Learning fast dictionaries for sparse decompositions using low-rank tensor decompositions

Cassio Fraga Dantas, Jérémy E Cohen, Rémi Gribonval

Learning fast dictionaries for sparse decompositions using low-rank tensor decompositions

Cássio F. Dantas[†], Jérémy E.Cohen[◇⋆], and Rémi Gribonval[†]

[†]Univ Rennes, Inria, CNRS, IRISA, France (cassio.fraga-dantas@inria.fr)
[◇]University of Mons, Belgium (jeremy.cohen@umons.ac.be)

Abstract. A new dictionary learning model is introduced where the dictionary matrix is constrained as a sum of R Kronecker products of K terms. It offers a more compact representation and requires fewer training data than the general dictionary learning model, while generalizing Tucker dictionary learning. The proposed Higher Order Sum of Kroneckers model can be computed by merging dictionary learning approaches with the tensor Canonic Polyadic Decomposition. Experiments on image denoising illustrate the advantages of the proposed approach.

Keywords: Kronecker product, tensor data, dictionary learning.

1 Introduction

Multi-dimensional data arise in a large variety of applications such as telecommunications, biomedical sciences, image and video processing to name a few [10].

Explicitly accounting for this tensorial structure of the data can be more advantageous than relying on its vectorized version and losing the original neighboring relations, besides providing more efficient and economic representation and subsequent processing. The Kronecker product structure arises naturally when dealing with multi-dimensional (tensorial) data, since it manages to recover the underlying tensorial nature of vectorized data samples. Indeed, when applied to a vectorized tensor, each composing factor of a Kronecker-structured linear operator acts independently on each mode of the data. Conveniently, such operators are more compact to store and can be applied much more efficiently than their unstructured counterpart.

Nevertheless, relatively little attention has been paid to exploiting this type of structure on representation learning methods such as dictionary learning algorithms, which aim at obtaining a set of explanatory variables (the dictionary) capable of sparsely approximating an input dataset. Conventional methods impose no particular structure to the dictionary matrix learned from the vectorized input samples, therefore completely disregarding a potential multi-dimensional characteristic of the data.

⋆ This author acknowledges the support by the F.R.S.-FNRS (incentive grant for scientific research n° F.4501.16)

In this paper, we provide a novel method to induce a generalized version of the Kronecker structure on the dictionary (namely, a sum of Kronecker products). To this end, we draw a parallel between the problem of approximating an arbitrary linear operator by a Kronecker-structured one and the low-rank tensor approximation problem.

The learned dictionary is constrained to the following structure:

$$\mathbf{D} = \sum_{r=1}^{R} \mathbf{D}_1^r \otimes \cdots \otimes \mathbf{D}_K^r = \sum_{r=1}^{R} \bigotimes_{k=1}^{K} \mathbf{D}_k^r;$$

which we refer to as a *rank-R K-Kronecker-structured* matrix (or simply (R, K)-KS and even K-KS when $R = 1$). In [17] R is referred to as the *separation rank*. As we will see in Section 1.1, this structure may lead to significant memory and computational savings when compared to an unstructured matrix. At the same time, because we allow sums of several Kronecker terms, we make the structure more flexible thus increasing its approximation capabilities with respect to the conventional Tucker dictionary model [4].

1.1 Motivations

The quest for Kronecker-structured linear operators has various motivations besides the suitability to multi-dimensional signals: 1) reduced computational complexity; 2) diminished memory requirements and 3) smaller sample complexity on learning applications.

The complexity savings on matrix-vector multiplications are explained as follows. For an $(n \times m)$ K-KS matrix $\mathbf{D} = \mathbf{D}_K \otimes \cdots \otimes \mathbf{D}_1$ with factors $\mathbf{D}_k \in \mathbb{R}^{n_k \times m_k}$, $n = \prod_{k=1}^{K} n_k$ and $m = \prod_{k=1}^{K} m_k$, the matrix-vector product $\mathbf{Dx}$ can be rewritten as

$$\mathbf{y} = (\mathbf{D}_K \otimes \cdots \otimes \mathbf{D}_1)\mathbf{x} \qquad \rightarrow \qquad \underline{\mathbf{Y}} = \underline{\mathbf{X}} \times_1 \mathbf{D}_1 \times_2 \cdots \times_K \mathbf{D}_K \tag{1}$$

