support constraint (2), the convolution of the K kernels $\mathbf{h} = (h^k)_{1\leq k\leq K} \in (\mathbb{R}^{\mathcal{P}})^K$, should approximate the target atom H, i.e.,

$$h^1 * \cdots * h^K \approx H.$$

Therefore, we propose to solve the following optimization problem

$$(P_0): \qquad \left\{ \begin{array}{l} \text{argmin}_{\mathbf{h}\in(\mathbb{R}^{\mathcal{P}})^K} \|\alpha * h^1 * \cdots * h^K - u\|_2^2, \\ \mid \ \text{supp}\left(h^k\right) \subset \text{rg}\left(S^k\right) \qquad ,\forall k \in \{1,\ldots,K\} \end{array} \right.$$

where $\|.\|_2$ stands for the usual Euclidean[2] norm in $\mathbb{R}^{\mathcal{P}}$.

The problem (P_0) is non convex. Thus, depending on the values of $K \geq 2$, $(S^k)_{1\leq k\leq K} \in (\mathcal{P}^S)^K$, $\alpha \in \mathbb{R}^{\mathcal{P}}$ and $u \in \mathbb{R}^{\mathcal{P}}$, it might be difficult or impossible to find a good approximation of a global minimizer of (P_0). The main objective of this paper is to study if such a problem bends itself to global optimization. Another important objective is to assess empirically if the computed compositions of convolutions provide good approximations of interesting atoms. The current paper contains the description of an algorithm for solving (P_0) and the analysis of its performance. However, before describing the proposed algorithm, we mention some links between the optimization problem (P_0) and some known issues in sparse representation.

1.2 Motivations

Our primary motivation for considering the observation model (1) comes from dictionary learning (DL). DL was pioneered by [10, 16] and has received a growing attention during the last ten years. It can be viewed as a way of representing data using a sparse representation. We invite the reader to consult [4] for more details about sparse representations and DL. Given a set of L images[3] $(u^l)_{1\leq l\leq L} \in (\mathbb{R}^{\mathcal{P}})^L$, the

[1] All the elements of $\mathbb{R}^{\mathcal{P}}$ are extended over $\mathbb{Z}^d$ by periodization.

[2] $\mathbb{R}^{\mathcal{P}}$ and $\mathbb{R}^S$ are endowed with the usual scalar product denoted $\langle .,. \rangle$ and the usual Euclidean norm denoted $\|\cdot\|_2$. We use the same notation whatever the vector space. We expect that the notation will not be ambiguous, once in context.

[3] Usually, DL is applied to small images such as patches extracted from large images.

archetype of the DL strategy is to look for a dictionary as the solution of the following optimization problem

$$\operatorname{argmin}_{\mathbb{H},(\alpha^l)_{1\leq l\leq L}} \quad \sum_{l=1}^{L} \|\mathbb{H}\alpha^l - u^l\|_2^2 + \lambda\|\alpha^l\|_s,$$

where $\mathbb{H}$ is a matrix whose columns are the atoms of the dictionary, $\lambda \geq 0$ is a parameter and $\|.\|_s$ is a sparsity-inducing norm such as the counting function (or ℓ_0 pseudo-norm) or the usual ℓ_1 norm. The DL optimization problem is sometimes formulated by imposing a constraint on $\|\alpha^l\|_s$. The resulting non-convex problem can be solved (or approximatively solved) by many methods including the "method of optimal direction" (MOD) [5] and, in a different manner, by K-SVD [1]. To better reflect the distribution of images, it can also be useful to increase the number of images and to use an online strategy [13]. Finally, note that an alternative model has been presented for task driven DL in [12]. Algorithmically, all these approaches rely on alternatively updating the codes $(\alpha^l)_{1\leq l\leq L}$ and the dictionary $\mathbb{H}$.

The problem considered in the current paper mimics an update step of the dictionary. In this context, α is fixed and the target atom H is a column of the dictionary $\mathbb{H}$. The dictionary $\mathbb{H}$ is made of translations of the target atom H. Notice that there is a straightforward way to apply the proposed update, iteratively with the correct entries, in order to learn several atoms. The main novelty of the proposed approach is to impose the learned atoms to be a composition of convolutions with sparse kernels. The interest for such a constraint is that it provides numerically effective dictionaries and permits to consider larger atoms. Indeed, the reconstruction operator

$$\begin{array}{ccc} \mathbb{R}^{\mathcal{P}} & \longrightarrow & \mathbb{R}^{\mathcal{P}} \\ \alpha & \longmapsto & \alpha * h^1 * \ldots * h^K \end{array}$$

and its adjoint can be computed by K convolutions with kernels of size S. As a consequence, the computation of the reconstruction operator and its adjoint have a computational complexity of $O(KS\#\mathcal{P})$, where $\#\mathcal{P}$ denotes the cardinality of the set $\mathcal{P}$. Depending on the support mappings $(S^k)_{1\leq k\leq K}$, this complexity can be much smaller than a convolution with a kernel filling in the "reachable support"

$$\mathcal{S} = \left\{ p \in \mathcal{P}, \exists (p_k)_{1\leq k\leq K} \in \operatorname{rg}(S_1) \times \cdots \times \operatorname{rg}\left(S^k\right), \sum_{k=1}^{K} p_k = p \right\}. \tag{3}$$

In the latter case, the computational complexity is indeed equal to $O(\#\mathcal{S}\#\mathcal{P})$ or $O(\#\mathcal{P}\log(\#\mathcal{P}))$ if the convolutions are computed using a Fast Fourier Transform (FFT). Moreover, when several atoms are considered, the convolutions with sparse kernels can be arranged according to a tree structure to save even more computing resources. The typical example of an existing dictionary having a similar structure is the dictionary made of undecimated wavelets [22]. Note that the ~~fast~~ FFT algorithm of Cooley-Tukey has also a related structure [2] (although it involves an up-sampling step that has not been included in the proposed model).

To conclude with this subject, having a numerically effective scheme for using a dictionary is crucial since the computational complexity of most algorithms favoring sparsity is proportional to the computational complexity of the matrix-vector multiplications involving $\mathbb{H}$ and its transpose. In particular, for the DL algorithms alternating a sparse coding step and a dictionary update step, the sparse coding steps require less computation resources. These resources are therefore available for the dictionary update.

1.3 Related Works

Before going ahead, it is interesting to describe the structures of the dictionaries that have been considered in DL. Structured and parametric dictionaries have recently been considered with increasing interest. Interested readers can find a concise bibliographical note on that subject in [19]. In particular, the structures studied so far include unions of orthobases [9], translation invariant dictionaries [11], dictionaries composed of patches with multiple sizes [14], dictionaries divided in ordered pieces (that are learnt from the residuals obtained when representing the sample patches with the previously computed atoms of the dictionary) [24], structures induced by structured codes [7, 8]. Other interesting dictionaries are characterized by several layers. Such dictionaries can be constructed as the composition of a fixed transform and learned dictionaries [20, 17]. They can also be dictionaries made of two layers based on a sparsifying transform and a sampling matrix (both layers can be learnt by the algorithm investigated in [3]). To the best of our knowledge, there only exists a few attempts for building dictionaries involving an arbitrary number of layers. In a slightly different context, dictionaries structured by Kronecker products have been proposed in [23]. Interestingly, despite the non-convexity of the corresponding energy, it is possible to find some of its global minima [25]. Dictionaries structured by wavelet-like trees (similar to one we are targeting in this paper) using a dictionary update based on a gradient descent have been studied in [21].

When compared to these dictionary structures, the structure of the proposed dictionary aims at obtaining a numerically efficient translation invariant dictionary, whose elementary atoms H can have large supports. Moreover, the update of the proposed structured dictionary reduces to a global optimization problem. Surprisingly, the proposed algorithm provides interesting solutions for relatively large values of the number of layers K, e.g., $K = 10$ seems very reasonable.

1.4 Paper Organization

The paper is organized as follows. Section 1 formulates the proposed dictionary update and provides motivations with references to previous works. A more practical problem formulation is introduced in Section 2. Section 3 presents an algorithm for approximating a dictionary atom as a composition of convolutions, in order to build a fast transform. The algorithm is based on an alternating least squares strategy whose steps are detailed carefully. Simulation results illustrating the performance of the proposed algorithm and its convergence properties are provided in Sections 4 and 5. Conclusions and future work are reported in Section 6.

2 Reformulating (P_0)

The problem (P_0) is not very tractable because it has many stationary points. Denote as $\mathbf{h} = (h^k)_{1\leq k\leq K} \in (\mathbb{R}^{\mathcal{P}})^K$ the sequence of kernels and as E the objective function of (P_0)

$$E(\mathbf{h}) = \|\alpha * h^1 * \cdots * h^K - u\|_2^2.$$

For any $k \in \{1,\ldots,K\}$, the gradient of the energy function $\frac{\partial E}{\partial h^k}$ can be calculated easily, leading to

$$\frac{\partial E}{\partial h^k}(\mathbf{h}) = \tilde{H}^k * (\alpha * h^1 * \cdots * h^K - u), \tag{4}$$

where

$$H^k = \alpha * h^1 * \cdots * h^{k-1} * h^{k+1} * \cdots * h^K, \tag{5}$$

and where the $\tilde{.}$ operator is defined for any $h \in \mathbb{R}^{\mathcal{P}}$ as

$$\tilde{h}_p = h_{-p}, \quad \forall p \in \mathcal{P}. \tag{6}$$

Note that the notation H^k has been used instead of $H^k(\mathbf{h})$ to improve readability.

As soon as $h^{k_1} = h^{k_2} = 0$ for two distinct values of k_1 and $k_2 \in \{1,\ldots,K\}$, we have $H^k = 0$, for all $k \in \{1,\ldots,K\}$, and thus

$$\frac{\partial E}{\partial h^k}(\mathbf{h}) = 0 \qquad \forall k \in \{1,\ldots K\}.$$

As a consequence, nothing prevents a minimization algorithm solving (P_0) to get stuck at one of these stationary points, although it is usually not a global minimizer of (P_0).

Furthermore, $\forall \mathbf{h} \in (\mathbb{R}^{\mathcal{P}})^K$ and $\forall (\mu_k)_{1\leq k\leq K} \in \mathbb{R}^K$ such that $\prod_{k=1}^K \mu_k = 1$, we have

$$E\left[(\mu_k h^k)_{1\leq k\leq K}\right] = E(\mathbf{h}),$$

while, for any $k \in \{1, \dots, K\}$,

$$\frac{\partial E}{\partial h^k} \left[(\mu_k h^k)_{1 \leq k \leq K} \right] = \frac{1}{\mu_k} \frac{\partial E}{\partial h^k} (\mathbf{h}).$$

This relation results in an unbalanced situation where the gradient depends on quantities which are irrelevant with regard to the value of the objective function.