where $\underline{\mathbf{Y}} \in \mathbb{R}^{n_1 \times \cdots \times n_K}$ and $\underline{\mathbf{X}} \in \mathbb{R}^{m_1 \times \cdots \times m_K}$ are the tensorized versions of $\mathbf{y} \in \mathbb{R}^{n_1 \ldots n_K}$ and $\mathbf{x} \in \mathbb{R}^{m_1 \ldots m_K}$ respectively [10] and $\times_k$ denotes the mode-k tensor matrix product. In other words, it comes down to multiplying each factor by the corresponding mode on the tensorized version $\underline{\mathbf{X}}$ of the vector $\mathbf{x}$.

Since the composing factors $\mathbf{D}_k$ are much smaller than $\mathbf{D}$, the total complexity for computing (1) can be significantly smaller than the usual $\mathcal{O}(nm)$. In particular, when the factors are all square (i.e. $n_k = m_k$ $\forall k$) the total number of operations is given by $(\prod_{k=1}^{K} n_k)(\sum_{k=1}^{K} n_k)$ [16] compared to $(\prod_{k=1}^{K} n_k)^2$ operations for an unstructured matrix of the same size. For a sum of R Kronecker products, the mentioned complexity is simply multiplied by R.

Besides, the total storage cost for the structured operator is proportional to $(\sum_{k=1}^{K} n_k m_k)$ instead of $(\prod_{k=1}^{K} n_k m_k)$. Similarly, recent studies [15] on the minimax risk for the dictionary identifiability problem showed that the necessary number of samples for reliable reconstruction, up to a given mean squared error, of a KS dictionary within its local neighborhood scales with $(m \sum_{k=1}^{K} n_k m_k)$ compared to $(m \prod_{k=1}^{K} n_k m_k)$ for unstructured dictionaries of the same size [9].

1.2 Related Work

The Kronecker structure was introduced in the Dictionary Learning domain by [8, 13] both addressing only 2-dimensional data (i.e. 2-KS dictionaries). The model was extended to the 3rd-order (3-KS dictionaries) [12,19] and even for an arbitrary tensor order [4, 7] based on the Tucker decomposition, a model coined as Tucker Dictionary Learning. However, none of these works include a sum of Kronecker terms. Even though the formulation in [7] would allow it, they restrict their analysis to the rank-one ($R = 1$) case.

The sum of Kronecker products model was initially explored in the covariance estimation community [3, 17] and was recently used in [5] as an extension of the existing Kronecker-structured dictionaries. But once again, these works addressed only the 2nd-order case ($K = 2$). We now extend [5] for an arbitrary tensor order, thus allowing for both $R \geq 1$ and $K \geq 2$. An advantage of our approach w.r.t [5] and [7] is that here we can choose the desired number of summing terms beforehand without needing to empirically adjust a regularization parameter.

Finally, similarly to what is introduced in this manuscript, Bastelier *et. al.* have recently discussed factorizations strategies for (R,K)-KS matrices, but with the goal of preserving the data structure and thus resorting to orthogonality constraints [2].

1.3 Notation

Throughout this document, we denote $\otimes$ the Kronecker Product and $\circ$ the outer product. Matrices (resp. tensors) are represented by bold uppercase (resp. underscored) letters and the (i, j)th entry of a matrix $\mathbf{D}$ is denoted $d(i, j)$. The vectorization operation, denoted $\text{vec}(\cdot)$, consists in stacking the columns of a matrix and $\text{unvec}(\cdot)$ stands for the converse operation.

2 A Dictionary Learning Algorithm for Tensorial Data

2.1 (R,K)-KS dictionary learning model

Given a training data set in a tensor format $\underline{\mathbf{Y}}$, a naive approach to learn a dictionary from $\underline{\mathbf{Y}}$ is to rearrange its entries into a matrix and apply a workhorse two-way dictionary learning algorithm. However, such a matricization of $\underline{\mathbf{Y}}$ erases mode-wise information, such as the 2D structure of images or color information.

In [5], we have derived a dictionary learning algorithm for dictionaries as sums of 2-Kronecker products, which account for the two-way structure of the original training data. We now wish to extend this approach to tensors $\underline{\mathbf{Y}}$ of any order. For this, we constrain the learned dictionary to be (R, K)-Kronecker-structured.

The dictionary learning model may be computed by solving the following minimization problem:

$$\min_{\mathbf{D} \in \mathcal{C}_K^R,\ \mathbf{X}} \frac{1}{2}\|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 + g(\mathbf{X}) \tag{2}$$

where $\mathcal{C}_K^R$ denotes the set of all (R, K)-KS matrices of size $(n \times m)$ and the training data, arranged as the columns of the matrix $\mathbf{Y} \in \mathbb{R}^{n \times N}$, is to be approximated as the product between the dictionary $\mathbf{D} \in \mathbb{R}^{n \times m}$ and the sparse representation matrix $\mathbf{X} \in \mathbb{R}^{m \times N}$. The sparsity of the columns of $\mathbf{X}$ is enforced by the penalty function g (classical examples are the ℓ_0 and ℓ_1 norms). We also impose the columns of the dictionary to have unit ℓ_2-norm.

Provided that the rank can be set to arbitrarily large values, any linear operator $\mathbf{D}$ may actually be written as a (R,K)-KS matrix [17]. On the other hand, a K-KS matrix has a very specific structure that may not be appropriate for learning a good dictionary in a generic scenario. Therefore, for small rank values, the proposed approach may be understood as a trade-off between precision and complexity, storage and robustness to small training sets.

2.2 Dictionary Learning Algorithm

Following the literature, we employ a sub-optimal alternating minimization strategy to tackle problem (2). At each step, respectively *dictionary update* and *sparse coding*, we optimize with respect to one variable, resp. $\mathbf{D}$ and $\mathbf{X}$, while fixing the other. The procedure is repeated N_{it} times.

The *sparse coding* step can be performed by any existing sparse regression algorithm. In our experiments we use the OMP [11] algorithm.

The difficulty in computing (2) lies in the (R,K)-KS structure imposed on $\mathbf{D}$. In fact, in Sections 3 and 4, we show that this constraint is equivalent to imposing a Canonical Polyadic Decomposition model on a rearranged tensor $\mathcal{R}_K(\mathbf{D})$ obtained from dictionary $\mathbf{D}$. Therefore, for the *dictionary update* step, we propose a projected gradient algorithm, where the projection onto the set of $\mathcal{C}_K^R$ (set of matrices written as a sum of R K-Kronecker products) is approximated by the CPD algorithm applied to the rearranged tensor $\mathcal{R}_K(\mathbf{D})$ as shown in Algorithm 1. The projection step is detailed in Section 4.

Algorithm 1 $\mathbf{D} = \text{DictionaryUpdate}(\mathbf{Y}, \mathbf{X}, R, \mathbf{n}, \mathbf{m})$

1: Initialize $\mathbf{D}_0$, $t = 0$
2: **while** $\|\mathbf{D}_{t+1} - \mathbf{D}_t\|_F > tol$ **do**
3: $\quad \mathbf{D}_{t+1} = \mathbf{D}_t - \gamma_t(\mathbf{D}_t\mathbf{X} - \mathbf{Y})\mathbf{X}^T$ ▷ Gradient step
4: $\quad \mathbf{D}_{t+1} = \text{HO-SuKroApprox}(\mathbf{D}_{t+1}, R, \mathbf{n}, \mathbf{m})$ ▷ Projection into $\mathcal{C}_K^R$
5: Normalize columns of $\mathbf{D}$

The step-size γ_t is determined with a backtracking line search. For acceleration purposes, the CPD can be initialized with the results of the previous iteration. Finally, note that the column normalization (line 5) can break the imposed structure. This is not a real concern, since the normalization coefficients can be stored separately, implying only m additional products in a matrix-vector operation. Put differently, it is equivalent to storing a dictionary in the form $\mathbf{D}\boldsymbol{\Sigma}$ where $\boldsymbol{\Sigma}$ is a diagonal matrix containing the inverse norm of each column of $\mathbf{D}$.