To address the two issues mentioned above and reduce the number of irrelevant stationary points, we propose to include an additional constraint for the norms of the kernels $h^k \in \mathbb{R}^{\mathcal{P}}$, $\forall k \in \{1, \dots, K\}$. More precisely, we consider a norm-to-one constraint $\|h^k\|_2 = 1$, $\forall k \in \{1, \dots, K\}$ and introduce an additional signed weight $\lambda \in \mathbb{R}$ to scale the result according to the target atom. To simplify notations, we write

$$\mathcal{D} = \left\{ h = (h^k)_{1 \leq k \leq K} \in (\mathbb{R}^{\mathcal{P}})^K | \forall k \in \{1, \dots, K\}, \|h^k\|_2 = 1 \text{ and } \operatorname{supp}\left(h^k\right) \subset \operatorname{rg}\left(S^k\right) \right\}$$

and define the following optimization problem

$$(P_1): \qquad \operatorname{argmin}_{\lambda \in \mathbb{R}, \mathbf{h} \in \mathcal{D}} \|\lambda \alpha * h^1 * \cdots * h^K - u\|_2^2,$$

It is straightforward to construct a solution of (P_0) from a solution of (P_1)[4]. Indeed, if λ and $(h^k)_{1 \leq k \leq K}$ are solutions of (P_1), the kernels $\mathbf{g} = (g^k)_{1 \leq k \leq K} \in (\mathbb{R}^{\mathcal{P}})^K$ defined by

$$g^1 = \lambda\, h^1 \text{ and } g^k = h^k, \quad \forall k \in \{2, \dots, K\}, \tag{7}$$

clearly satisfy the constraints of (P_0). Moreover, for any $(f^k)_{1 \leq k \leq K} \in (\mathbb{R}^{\mathcal{P}})^K$ satisfying these constraints[5], we have

$$\begin{aligned} \|\alpha * g^1 * \dots * g^K - u\|_2^2 &= \|\lambda \alpha * h^1 * \dots * h^K - u\|_2^2, \\ &\leq \left\| \left(\prod_{k=1}^K \|f^k\|_2 \right) \alpha * \frac{f^1}{\|f^1\|_2} * \dots * \frac{f^K}{\|f^K\|_2} - u \right\|_2^2, \\ &\leq \|\alpha * f^1 * \dots * f^K - u\|_2^2. \end{aligned}$$

As a consequence, the kernels $(g^k)_{1 \leq k \leq K}$ defined by (7) form a solution of (P_0).

Let us now analyze the properties of the optimization problem (P_1). The first property takes the form of the following proposition.

Proposition 2.1 (Existence of a solution). *For any $(u, \alpha, (S^k)_{1 \leq k \leq K}) \in \left(\mathbb{R}^{\mathcal{P}} \times \mathbb{R}^{\mathcal{P}} \times (\mathcal{P}^S)^K\right)$, if*

$$\forall \mathbf{h} \in \mathcal{D}, \qquad \alpha * h^1 * \dots * h^K \neq 0, \tag{8}$$

then the problem (P_1) has a minimizer.

Proof. First notice that $\mathcal{D}$ is a compact set. Moreover, when (8) holds, the objective function of (P_1) is coercive in λ. Thus, for any threshold μ, it is possible to build a compact set such that the objective function evaluated at any $(\lambda, \mathbf{h})$ outside this compact set is larger than μ. As a consequence, we can extract a converging subsequence from any minimizing sequence. Since the objective function of (P_1) is continuous and its domain is closed, the limit of this subsequence is a minimizer of (P_1). □

Note that there might be refined alternatives to the condition (8). However, the investigation of the tightest condition for the existence of a minimizer of (P_1) is clearly not the subject of this paper. Concerning the existence of a solution~~, the main properties of (P_1) are that the kernels $\mathbf{h} \in (\mathbb{R}^{\mathcal{P}})^K$ are constrained to live in a compact set. Also, the objective function of problem (P_1) is coercive with respect to λ provided (8) is satisfied. However,~~ notice that the objective function of (P_1) is not necessarily coercive, e.g., it is not coercive if there exists $\mathbf{h} \in (\mathbb{R}^{\mathcal{P}})^K$ such that $\alpha * h^1 * \dots * h^K = 0$.

[4] Note that, although this is not necessary to our construction, solutions of (P_1) can be computed from solutions of (P_0) similarly.

[5] We further assume that $\|f^k\|_2 \neq 0$, for all $k \in \{1, \dots, K\}$, since the inequality is otherwise trivial.

In this situation, a minimizing sequence might be such that $\lambda\alpha * h^1 * \ldots * h^K$ and $(h^k)_{1\leq k\leq K}$ have accumulation points but $\alpha * h^1 * \ldots * h^K$ goes to 0 and λ goes to infinity. Note finally that we typically expect the condition (8) to hold as soon as the supports $(S^k)_{1\leq k\leq K} \in (\mathcal{P}^S)^K$ and $\text{supp}\,(\alpha)$ are sufficiently localized. In our experiments, we have never encountered a situation where $\alpha * h^1 * \ldots * h^K$ equals zero.

Let us now give structural properties of (P_1). The objective function of (P_1) is a polynomial of degree $2K$. It is infinitely differentiable and non-negative. The objective function of (P_1) is non-convex. However, for any $k \in \{1,\ldots,K\}$, the objective function of (P_1) is convex and quadratic with respect to h^k. Finally, $\mathcal{D}$ is a smooth but non convex set. Computing an orthogonal projection onto $\mathcal{D}$ is easy.

3 The alternating least squares (ALS) algorithm

3.1 Principle of the algorithm

The objective function in (P_1) being non-convex, there is no guaranty to find a global minimum of (P_1). However, it makes sense to build a method finding a stationary point of (P_1). For that purpose, we propose to alternate minimizations with respect to the kernels h^k, $\forall k \in \{1,\ldots,K\}$. More precisely, for any $k \in \{1,\ldots,K\}$, we propose to solve (alternatively) the following least square (LS) problem

$$(P_k): \quad \begin{cases} \text{argmin}_{\lambda\in\mathbb{R}, h\in\mathbb{R}^{\mathcal{P}}} \|\lambda\alpha * h^1 * \cdots * h^{k-1} * h * h^{k+1} * \ldots * h^K - u\|_2^2, \\ \mid \ \text{supp}\,(h) \subset \text{rg}\left(S^k\right) \text{ and } \|h\|_2 = 1 \end{cases}$$

where the kernels $(h_p^{k'})_{p\in\mathcal{P}}$ are fixed $\forall k' \neq k$. The resulting ALS algorithm is described in Algo. 1.

Algorithm 1: Overview of the ALS algorithm

```
Input:
$u$: target measurements;
$\alpha$: known coefficients;
$(S^k)_{1\leq k\leq K}$: supports of the kernels $(h^k)_{1\leq k\leq K}$.
Output:
$\lambda$ and kernels $(h^k)_{1\leq k\leq K}$ such that $\lambda h^1 * \ldots * h^K \approx H$.

begin
    Initialize the kernels $(h^k)_{1\leq k\leq K}$;
    while not converged do
        for k = 1 ,..., K do
            Update $\lambda$ and $h^k$ with a minimizer of $(P_k)$.
end
```

3.2 Resolution of (P_k)

Before studying the existence of a minimizer of (P_k), let us rewrite the problem (P_k) in a simpler form. Since any convolution can be described as a matrix-vector product, the problem (P_k) is equivalent to

$$(P_k): \quad \begin{cases} \text{argmin}_{\lambda\in\mathbb{R}, h\in\mathbb{R}^S} \|\lambda C_k h - u\|_2^2 \\ \mid \ \|h\|_2 = 1 \end{cases}$$

where the $\#\mathcal{P} \times S$ matrix C_k will be defined later and where we consider that u has been vectorized. In order to solve this problem, we define

$$(P_k'): \qquad \text{argmin}_{h\in\mathbb{R}^S} \|C_k h - u\|_2^2.$$

It is clear that (P'_k) has a minimizer $h^* \in \mathbb{R}^S$. Moreover, by computing a stationary point of the quadratic objective function of the problem (P'_k), we obtain:

$$h^* = (C_k^T C_k)^{\dagger} C_k^T u, \tag{9}$$

where $(C_k^T C_k)^{\dagger}$ is the pseudo-inverse of $C_k^T C_k$. Setting

$$\lambda = \|h^*\|_2 \qquad \text{and} \qquad h^k = \begin{cases} \frac{h^*}{\|h^*\|_2} & \text{, if } \|h^*\|_2 \neq 0, \\ \frac{1}{\sqrt{S}} \mathbb{1}_{\{1,\dots,S\}} & \text{, otherwise,} \end{cases} \tag{10}$$

where $\mathbb{1}$ is the indicator function, one can show that any $(\mu, g) \in \mathbb{R} \times \mathbb{R}^S$ satisfying the constraints of (P_k) is such that:

$$\begin{array}{rcl} \|\lambda C_k h^k - u\|_2^2 & = & \|C_k h^* - u\|_2^2, \\ & \leq & \|C_k(\mu g) - u\|_2^2 = \|\mu C_k g - u\|_2^2. \end{array}$$

As a consequence, (P_k) has a minimizer and it is provided by (10):

$$\left(\lambda, h^k\right) \in \begin{array}{l} \operatorname{argmin}_{\lambda \in \mathbb{R}, h \in \mathbb{R}^S} \|\lambda C_k h - u\|_2^2 \\ \mid \ \|h\|_2 = 1 \end{array}$$

Altogether, we obtain the update rule described by (9) and (10). In order to apply these formulas, the main computational difficulties are to compute $C_k^T u$, $C_k^T C_k$ and the pseudo inverse of $C_k^T C_k$. These computations are the subject of the next section.

3.3 Computing $C_k^T u$ and $C_k^T C_k$

Considering Dirac delta functions for $h \in \mathbb{R}^S$ and the linearity of C_k, we obtain for any $h \in \mathbb{R}^S$

$$(C_k h)_p = \sum_{s=1}^{S} H^k_{p - S^k(s)} h_s, \qquad \forall p \in \mathcal{P},$$

where H^k is defined in (5). In other words, each column of C_k is a vectorization of $(H^k_{p-S^k(s)})_{p \in \mathcal{P}}$. For any $p' \in \mathcal{P}$, denote as $\tau_{p'}$ the translation operator such that $(\tau_{p'} v)_p = v_{p-p'}$, $\forall (v, p) \in \mathbb{R}^{\mathcal{P}} \times \mathcal{P}$. Using this notation, the s^{th} column of C_k is a vectorization of $\tau_{S^k(s)} H^k$. Therefore, the s^{th} line of C_k^T is the transpose of a vectorization of $\tau_{S^k(s)} H^k$. We finally have

$$(C_k^T v)_s = \langle \tau_{S^k(s)} H^k, v \rangle, \qquad \forall v \in \mathbb{R}^{\mathcal{P}}. \tag{11}$$

Note that the computational complexity for computing H^k is $O((K-1)S\#\mathcal{P})$. Once H^k has been computed, the cost for computing $(C_k^T u)_s$ is $O(\#\mathcal{P})$, $\forall s \in \{1, \dots, S\}$, and therefore the cost for computing $C_k^T u$ is $O(S\#\mathcal{P})$. Altogether, we obtain a complexity $O(KS\#\mathcal{P})$.