3 Rearrangement: transforming a K-Kronecker-structured matrix into a low-rank tensor of order K

This section introduces a rearrangement from a matrix to a multidimensional array that links the inference of the Kronecker structure for matrices to the low-rank approximation problem for multidimensional arrays *i.e.* higher-order tensors.

3.1 The second order case

Consider a matrix $\mathbf{D} \in \mathbb{R}^{n_1 n_2 \times m_1 m_2}$ such that $\mathbf{D} = \mathbf{D}_1 \otimes \mathbf{D}_2$, where $\mathbf{D}_1 \in \mathbb{R}^{n_1 \times m_1}$ and $\mathbf{D}_2 \in \mathbb{R}^{n_2 \times m_2}$. Then one can define an operator $\mathcal{R}_2 : \mathbb{R}^{n_1 n_2 \times m_1 m_2} \to \mathbb{R}^{n_2 m_2 \times n_1 m_1}$ which rearranges the elements of $\mathbf{D}$ in such way that the output $\mathbf{D}^\pi = \mathcal{R}_2(\mathbf{D})$ is a rank-1 matrix [18] given by

$$\mathbf{D}^\pi = \mathcal{R}_2(\mathbf{D}) = \mathrm{vec}(\mathbf{D}_2)\,\mathrm{vec}(\mathbf{D}_1)^T = \mathrm{vec}(\mathbf{D}_2) \circ \mathrm{vec}(\mathbf{D}_1) \tag{3}$$

It consists in vectorizing the jth-block in $\mathbf{D}$ to form the jth-column of $\mathcal{R}(\mathbf{D})$, running through the blocks column-wise. This process is illustrated in Figure 1.

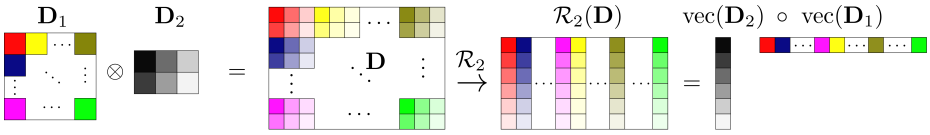

Fig. 1: Rearrangement operation

Since the isomorphism $\mathcal{R}_2$ is linear, when applied to a $(R, 2)$-KS matrix $\mathbf{D} = \sum_{r=1}^{R} \mathbf{D}_1^r \otimes \mathbf{D}_2^r$, it outputs a rank-$R$ matrix $\mathbf{D}^\pi = \sum_{r=1}^{R} \mathrm{vec}(\mathbf{D}_2^r) \circ \mathrm{vec}(\mathbf{D}_1^r)$.

3.2 Generalizing to higher order

Now, suppose a K-KS matrix $\mathbf{D} \in \mathbb{R}^{n_1 n 2 \ldots n_K \times m_1 m 2 \ldots m_K}$ given by

$$\mathbf{D} = \mathbf{D}_1 \otimes \cdots \otimes \mathbf{D}_K = \bigotimes_{k=1}^{K} \mathbf{D}_k \tag{4}$$

with $\mathbf{D}_k \in \mathbb{R}^{n_k \times m_k}$ for $k \in \{1, \ldots, K\}$.

We can generalize the rearrangement operator $\mathcal{R}_2$ (for 2-KS matrices) to an operator $\mathcal{R}_K : \mathbb{R}^{n_1 n 2 \ldots n_K \times m_1 m 2 \ldots m_K} \to \mathbb{R}^{n_K m_K \times \cdots \times n_1 m_1}$ (for K-KS matrices) in a recursive manner. For this, let's suppose $\mathcal{R}_{K-1}$ known, such that

$$\mathbf{D} = \bigotimes_{k=2}^{K} \mathbf{D}_k \qquad \Rightarrow \qquad \underline{\mathbf{D}}^\pi = \mathcal{R}_{K-1}(\mathbf{D}) = \mathrm{vec}(\mathbf{D}_K) \circ \cdots \circ \mathrm{vec}(\mathbf{D}_2), \tag{5}$$

which outputs a $(K-1)$th-order rank-1 tensor for an input $(K-1)$-KS matrix, and try to define $\mathcal{R}_K$ from it.

Note that we can rewrite $\mathbf{D} = \mathbf{D}_1 \otimes (\mathbf{D}_2 \otimes \cdots \otimes \mathbf{D}_K)$ which means that $\mathbf{D}$ is composed of n_1 by m_1 blocks given by $d_1(i,j)\,(\mathbf{D}_2 \otimes \cdots \otimes \mathbf{D}_K)$ in which we can directly apply $\mathcal{R}_{K-1}$. If we again run through all these blocks columnwise applying $\mathcal{R}_{K-1}$ and stacking the resulting $(K-1)$th-order tensors along dimension K, we will obtain a (K)-order rank-1 tensor given by:

$$\underline{\mathbf{D}}^{\pi} = \mathcal{R}_K(\mathbf{D}) = (\mathrm{vec}(\mathbf{D}_K) \circ \cdots \circ \mathrm{vec}(\mathbf{D}_2)) \circ \mathrm{vec}(\mathbf{D}_1)$$

Therefore, we can define a recursive algorithm to calculate $\mathcal{R}_K$ which has $\mathcal{R}_2$ defined in Section 3.1 as a base case. Actually, we can go even further and decompose $\mathcal{R}_2$ recursively in the exact same way, in which case the base case $\mathcal{R}_1$ becomes a simple vectorization operation.The described procedure is illustrated in Figure 2 for the particular case of $K=3$.

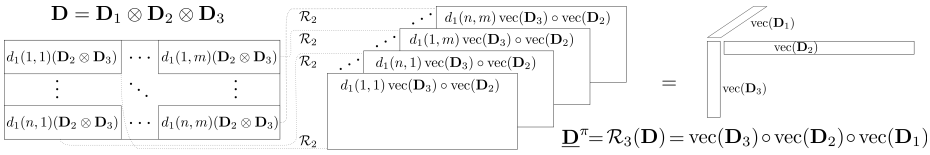

Fig. 2: Rearrangement operation $\mathcal{R}_K$ for $K=3$.

Just like in Section 3.1, note that for an input (R,K)-KS matrix, the rearrangement outputs a sum of R rank-1 tensors with factors $\mathrm{vec}(\mathbf{D}_i)$ sorted in the reverse lexicographic order: $\underline{\mathbf{D}}^{\pi} = \sum_{r=1}^{R} \mathrm{vec}(\mathbf{D}_K) \circ \cdots \circ \mathrm{vec}(\mathbf{D}_1)$.

3.3 Inverse rearrangement

The inverse rearrangement $\mathcal{R}_K^{-1}$ consists simply in switching the input and output indexes of the previous operator so to yield $\mathbf{D} = \mathcal{R}_K^{-1}(\underline{\mathbf{D}}^{\pi})$. In practical terms, the resulting recursive algorithm consists in progressively reconstructing the blocks of $\mathbf{D}$ from their vectorized versions in $\underline{\mathbf{D}}^{\pi}$. The base case is now a matricization (unvec) operation.

4 From SVD to CPD

Let $\mathbf{D} \in \mathbb{R}^{n \times m}$ and consider the following constrained approximation problem:

$$\min_{\hat{\mathbf{D}} \in \mathcal{C}_K^R} \|\hat{\mathbf{D}} - \mathbf{D}\|_F^2 \tag{6}$$

With the help of the rearrangement operators defined in Section 3, the task of approximating any given matrix $\mathbf{D}$ comes down to low-rank approximation of the Kth-order rearranged tensor $\underline{\mathbf{D}}^{\pi} = \mathcal{R}_K(\mathbf{D})$.

When the targeted structure is a (sum of) 2-Kronecker product(s), i.e. $K=2$, we are interested in the low-rank approximation of $\mathbf{D}^\pi = \mathcal{R}_2(\mathbf{D})$ which is still a matrix (2nd-order tensor). This is easily achieved – and also optimally, from Eckart-Young theorem – via the SVD.