We can immediately deduce the form of $C_k^T C_k$. Indeed, each of its column is obtained by applying (11) in which we replace v by the column vector $\tau_{S^k(s')} H^k$, for some $s' \in \{1, \dots, S\}$. Therefore the coefficient of $C_k^T C_k$ at the location $(s, s') \in \{1, \dots, S\}^2$ is

$$(C_k^T C_k)_{s,s'} = \langle \tau_{S^k(s)} H^k, \tau_{S^k(s')} H^k \rangle. \tag{12}$$

This Gram matrix is symmetric, positive semidefinite and of size $S \times S$. Once H^k has been computed, the computational complexity for computing $C_k^T C_k$ is $O(S^2\#\mathcal{P})$. The computation of its pseudo-inverse is a well studied problem and is a step of the algorithm that can be optimized. An off-the-shelf implementation using a singular value decomposition (SVD) typically requires $O(S^3)$ operations.

Algorithm 2 summarizes all the steps required for the proposed ALS algorithm. The overall computational complexity is typically $O((K+S)KS\#\mathcal{P})$ per iteration[6]. It can be reasonably applied in situations where $KS(K+S)$ is not to large. The most demanding case considered in the experiments described in this paper corresponds to $KS^2 = 6250$ (corresponding to $K = 10$ and $S = 25$). In order to choose the number of iterations in the "while" loop, we have used the relative difference between the values of the objective function of (P_k) for two consecutive iterations. When this difference is lower than 10^{-3}, we consider that we have reached a stationary point, and the algorithm stops.

Algorithm 2: Detailed ALS algorithm

```
Input:
$u$: target measurements;
$\alpha$: known coefficients;
$(S^k)_{1\leq k\leq K}$: supports of the kernels $(h^k)_{1\leq k\leq K}$.
Output:
$(h^k)_{1\leq k\leq K}$: convolution kernels such that $h^1 * \ldots * h^K \approx H$.

begin
    Initialize the kernels $((h^k_p)_{p\in\mathcal{P}})_{1\leq k\leq K}$;
    while not converged do
        for k = 1 ,..., K do
            Compute $H^k$ according to (5)                                  $O((K-1)S\#\mathcal{P})$
            Compute $C_k^T C_k$ and $C_k^T u$ according to (12) and (11);   $O((S+1)S\#\mathcal{P})$
            Compute $h^*$ according to (9);                                 $O(S^3)$
            Update $h^k$ and $\lambda$ according to (10);                   $O(S)$
end
```

3.4 Convergence of the algorithm

Proposition 3.1 (Convergence of Algorithm 2). *For any* $(u, \alpha, (S^k)_{1\leq k\leq K}) \in \left(\mathbb{R}^{\mathcal{P}} \times \mathbb{R}^{\mathcal{P}} \times (\mathcal{P}^S)^K\right)$, *if*

$$\forall \mathbf{h} \in \mathcal{D}, \qquad \alpha * h^1 * \ldots * h^K \neq 0, \tag{13}$$

the following statements hold:

1. *The sequence generated by Algorithm 2 is bounded and has limit points. The value of the objective function is the same for all the limit points of the sequence generated by the algorithm.*
2. *For any limit point* $(\lambda^*, \mathbf{h}^*) \in \mathbb{R} \times (\mathbb{R}^{\mathcal{P}})^K$, *if the matrix* C_k *generated using* $\mathbf{h}^*$ *is full column rank* $\forall k \in \{1, \ldots, K\}$, *then the limit point* $(\lambda^*, \mathbf{h}^*)$ *is a stationary point for the problem* (P_1).

Proof. The first item of proposition 3.1 can be obtained directly since 1) the sequence of kernels generated by the algorithm belong $\mathcal{D}$ and $\mathcal{D}$ is compact, 2) the objective function of (P_1) is coercive with respect to λ when (13) holds, and 3) the objective function decreases during the iterative process and is continuous. Concerning the second item of proposition 3.1, since every element of the sequence generated by Algorithm 2 belong to $\mathbb{R} \times \mathcal{D}$, and since $\mathbb{R} \times \mathcal{D}$ is a closed set, any limit point $(\lambda^*, \mathbf{h}^*)$ also belongs to $\mathbb{R} \times \mathcal{D}$. We denote by F the objective function of (P_1), by $(\lambda^n, \mathbf{h}^n)_{n\in\mathbb{N}}$ the sequence generated by Algorithm 2 and by T the mapping obtained when applying the "for" loop of Algorithm 2. We have, $\forall n \in \mathbb{N}$

$$(\lambda^{n+1}, \mathbf{h}^{n+1}) = T(\lambda^n, \mathbf{h}^n).$$

[6] In the practical situations we are interested in, $\#\mathcal{P} \gg S$ and S^3 can be neglected when compared to $(K+S)S\#\mathcal{P}$.

We denote by $(\lambda^o, \mathbf{h}^o)_{o\in\mathbb{N}}$ a subsequence of $(\lambda^n, \mathbf{h}^n)_{n\in\mathbb{N}}$ converging to a limit point $(\lambda^*, \mathbf{h}^*)$. The following statements are trivially true:

$$\begin{aligned}\lim_{o\to\infty} F(\lambda^o, \mathbf{h}^o) &= F(\lambda^*, \mathbf{h}^*) \\ &= \lim_{o\to\infty} F\left(T(\lambda^o, \mathbf{h}^o)\right)\end{aligned}$$

However, if all the matrices C_k generated using $\mathbf{h}^*$ are full column rank, when k varies inside $\{1,\ldots,K\}$, then T is a continuous mapping in the vicinity of $\mathbf{h}^*$. Therefore, the above equalities garantee that

$$F(\lambda^*, \mathbf{h}^*) = F\left(T(\lambda^*, \mathbf{h}^*)\right).$$

As a consequence, for every $k \in \{1,\ldots,K\}$, $(\lambda^*, \mathbf{h}^{*,k})$ is a minimizer and therefore a stationary point of (P_k). We can then deduce that $(\lambda^*, \mathbf{h}^*)$ is a stationary point of (P_1). □

3.5 Initialization of the algorithm and restart

First, it is interesting to note that the ALS algorithm does not need any initialization for λ. Moreover, the initial values $(h^k)_{1\leq k\leq K}$ of the sequence must satisfy the constraints and therefore belong to $\mathcal{D}$. When the problem (P_1) has a global minimizer, we denote by $\mathbb{I} \subset \mathcal{D}$ the non-empty convergence set such that when the ALS algorithm is initialized inside $\mathbb{I}$, it converges to a global minimizer. Surprisingly, after running intensively the ALS algorithm, it appears that in many situations $\mathbb{I}$ is actually large. In order to illustrate this aspect, we have chosen a simple initialization. It consists of initializing our algorithm by drawing a random variable uniformly distributed in $\mathcal{D}$. This is easily achieved [15] by using[7]

$$h^k = \frac{h}{\|h\|_2}, \qquad \text{with} \qquad h \sim \mathcal{N}_S(0, Id),$$

where $\mathcal{N}_S(0, Id)$ is the centered normal distribution in $\mathbb{R}^S$. Our experiments will show that $\mathbb{P}(\mathbf{h} \notin \mathbb{I})$ is often far from 1 when $\mathbf{h}$ is uniformly distributed in D.

Moreover, an advantage of this random initialization is that, in order to explore $\mathcal{D}$, we can use a "restart" strategy. More precisely, we propose to run the ALS algorithm R times, for $R \in \mathbb{N}$, and to return the result for which the objective function is the smallest. The probability that such a strategy fails to provide a global minimizer is equal to the probability that none of the R independent initializations belong to $\mathbb{I}$, i.e.,

$$\mathbb{P}(\text{not global}) = [\mathbb{P}(\mathbf{h} \notin \mathbb{I})]^R$$

which decays rapidly to 0, when $\mathbb{P}(\mathbf{h} \in \mathbb{I})$ is not negligible. For instance, to guarantee

$$\mathbb{P}(\text{not global}) \leq \varepsilon,$$

for $\varepsilon > 0$, we must take

$$R \geq R_\varepsilon = \frac{\log(\varepsilon)}{\log\left(\mathbb{P}(\mathbf{h} \notin \mathbb{I})\right)}. \tag{14}$$

Note that the number of restarts does not increase much when ε decreases. However, when $\mathbb{P}(\mathbf{h} \in \mathbb{I})$ is small (or negligible) we have

$$R \geq \frac{\log(\varepsilon)}{\log\left[1 - \mathbb{P}(\mathbf{h} \in \mathbb{I})\right]} \sim \frac{-\log(\varepsilon)}{\mathbb{P}(\mathbf{h} \in \mathbb{I})}.$$

Such a strategy is therefore only reasonable when $\mathbb{P}(\mathbf{h} \in \mathbb{I})$ is not too small.

[7] For simplicity, in the formula below, we do not mention the mapping of $\mathbb{R}^S$ into $\mathbb{R}^{\mathcal{T}}$ necessary for building h^k.

4 Approximation experiments

4.1 Simulation scenario

Our first goal is to empirically assess the ability of a composition of convolutions to approximate a target atom $H \in \mathbb{R}^{\mathcal{P}}$. We are also interested in observing the influence of the number of kernels K and of the kernel supports on the approximation error. In order to do so, this section presents results obtained for several 1D and 2D target atoms H (i.e. $d = 1$ or 2) that have been selected from dictionaries commonly used in signal and image processing. For all our experiments, we consider $K \leq 11$ and $S \leq 25$.

Given a dimension $d \in \{1,2\}$ and a size $c \in \mathbb{N}$, we consider the support mappings $(S^k)_{1\leq k\leq K} \in (\mathcal{P}^S)^K$ such that for all $k \in \{1,\dots,K\}$

$$\mathrm{rg}\left(S^k\right) = k\{1,2,\dots,c\}^d. \qquad (15)$$

Figure 1 shows translations of this support mapping for $K = 4$ and $c = 3$.