The task gets harder if we want to generalize to a (sum of) K-Kronecker product(s), since a low-rank approximation of an order-K tensor $\underline{\mathbf{D}}^\pi = \mathcal{R}_K(\mathbf{D})$ is required. In this work, we propose to use the a Canonical Polyadic Decomposition (CPD) [10] to approximate the tensor $\underline{\mathbf{D}}^\pi$ with a sum of R rank-one tensors.

$$\{\hat{\mathbf{d}}_k^r\}_{k=\{1,\dots,K\}}^{r=\{1,\dots,R\}} = \text{CPD}(\underline{\mathbf{D}}^\pi, R) \quad \text{such that} \quad \hat{\underline{\mathbf{D}}}^\pi = \sum_{r=1}^{R} \hat{\mathbf{d}}_K^r \circ \cdots \circ \hat{\mathbf{d}}_1^r \tag{7}$$

We can see that each composing $\hat{\mathbf{D}}_k^r$ can be obtained from the corresponding vector $\hat{\mathbf{d}}_k^r$ through a simple matricization operation: $\hat{\mathbf{D}}_k^r = \text{unvec}(\hat{\mathbf{d}}_k^r)$. The resulting approximation is thus a (R,K)-KS matrix with factors $\hat{\mathbf{D}}_k^r$.

The proposed procedure to obtain $\hat{\mathbf{D}}$ and its factors $\hat{\mathbf{D}}_k^r$ is summarized in Algorithm 2, which we call HO-SuKro (**H**igher **O**rder **Su**m of **Kro**neckers) Approximation algorithm. It is parametrized by the targeted rank R and two vectors $\mathbf{n}$ and $\mathbf{m}$, such that $n = \prod_{k=1}^K n_k$ and $m = \prod_{k=1}^K m_k$, determining the dimensions of the factors $\hat{\mathbf{D}}_k^r$.

Algorithm 2 $\left[\hat{\mathbf{D}}\ ,\ \{\hat{\mathbf{D}}_k^r\}_{k=\{1,\dots,K\}}^{r=\{1,\dots,R\}}\right] = \text{HO-SuKroApprox}(\mathbf{D}, R, \mathbf{n}, \mathbf{m})$

1: $\mathcal{R}_K(\mathbf{D}) = \text{Rearrange}(\mathbf{D}, \mathbf{n}, \mathbf{m})$ ▷ Rearranging input matrix
2: $\{\hat{\mathbf{d}}_k^r\}_{k=\{1,\dots,K\}}^{r=\{1,\dots,R\}} = \text{CPD}(\mathcal{R}_K(\mathbf{D}), R)$ ▷ CPD on rearranged tensor
3: $\{\hat{\mathbf{D}}_k^r\}_{k=\{1,\dots,K\}}^{r=\{1,\dots,R\}} = \text{unvec}(\hat{\mathbf{d}}_k^r)$ ▷ Recovering factors $\hat{\mathbf{D}}_k^r$
4: **return** $\left[\hat{\mathbf{D}} = \sum_{r=1}^R \hat{\mathbf{D}}_1^r \otimes \cdots \otimes \hat{\mathbf{D}}_K^r\ ,\ \{\hat{\mathbf{D}}_k^r\}_{k=\{1,\dots,K\}}^{r=\{1,\dots,R\}}\right]$

5 Experiments

To evaluate the proposed dictionary learning algorithm, we have performed some color image denoising experiments following the set-up introduced in [6]. The test images, 512×512×3 color images, are corrupted by a white Gaussian noise with different standard deviations σ. In these experiments we thus have $K = 3$.

The dictionary is trained from a set of vectorized (6×6×3)-pixel patches extracted uniformly from the noisy image itself and then used to reconstruct all overlapping patches with a one-pixel step. The denoised image is obtained by averaging the pixel values of the overlapping patches. The simulation parameters are as follows: sample dimension (n) is 108, number of atoms (m) is 864, number of training samples (N) is 40000, convergence tolerance (tol) is $\|\mathbf{D}\|_F \times 10^{-4}$ and number of iterations (N_{it}) is 20.

The sparse coding step is performed by the OMP algorithm. The PSNR of the reconstructed images are evaluated as follows: $\text{PSNR} = 255^2 N_{px} / \left(\sum_{i=1}^{N_{px}} (y_i - \hat{y}_i)^2 \right)$ where 255 is the maximum pixel value, N_{px} is the total number of pixels on the input image, y_i and $\hat{y}_i$ are respectively the i-th pixel value on the input and reconstructed image. The results are averaged over five noise realizations.