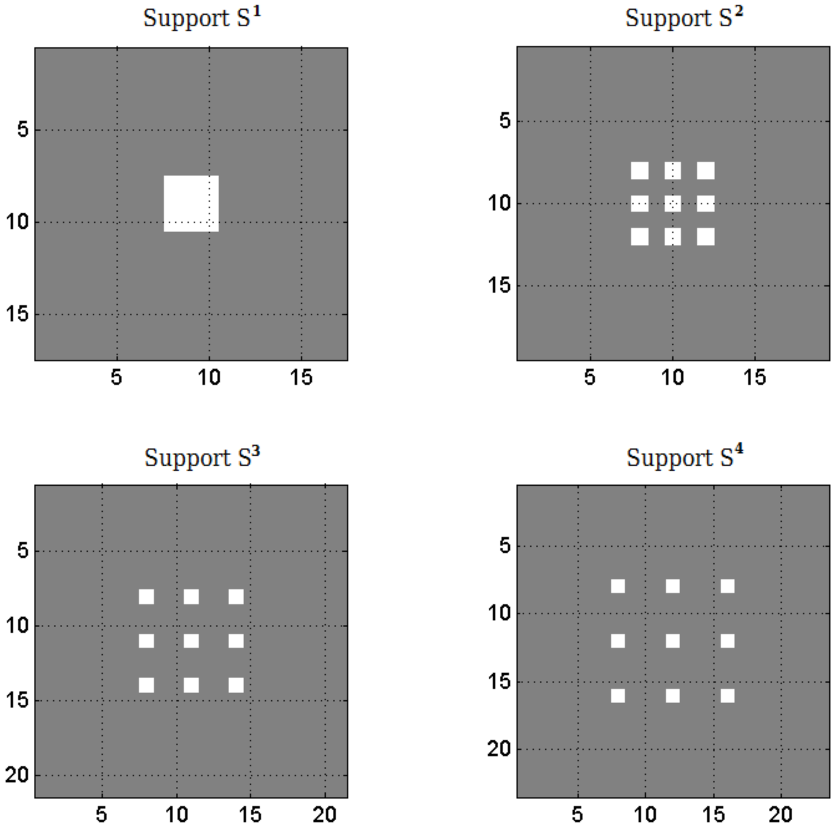


Figure 1: Translations of supports $\mathrm{rg}\left(S^k\right)$ described by (15), for $k \in \{1,2,3,4\}$ and for $c = 3$ (i.e. $S = 3 \times 3$.

Notice that, for this kernel supports, it is not difficult to check that the reachable support (see (3)) is

$$\begin{aligned}\mathcal{S} &= \left\{\sum_{k=1}^{K} k, \dots, \sum_{k=1}^{K} ck\right\}^{d} \\ &= \left\{\frac{K(K+1)}{2}, \dots, c\frac{K(K+1)}{2}\right\}^{d}\end{aligned}$$

Therefore, its width is given by $\frac{K(K+1)(c-1)}{2}$ and its size (length or area) is given by $\left(\frac{K(K+1)(c-1)}{2}\right)^{d}$. This has to be compared with the size of the search space which is given by Kc^{d}. The ratio between these two quantities correspond to "compression ratio" when describing the atom with convolution kernels. Asymptotically, this ratio behaves like $\frac{1}{K^{2d}}$. This suggest that the approximation error should increase, when K is small, at least when the reachable support corresponds to the support of H. The ratio between the computation complexity of the straightforward computation of the convolutions with H and the convolution using the kernels is $\frac{\#H}{Kc^{d}}$.

For all the experiments in Section 4, we consider a size $N \in \mathbb{N}$, a dimension $d \in \{1,2\}$ and take $\mathcal{P} = \{1,\dots,N\}^{d}$. We consider a target atom $H \in \mathbb{R}^{N^{d}}$, a code $\alpha \in \mathbb{R}^{N^{d}}$ and a Gaussian noise $b \in \mathbb{R}^{N^{d}}$ of standard deviation $\sigma \geq 0$. For each experiment, the quantities N, d, H, α, σ are provided in the subsection describing the experiment. Given these quantities, we compute u according to (1). Then, Algorithm 2 has been run for a given number R of restarts and the result having the smallest objective function value is kept. The value of R depends on the experiment and is provided in the corresponding subsection. In the sequel, the result of this process is denoted $(\lambda, (h^{k})_{1 \leq k \leq K}) \in \mathbb{R} \times (\mathbb{R}^{\mathcal{P}})^{K}$.

Given a result $(\lambda, (h^{k})_{1 \leq k \leq K}) \in \mathbb{R} \times (\mathbb{R}^{\mathcal{P}})^{K}$, we evaluate the quality of the approximation of H by $\lambda h^{1} * \cdots * h^{K}$ using the peak-signal-to-noise ratio (PSNR). Moreover, in order to take into account that the size of the support of H might be much smaller than $\#\mathcal{P}$, the PSNR is normalized according to the size of the support of H. More precisely, it is defined by

$$\mathrm{PSNR}_{H} = 10.\log_{10}\left(\frac{r^{2}}{\mathrm{MSE}_{H}}\right).$$

where $r = \max_{p \in \mathcal{P}}(H_{p}) - \min_{p \in \mathcal{P}}(H_{p})$ is the dynamic range of the atom H and the mean-square-error (MSE) is defined by:

$$\mathrm{MSE}_{H} = \frac{\|\lambda h^{1} * \cdots * h^{K} - H\|_{2}^{2}}{\#\operatorname{supp}(H)}.$$

We also provide a feature reflecting both the quality of the convergence and the level of regularization induced by the structure of the composition of convolution. It takes the form

$$\mathrm{Conv} = \frac{\|\lambda \alpha * h^{1} * \cdots * h^{K} - u\|_{2}^{2}}{\|u\|_{2}^{2}}. \tag{16}$$

The latter feature is more difficult it interpret. If it is large, either the convergence has not been reached or the values of K and S are too small to obtain a good approximation of H. If it is small, the result is close to a global minimum and the values of K and S do allow a good approximation of u. Notice that the latter property might be a drawback when u is contaminated by a strong noise.

4.2 1D targets

4.2.1 Modified Discrete Cosine Transform

The modified discrete cosine transform (MDCT) has been used successfully in several signal processing (i.e. $d = 1$) applications such as, for instance, audio coding [18]. The aim of this experiment is to approximate a MDCT with a composition of convolutions. In order to do so, we apply the inverse MDCT to a Dirac delta function located at a given frequency, in a signal of size $N = 256$

. We then apodize the MDC using the sine window $(w_p)_{1\leq p\leq 256}$ defined, for all $p \in \{1,\ldots,256\}$, by $w_p = \sin[\pi\frac{p}{256}]$. This window is, for instance, used in MDCT analysis for time-domain aliasing cancellation .

We have tested the frequency 5 and 50 and obtained, in that manner two target atoms H. The code α is a Dirac delta function located at $p = 1$. The noise standard deviation is $\sigma = 0$. This is a simple case where u is just a translation of H. We have used $R = 50$ restarts. For this experiment, we consider $K = 8$, $S = c = 9$. Moreover, as for all the experiments in this section, the supports of the kernels are according to (15).

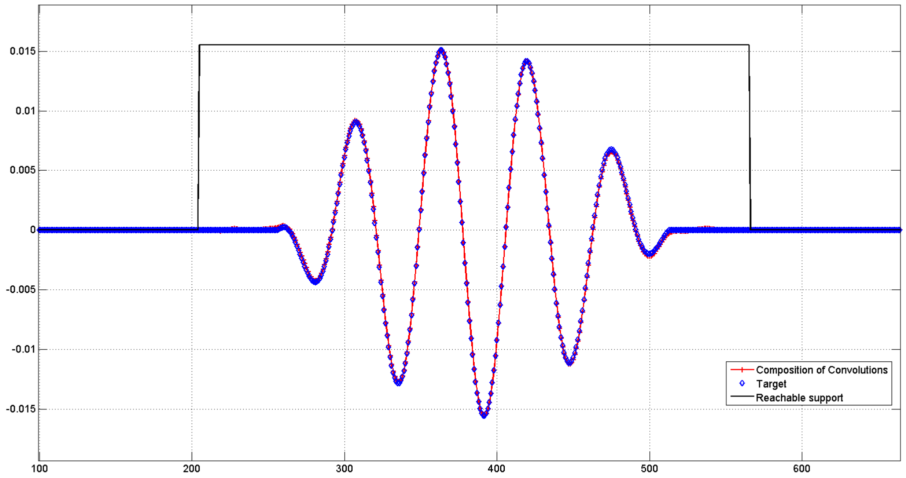


Figure 2: Approximation of a low frequency MDCT (apodized by a sine window) by the convolution of $K = 8$ kernels of sparsity $S = 9$. We have $\text{PSNR}_H = 48.17\text{dB}$.

We display on Figure 2 and 3 the results obtained respectively for the frequencies 5 and 50. More precisely, on each of these figure we represent the approximation $h^1 * \cdots * h^K$, H and the reachable support . Notice that the resulting approximations are very accurate.

We also run the same experiment for the frequency 50 and when u contains an additive white Gaussian noise of variance $\sigma^2 = 10^{-5}$. The normalized value PSNR_H between u and H is 20 dB. We give in Table 1 the values of PSNR_H, for $5 \leq K \leq 11$ and $5 \leq S \leq 9$. In most cases, PSNR_H is larger than that of the noisy data. This means that some denoising has been obtained.

4.2.2 Sinc function

This experiment consists of approximating the sinc function used to perform a linear zoom of factor $Z = 3$ of a signal of size 128 . We therefore have $d = 1$ and $N = 3 * 128 = 384$. The target atom H is a sinc function obtained by computing the inverse Fourier transform of the characteristic function of a centered interval of length $N/3$. The signal to be zoomed corresponds to the first 128 values of the 128th column of the Barbara image. The code α has been built by upsampling

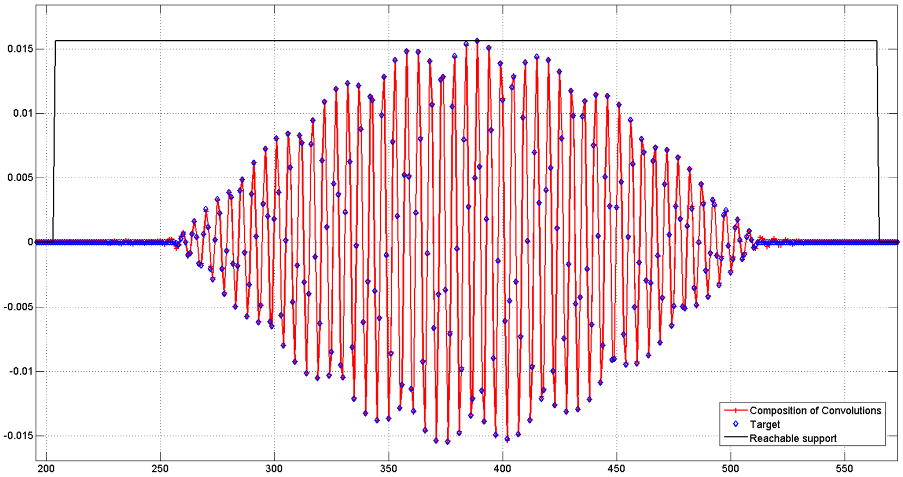


Figure 3: Approximation of a high frequency MDCT (apodized by a sine window) by the convolution of $K = 8$ kernel of sparsity $S = 9$. We have $\text{PSNR}_H = 52.16$dB.

PSNR_H(dB)	$K = 5$	$K = 7$	$K = 9$	$K = 11$
$S = 3$	12.93	13.93	14.60	14.60
$S = 5$	14.06	16.29	20.72	25.30
$S = 7$	15.81	19.35	23.92	29.04
$S = 9$	17.55	24.31	26.15	26.43

Table 1: MDCT (frequency 50) approximation. PSNR_H for several values of K and S. The normalized value PSNR_H between u and H is 20 dB.

this signal by a factor $Z = 3$. The upsampling is performed by inserting 2 zeros between every couple of neighbors of the initial signal. The signal u has been set according to (1), for different noise levels.

For the first experiment, we set $\sigma = 0$ (noise-free scenario) and use $R = 10$ restarts of Algorithm 2. We consider $K = S = 9$. Moreover, as for all the experiments in this section, the supports of the kernels are according to (15).

The code α, and the comparison between the estimated $\lambda\alpha * h^1 * \cdots * h^K$ and the measure u are displayed on Figure 4. First, we observe that $\lambda\alpha * h^1 * \cdots * h^K$ accurately approximates the function u. This suggests that the algorithm has probably reached a global minimum. This is confirmed by the value of Conv =?? which is very small.

We display on Figure 5, the target sinc atom H and the approximation result $\lambda * h^1 * \cdots * h^K$. We see that the resulting composition of convolutions $\lambda * h^1 * \cdots * h^K$ is a good approximation of the sinc function.

We also run the same experiment, for $K \in \{6,7,8,9,10\}$ and $S \in \{5,6,7,8,9\}$, $R = 50$ and when u contains an additive white Gaussian noise of variance $\sigma^2 = 3$. The normalized value PSNR_H between u and $\alpha * H$ is 28.20 dB.

Table 2 contains the values of PSNR_H In most cases, PSNR_H is a little larger than that of the noisy data, which means that we have achieved some degree of denoising. Note that for $S = 7$, when K reaches 7, PSNR_H starts decreasing.

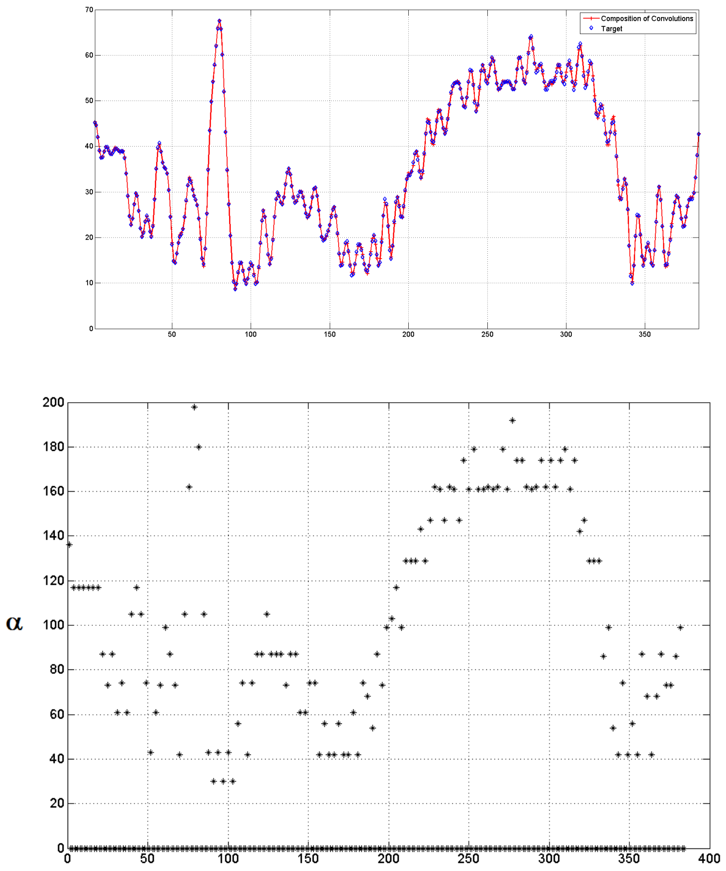


Figure 4: Approximation of a 1D sinc function for $(K,S) = (9,9)$. Top: Comparison between the estimated $\lambda\alpha * h^1 * \cdots * h^K$ and the measure u. We have ; Bottom: The code α.

.

PSNR_H(dB)	$K = 3$	$K = 5$	$K = 7$	$K = 9$
$S = 3$	27.85	30.95	34.12	30.64
$S = 5$	29.08	29.23	32.24	34.02
$S = 7$	34.39	36.22	24.04	14.51

Table 2: 1D sinc function approximation in the context of a zoom. PSNR_H for several values of K and S. The normalized value PSNR_H between u and $\alpha * H$ is 28.20 dB.

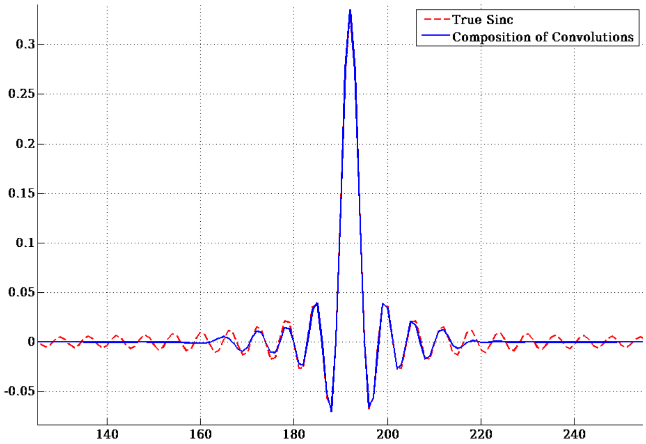


Figure 5: Approximation of a 1D sinc function with $(K,S) = (9,9)$. The target sinc atom H and the composition of convolutions $\lambda h^1 * \cdots * h^K$. We have $\text{PSNR}_H = 44.47$dB.

Conv	$K=3$	$K=5$	$K=7$	$K=9$
$S=3$	1.39	1.26	1.17	1.17
$S=5$	0.97	0.94	0.94	0.94
$S=7$	0.91	0.90	0.92	0.91

Table 3: Convergence criterion for the zoom experiment. Conv for several values of K and S.

4.3 2D targets

4.3.1 Curvelet

The aim of this experiment is to approximate curvelet atom H in an image (i.e. $d = 2$) of size $N \times N$ with $N = 128$. The curvelet is obtained by applying the inverse curvelet transform to a Dirac delta function, using the MCALAB toolbox [6]. The code α corresponds to a Dirac delta function located at the barycenter of the curvelet. Once again, the support mapping applied is the one described in (15), with either $c = 3$ or $c = 5$. We every values of K satisfying $3 \leq K \leq 9$. We consider $\sigma = 0$ sothat u is a simple translation of H. We used $R = 10$ restarts.

We display on Figure 6 the target atom H and $\lambda h^1 * \cdots * h^K$, for $K = 7$ and $S = 5 * 5$. We observe that although $\text{PSNR}_H = 49.2$dB, the accuracy is not homogeneous. In particular, the tails of curvelet is not properly captured.

PSNR_H(dB)	$K=3$	$K=5$	$K=7$	$K=9$
$S = 3\times 3$	36.46	40.28	40.56	41.28
$S = 5\times 5$	43.44	49.70	50.27	46.82

Table 4: Curvelet: PSNR_H for several values of K.

Table 4 contains the values of PSNR_H for various values of K and c. Notice that in the setting of the current experiment, we expect that increasing K and S improves the accuracy. Despite the value

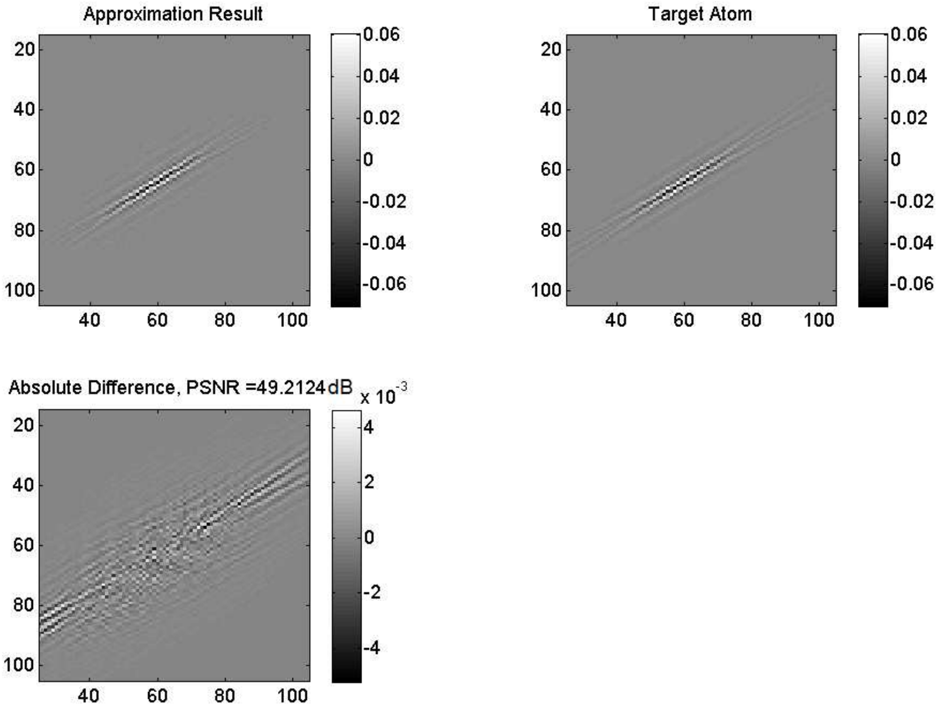


Figure 6: Curvelet approximation with $K = 7$ and $S = 5 \times 5$. Comparison between $\lambda h^1 * \ldots * h^7$ and the target curvelet atom H. We have $\mathrm{PSNR}_H = 54.62$.

PSNR_H value for $S = 5 \times 5$ and $K = 9$, this is what we observe. This can only be caused by some lack of convergence for that particular case. Note that the reachable support associated with these parameters always contains most of the support of the atom.