Figure 3 shows the denoised image PSNR (in dB) as a function of the number of Kronecker summing terms in the dictionary (i.e. the rank R of the rearranged tensor). We compare our results to an unstructured dictionary with the same size learned by the K-SVD [1] algorithm and to the 3D-ODCT analytic dictionary, which is actually a 3-Kronecker dictionary as well (although not trained from the data). 3D-ODCT is also used for initializing the proposed HO-SuKro[1].

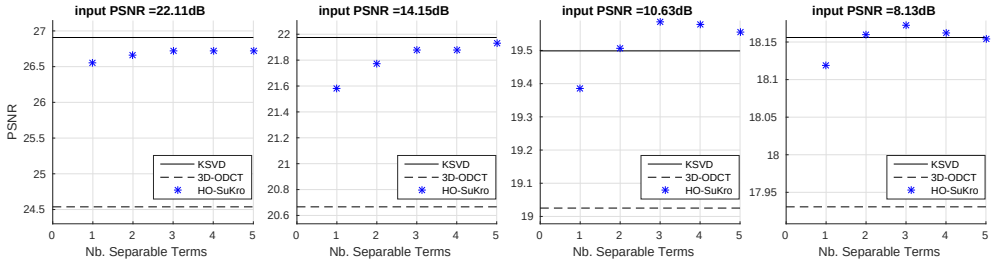

Fig. 3: PSNR vs. Number of separable terms (R) for the mandrill image.

Compared to the fixed 3D-ODCT dictionary, our algorithm achieves considerably better denoising results, even for one single separable term ($R = 1$) which is the exact same structure as the 3D-ODCT. It remains slightly inferior to a K-SVD dictionary in most cases, but with the advantage of the reduced application complexity due to the Kronecker structure (see Table 1). The chosen structure proved well suited to this kind of application, since its introduction compromised very little the performance. Actually, it even enhanced the denoising capabilities in higher noise scenarios, which we attribute to an overfitting prevention due to the structure constraint. The addition of separable terms tends to improve the performance until a certain point after which it saturates – or even deteriorates at higher noise.

Table 1 shows the theoretical complexity savings provided by the Kronecker structure for matrix vector operations as well as its storage cost compared to an unstructured dictionary. The gains are around one and two orders of magnitude in this case. Obviously, for the complexity gains to be observed in practice, the matrix-vector multiplications should be performed according to equation (1).

[1] The 1-D $n \times m$ overcomplete DCT dictionary, as defined in [14], is a cropped version of the orthogonal $m \times m$ DCT matrix. The K-dimensional ODCT is the Kronecker product of K 1-D ODCT.

[2] Complexity cost for HO-SuKro is trivially obtained considering eq. (1). Storage cost is the total number of elements in all factors: $R(\sum_k n_k m_k)$.

Table 1: Complexity costs (for matrix-vector multiplications) and storage costs[2]

	HO-SuKro	Unstructured	Ratio (Unstructured/HO-SuKro)
Complexity (# operations)	$12960 \times R$	186624	$14.4/R$
Storage (# parameters)	$162 \times R$	93312	$576/R$

In Figure 4 we evaluate the robustness of the learning algorithm to a reduction on the training set size. It shows the ΔPSNR, defined as the difference with respect to the PSNR obtained by ODCT.

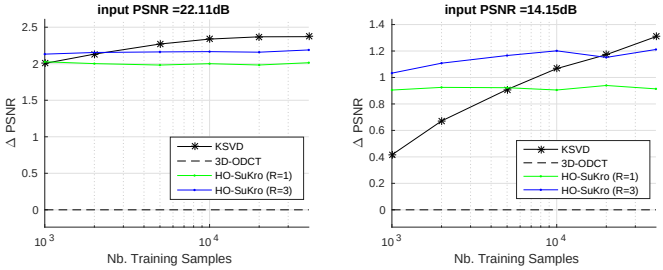

Fig. 4: PSNR vs. Number of training samples for the mandrill image.