Finally, Figure 7 shows the kernels $(h^k)_{1 \leq k \leq K}$ computed for $K = 7$ and $S = 5 \times 5$. We see that many kernel coefficients are close to zero, i.e., only the coefficients along the main curvelet axis have significant values. It is obvious that the simple isotropic dilation of the supports defined by (15) is not appropriate for this curvelet. This raises the question of the adaptation of support mappings $(S^k)_{1 \leq k \leq K}$ to the atom's geometry.

4.3.2 Cosine

The aim of this experiment is to approximate an atom representing a 2D cosine function in an image of size 64×64 (i.e. $d = 2$ and $N = 64$). In the context of image processing, such an atom can be seen as a large local cosines or a Fourier atoms. Both are widely used in image processing. The interest of this atom is that it covers the whole image and is of a rather large support. Beside, patches of this size are difficult to handle with existing dictionary learning strategies. The considered atom is given by

$$H_p = \cos\left(2\pi \frac{\langle p, (\quad,\quad)\rangle}{N}\right) \qquad ,\forall p \in \{1, \ldots, 64\}^2.$$

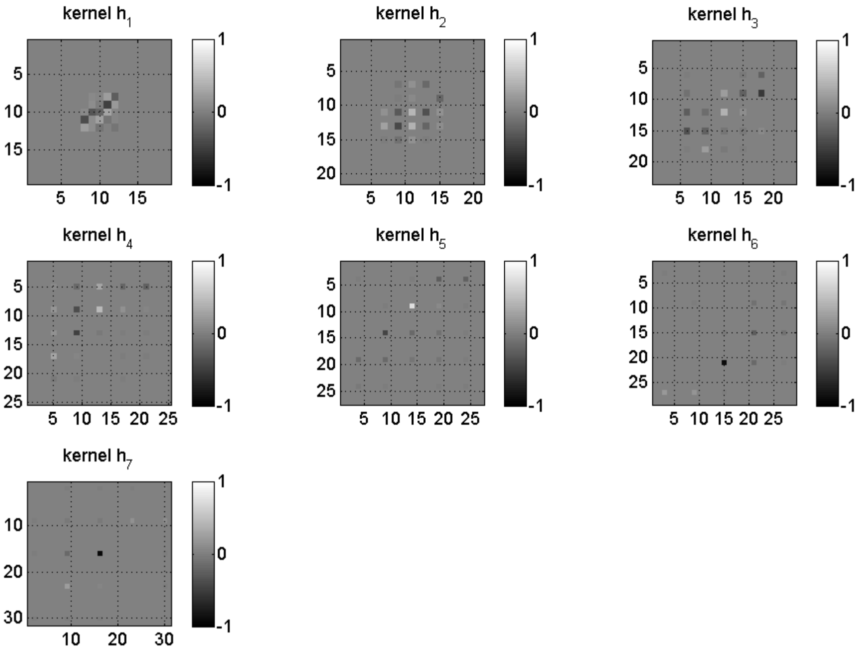


Figure 7: Curvelet approximation for $K = 7$ and $S = 5 \times 5$. The computed kernels $(h^k)_{1 \leq k \leq 7}$.

The code α is a Dirac delta function located at the center of the image. The standard deviation of the noise is $\sigma = 0$. Therefore, u is a simple translation of the cosine atom H.

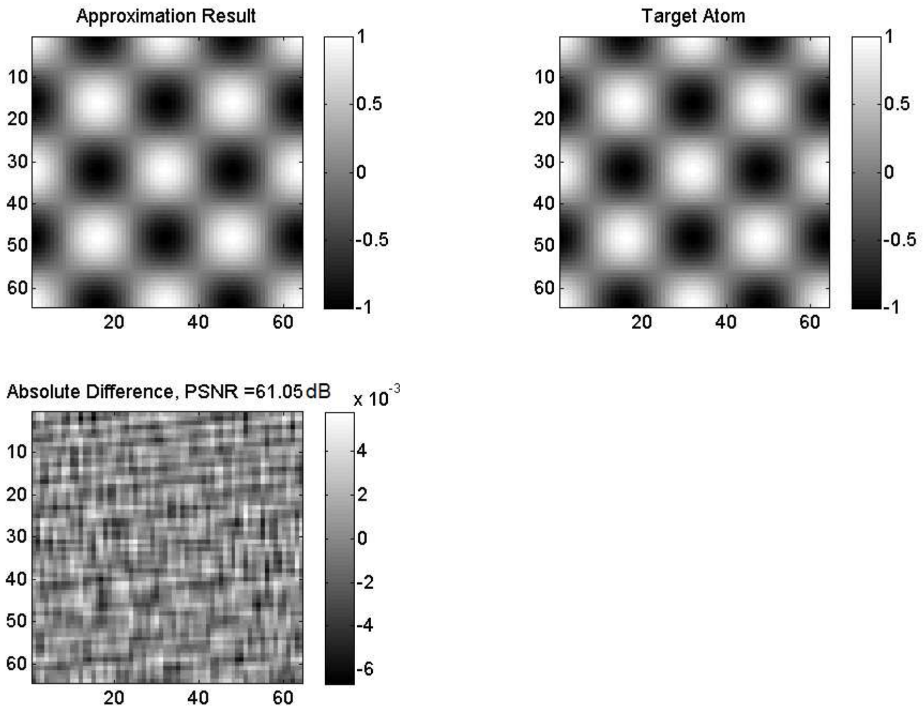
The support mapping is the same as for the previous experiment (see (15)) with, either $c = 3$ or $c = 5$. We tested values of K such that $3 \leq K \leq 9$. We used $R = 10$ restarts and selected the best match for each parameter setting.

PSNR_H(dB)	$K = 3$	$K = 5$	$K = 7$	$K = 9$
$S = 3 \times 3$	12.30	12.98	15.61	49.05
$S = 5 \times 5$	12.47	18.10	61.05	113.00

Table 5: 2D Cosine: PSNR_H for several values of K and S.

Table 5 shows examples of approximation performance in terms of PSNR_H. In this experiment, since minimizing the objective function of (P_1) is equivalent to maximizing the PSNR_H, we expect PSNR_H to increase with the parameters. Despite the non-convexity of (P_1), that is indeed what we observe in Table 5. Notice that the values of PSNR_H are rather large. For $K = 9$ and $c = 5$, the "compression ratio" (i.e. the ratio between the number of variables describing the kernels and the image of the cosine) is $\frac{9 \times 5 \times 5}{64 \times 64} \sim \frac{1}{20}$.

The result $\lambda h^1 * \cdots * h^K$, the target atom as well as the difference between these images are displayed in Figure 8, for $K = 7$ and $S = 5 \times 5$. The accuracy is very good for the approximation of a cosine with a composition of convolutions.



Approximation Result

Target Atom

Absolute Difference, PSNR =61.05dB

Figure 8: Cosine approximation with $K = 7$ and $S = 5 \times 5$.

4.3.3 Sinc 2D

The aim of this experiment is to approximate an atom representing a 2D sinc function in an image of size 128 × 128 (i.e. $d = 2$ and $N = 128$). The sinc target atom H is built the same way as in Section 4.2.2 using the inverse Fourier transform of a centered square whose side is of length . The code α is a Dirac delta function located at the center of the image. We consider a noise free case and set $\sigma = 0$.

The support mapping is again according to (15). Notice that we are not using the knowledge that the sinc function is separable. We have tested the values of K such that $3 \leq K \leq 9$. We used $R = 10$ restarts and selected the best match for each parameter setting.

PSNR$_H$(dB)	$K = 3$	$K = 5$	$K = 7$	$K = 9$
$S = 3 \times 3$	46.35	47.99	49.43	50.84
$S = 5 \times 5$	49.32	53.08	54.62	46.76

Table 6: 2D Sinc: PSNR$_H$ for several values of K and S.

Table 6 shows examples of approximation performance in terms of PSNR$_H$. As with previous experiments where α is a dirac delta function and $\sigma = 0$, we expect PSNR$_H$ to increase with the parameters. For this experiment, this is not the case since PSNR$_H$ decreases for $K = 9$ and $S = 5 \times 5$. This might be due to a lack of convergence.

The result $\lambda h^1 * \cdots * h^K$, the target atom as well as the difference between these images are displayed in Figure 9, for $K = 7$ and $S = 5 \times 5$.

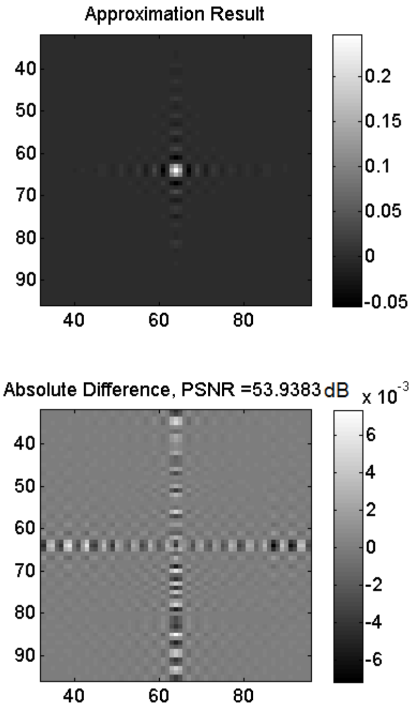


Figure 9: 2D Sinc function approximation for $K = 7$ and $S = 5 \times 5$.

4.3.4 Experiments based on a wavelet decomposition

In this experiment, we consider a scenario better reflecting the difficulties of dictionary learning. More precisely, we consider $d = 2$, $N = 512$, the target atom H is an atom of wavelet and code α is resulting from the wavelet coefficient of a natural image. ~~The objective is to approximate the image with an atom that will itself be a good approximation of the wavelet used in the decomposition.~~ More precisely, the following operations have been conducted

- Select an Image. (We took the Barbara image.)
- Compute the wavelet transform of the image using the Daubechies wavelet DB4 at level L. (We used the official MatLab wavelet toolbox and took $L = 3$.)
- Select the set of coefficients associated with an orientation and a given decomposition level l such that $1 \leq l \leq L$. (We took the low frequency at level $l = L = 3$ for the first experiment and the horizontal detail at level $l = 3$ for the second experiment.)
- Set the non selected wavelet coefficients to zero and compute the inverse wavelet transform. Add Gaussian white noise of variance σ^2 to obtain u. (We took $\sigma^2 = 5$.)
- Take α equal to a zoom of factor 2^l of the selected coefficients. The zoom consists in interpolating with zeros. Note that the code α has the same size as u.

- Solve problem (P_k) with the code α, the target atom u, $R = 1$, with a support mapping defined by (15) for the parameters $K = 6$ and $S = 3 \times 3$ (i.e. $c = 3$). Notice, that we do not take into account that we know supports of a composition of convolution that leads to wavelet atoms.

The experiments for the low frequency wavelet atom at level 3 and the horizontal detail wavelet atom are respectively shown in figures 10 and 11. Notice that the conditioning of α is less favorable for the estimation of the low frequency wavelet atom.

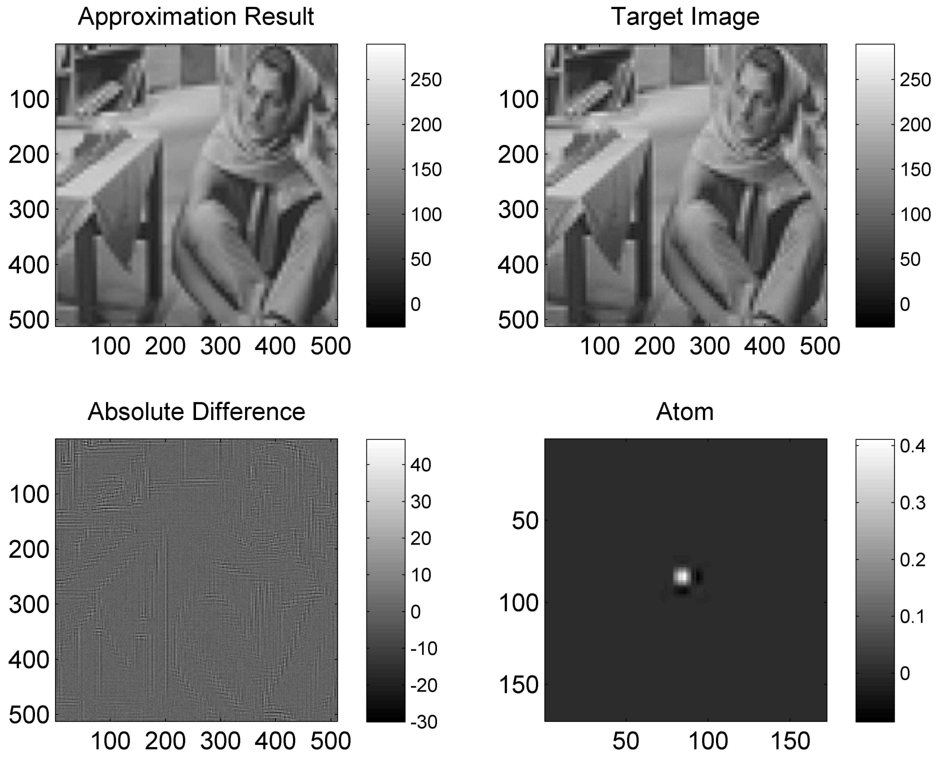


Figure 10: Estimation of the low frequency wavelet atom at level 3 (Conv = 3.1 . Top left: $\lambda\alpha * h^1 * \cdots * h^K$. Top right: target measure. Bottom left: difference $\lambda\alpha * h^1 * \cdots * h^K - u$. Bottom right: composition of convolutions $\lambda * h^1 * \cdots * h^K$ approximating the target wavelet atom.

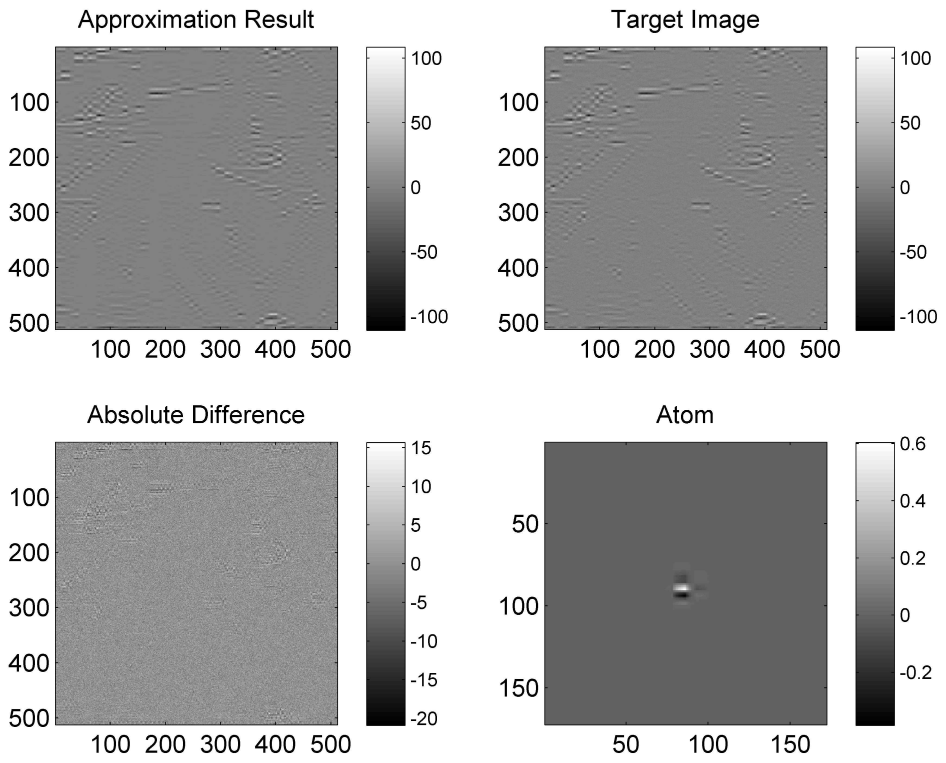


Figure 11: Estimation of the horizontal detail atom at level 3 (Conv $= 1.2$. Top left: $\lambda\alpha * h^1 * \cdots * h^K$. Top right: target measure. Bottom left: difference $\lambda\alpha * h^1 * \cdots * h^K - u$. Bottom right: composition of convolutions $\lambda * h^1 * \cdots * h^K$ approximating the target wavelet atom.

5 Convergence Assessment

5.1 Simulation Scenario

This section evaluates

$$\mathbb{P}\,(\text{not global}) \qquad \text{and} \qquad R_\varepsilon = \frac{\log(\varepsilon)}{\log\left(1-\mathbb{P}\left(\mathbf{h}\in\mathbb{I}\right)\right)}$$

for various supports, kernels and noise level. All the experiments have been conducted with one-dimensional signals of size $\#\mathcal{P} = 128$ and $(K,S) \in \{2,\ldots,7\}\times\{2,\ldots,10\}$. We have considered random support mappings $\mathbf{S} = (S^k)_{1\leq k\leq K}$. For every $k \in \{1,\ldots,K\}$, the support mapping S^k maps $\{1,\ldots,S\}$ into S distinct elements randomly drawn according to a uniform distribution in $\{1,...,10\}$. Moreover, for any $(k_1,k_2) \in \{1,\ldots,K\}^2$, with $k_1 \neq k_2$, $\mathrm{rg}\left(S^{k_1}\right)$ and $\mathrm{rg}\left(S^{k_2}\right)$ are independent random vectors. We also consider K independent random kernels

$$h_p^k \begin{cases} \sim \mathcal{N}(0,1) & \text{, if } p \in \mathrm{rg}\left(S^k\right) \\ = 0 & \text{, otherwise.} \end{cases}$$

Finally, the code is set to $\alpha = (1,0,\ldots,0)$ (i.e., no translation) and the image u is obtained by convolving the kernels

$$u = \alpha * h^1 * \cdots * h^K + b$$

where $b \sim \mathcal{N}(0,\sigma^2 \mathbb{1}_{\mathcal{S}})$, σ^2 is the noise variance and the set $\mathcal{S}$ is the "reachable support" defined in (3). Note that u is zero outside of the reachable support.

5.2 Performance measure

Given a problem defined by $(u,\alpha,\mathbf{S})$, a global minimizer $\mathbf{h}^* = (h^{*,k})_{1\leq k\leq K} \in (\mathbb{R}^{\mathcal{P}})^K$ of (P_0) and a solution $\overline{\mathbf{h}} = (\overline{h}^k)_{1\leq k\leq K} \in (\mathbb{R}^{\mathcal{P}})^K$ provided by Algorithm 2, we denote the approximation error by

$$E_a(u,\alpha,\mathbf{S}) = \|\alpha * h^{*,1} * \ldots * h^{*,K} - u\|_2^2.$$

For the ~~solutions~~ sample problem constructed in the previous section, we expect that

$$E_a(u,\alpha,\mathbf{S}) \leq \sigma^2\left(\#\mathcal{S}\right),$$

where σ^2 is the noise variance. Moreover, when $\sigma = 0$, we know that $E_a(u,\alpha,\mathbf{S}) = 0$. We also denote the numerical error by

$$E_n\left(\overline{\mathbf{h}},u,\alpha,\mathbf{S}\right) = \|\alpha * \overline{h}^1 * \ldots * \overline{h}^K - u\|_2^2 - E_a(u,\alpha,\mathbf{S}).$$

The only quantity that we can actually observe is the sum of these two errors

$$\|\alpha * \overline{h}^1 * \ldots * \overline{h}^K - u\|_2^2 = E_a(u,\alpha,\mathbf{S}) + E_n((\overline{\mathbf{h}},u,\alpha,\mathbf{S}).$$

We therefore consider that Algorithm 2 has converged to a global minimum if

$$\|\alpha * \overline{h}^1 * \ldots * \overline{h}^K - u\|_2^2 \leq \sigma^2\left(\#\mathcal{S}\right) + 10^{-4}\|u\|_2^2. \tag{17}$$

Of course, this notion is not very accurate when σ^2 is large.

5.3 Evaluation of $\mathbb{P}\,(\textbf{not global})$

For any fixed $(K,S) \in \{2,\ldots,6\} \times \{2,\ldots,10\}$, we have generated $L = 50K^2$ experiments. Each experiment or input of the algorithm is depicted by an index $l \in \{1,\ldots,L\}$. For every experiment, we consider $R = 25$ random initializations according to a uniform distribution defined in the set of constraints associated with (P_1), as described in Section 3.5. The corresponding outcome of Algorithm 2 is referred to as the rth result with $r \in \{1,\ldots,R\}$. Finally, for any $(l,r) \in \{1,\ldots L\} \times \{1,\ldots,R\}$, we introduce the following indicator function

$$\mathbb{1}\,(l,r) = \begin{cases} 1, & \text{if (17) holds for the } r\text{th result obtained from the } l\text{th input,} \\ 0, & \text{otherwise.} \end{cases}$$

The probability of reaching a global minimum of problem (P_1) is estimated as follows

$$\mathbb{P}\,(\text{global minimizer}) \simeq \frac{1}{LR} \sum_{l=1}^{L} \sum_{r=1}^{R} \mathbb{1}\,(l,r).$$

5.4 Results

Figures 12 and 13 show the results obtained in the noiseless ($\sigma = 0$) and noisy ($\sigma = 0.1 * \sqrt{5}$) cases respectively. In each figure, the curves show $\mathbb{P}\,(\text{global minimizer})$ or $\frac{\log(\varepsilon)}{\log(1-\mathbb{P}((h^k)_{1\leq k\leq K}\in\mathbb{I}))}$ for a given value of K whereas the x axis indicates the value of S. We can see that for very sparse kernels ($S \leq 3$), the probability of success is quite high. However, this probability drops significantly when the support size increases. Surprisingly, $\mathbb{P}\,(\text{global minimizer})$ ~~generally~~ increases when the support size increases. The more kernels we use (i.e., the larger K), the steeper the decrease and increase. These results show that it is possible to obtain convergence to a global minimum with only a few restarts of the proposed algorithm even for relatively large values of K~~provided K has been chosen properly~~. The last experiments obtained in the noisy case show similar patterns. As a consequence, the described convergence properties seem to be robust to noise.

6 Conclusions

to be written. Include future work

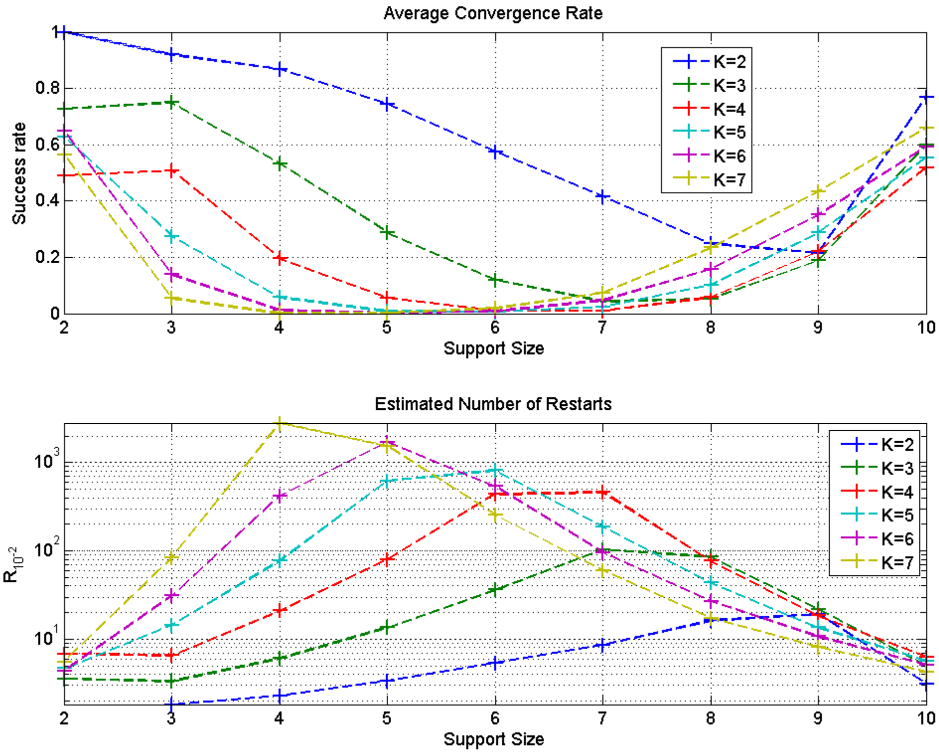


Figure 12: Convergence test for $\sigma = 0$: Average rate of convergence to a global minimum (top) for every $K \in \{2,\dots,7\}$ and corresponding number of restarts R_ε to guarantee $\mathbb{P}\,$(global minimizer) $\geq 99\%$ (bottom). For every $K \in \{2,\dots,7\}$, the rate of convergence has been averaged over $L = 50K^2$ inputs from which we have computed $R = 25$ outputs.

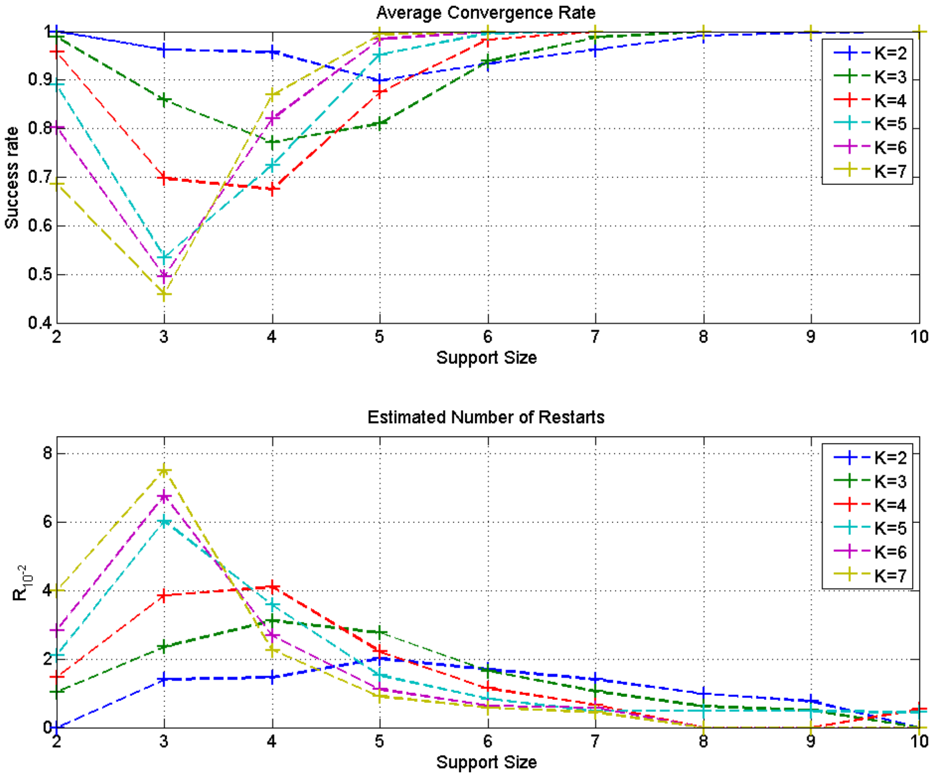


Figure 13: Convergence test for $\sigma^2 = 5.10^{-2}$: Average rate of convergence to a L_2 ball of radius $\sigma\sqrt{\sharp\mathcal{S}}$ around a global minimum (top) for every $K \in \{2,\dots,6\}$ and corresponding number of restarts R_ε to guarantee $\mathbb{P}(\text{global minimizer}) \geq 99\%$ (bottom). For every $K \in \{2,\dots,6\}$, the rate of convergence has been averaged over $L = 50K^2$ inputs, from which we have computed $R = 25$ outputs.

References

[1] M. Aharon, M. Elad, and A. Bruckstein. The k-svd, an algorithm for designing overcomplete dictionaries for sparse representation. IEEE Trans. Signal Process., 54(11):4311–4322, 2006.

[2] J. Cooley and J.W. Tukey. An algorithm for the machine calculation of complex fourier series. Mathematics of Computation, 19:297–301, 1965.

[3] J.M. Duarte-Carvajalino and G. Sapiro. Learning to sense sparse signals: Simultaneous sensing matrix and sparsifying dictionary optimization. IEEE Trans. Image Process., 18(7):1395–1408, 2009.

[4] M. Elad. Sparse and redundant representations: From theory to applications in signal and image processing. Springer, 2010.

[5] K. Engan, S. O. Aase, and J. Hakon Husoy. Method of optimal directions for frame design. In Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing (ICASSP), pages 2443–2446, 1999.

[6] J. Fadili, J.-L. Starck, and D. Donoho M. Elad. Mcalab: Reproducible research in signal an image decomposition and inpainting. IEEE Computing in Science and Engineering, 2010.

[7] R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. Proximal methods for sparse hierarchical dictionary learning. In Proc. Int. Conf. Machine Learning (icml), 2010.

[8] R. Jenatton, J. Mairal, G. Obozinski, and F. Bach. Proximal methods for hierarchical sparse coding. J. Mach. Learning Research, 12:2297–2334, 2011.

[9] S. Lesage, R. Gribonval, F. Bimbot, and L. Benaroya. Learning Unions of Orthonormal Bases with Thresholded Singular Value Decomposition. In Proc. IEEE Int. Conf. Acoust., Speech, and Signal Processing (ICASSP), volume V, pages V/293–V/296, Philadelphia, PA, United States, 2005.

[10] M.S. Lewicki and T.J. Sejnowski. Learning overcomplete representations. Neural Computation, 12(2):337–365, 2000.

[11] B. Mailhé, S. Lesage, R. Gribonval, F. Bimbot, and P. Vandergheynst. Shift-invariant dictionary learning for sparse representations: extending K-SVD. In Proc. European Signal Process. Conf. (EUSIPCO), page 5 p., 2008.

[12] J. Mairal, F. Bach, and J. Ponce. Task-driven dictionary learning. IEEE Trans. Patt. Anal. Mach. Intell., 34(4):791–804, 2012.

[13] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. J. Mach. Learning Research, 11:10–60, 2010.

[14] J. Mairal, G. Sapiro, and M. Elad. Learning multiscale sparse representations for image and video restoration. SIAM Multiscale Modeling and Simulation, 7(1):214–241, 2008.

[15] M. E Muller. A note on a method for generating points uniformly on n-dimensional spheres. Comm. Assoc. Comput. Mach., 2:19–20, April 1959.

[16] B. A. Olshausen and D. J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? Vision Research, 37(23):3311 – 3325, 1997.

[17] B. Ophir, M. Lustig, and M. Elad. Multi-scale dictionary learning using wavelets. IEEE J. Sel. Topics Signal Process., 5(5):1014–1024, 2011.

[18] T. Painter and A. Spanias. Perceptual coding of digital audio. Proc. IEEE, 88(4):451–515, 2000.

[19] R. Rubinstein, A.M. Bruckstein, and M. Elad. Dictionaries for sparse representation. Proc. IEEE - Special issue on applications of sparse representation and compressive sensing, 98(6):1045–1057, 2010.

[20] R. Rubinstein, M. Zibulevsky, and M. Elad. Double sparsity: Learning sparse dictionaries for sparse signal approximation. IEEE Trans. Signal Process., 58(3):1553–1564, 2010.

[21] P. Sallee and B.A. Olshausen. Learning sparse multiscale image representations. Advances in neural information processing systems, 15:1327–1334, 2003.

[22] J.-L. Starck, J. Fadili, and F. Murtagh. The undecimated wavelet decomposition and its reconstruction. IEEE Trans. Image Process., 2007.

[23] S. Zhou T. Tsiligkaridis, AO Hero. Convergence properties of Kronecker graphical lasso algorithms. IEEE Trans. Signal Process., 2013. accepted for publication.

[24] J.J. Thiagarajan, K.N. Ramamurthy, and A. Spanias. Multilevel dictionary learning for sparse representation of images. In Proc. IEEE Digital Signal Process. Workshop and IEEE Signal Process. Education Workshop (DSP/SPE), pages 271–276, 2011.

[25] A. Wiesel. Geodesic convexity and covariance estimation. IEEE Trans. Signal Process., 60(12):6182–6189, Dec. 2012.