Note that the Kronecker-structured dictionaries become more competitive as the size of the training set decreases, to the point of consistently outperforming K-SVD for small enough training sets. This result goes in line with the theoretical results in [15] suggesting a smaller sample complexity for Kronecker-structured dictionaries.

6 Conclusion

To improve on storage, robustness to sample size and computational complexity, a new dictionary learning model was introduced where the dictionary is constrained as a sum of R Kronecker products of K terms. Such a constraint arises naturally when the original data are contained in a Kth-order tensor, such as a collection of color images. We have drawn a parallel between this sum of Kroneckers constraint and the tensor Canonical Polyadic Decomposition, the latter being used as a projection algorithm for imposing the structure constraint. Encouraging results are shown on color image denoising, and future works will study both practical and theoretical implications of the sum of Kronecker constraint on the performance of the dictionary learning algorithm.

References

1. Aharon, M., Elad, M., Bruckstein, A.: K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. IEEE Transactions on Signal Processing 54(11), 4311–4322 (2006)

2. Batselier, K., Wong, N.: A constructive arbitrary-degree kronecker product decomposition of tensors. Numerical Linear Algebra with Applications (2017)
3. Bijma, F., De Munck, J., M Heethaar, R.: The spatiotemporal meg covariance matrix modeled as a sum of kronecker products 27, 402–15 (2005)
4. Caiafa, C.F., Cichocki, A.: Multidimensional compressed sensing and their applications. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 3(6), 355–380 (2013)
5. Dantas, C., da Costa, M.N., Lopes, R.: Learning dictionaries as a sum of kronecker products. IEEE Signal Processing Letters (2017)
6. Elad, M., Aharon, M.: Image denoising via sparse and redundant representations over learned dictionaries. IEEE Transactions on Image Processing 15(12), 3736–3745 (2006)
7. Ghassemi, M., Shakeri, Z., Sarwate, A.D., Bajwa, W.U.: STARK: Structured Dictionary Learning Through Rank-one Tensor Recovery. In: 2017 IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP) (2017)
8. Hawe, S., Seibert, M., Kleinsteuber, M.: Separable dictionary learning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 438–445 (2013)
9. Jung, A., Eldar, Y.C., Grtz, N.: On the minimax risk of dictionary learning. IEEE Transactions on Information Theory 62(3), 1501–1515 (2016)
10. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM review 51(3), 455–500 (2009)
11. Pati, Y.C., Rezaiifar, R., Krishnaprasad, P.S.: Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In: 1993 Conference Record of The 27th Asilomar Conference on Signals, Systems and Computers. pp. 40–44 vol.1 (1993)
12. Peng, Y., Meng, D., Xu, Z., Gao, C., Yang, Y., Zhang, B.: Decomposable nonlocal tensor dictionary learning for multispectral image denoising. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition. pp. 2949–2956 (2014)
13. Roemer, F., Del Galdo, G., Haardt, M.: Tensor-based algorithms for learning multidimensional separable dictionaries. In: 2014 IEEE Conference on International Acoustics, Speech and Signal Processing (ICASSP). pp. 3963–3967. IEEE (2014)
14. Rubinstein, R., Zibulevsky, M., Elad, M.: Double sparsity: Learning sparse dictionaries for sparse signal approximation. IEEE Transactions on Signal Processing 58(3), 1553–1564 (2010)
15. Shakeri, Z., Bajwa, W.U., Sarwate, A.D.: Sample complexity bounds for dictionary learning of tensor data. In: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 4501–4505 (2017)
16. Tadonki, C., Philippe, B.: Parallel numerical linear algebra. chap. Parallel Multiplication of a Vector by a Kronecker Product of Matrices, pp. 71–89. Nova Science Publishers, Inc., Commack, NY, USA (2001)
17. Tsiligkaridis, T., Hero, A.O.: Covariance estimation in high dimensions via kronecker product expansions. IEEE Transactions on Signal Processing 61(21), 5347–5360 (2013)
18. Van Loan, C.F., Pitsianis, N.: Approximation with kronecker products. In: Linear algebra for large scale and real-time applications, pp. 293–314. Springer (1993)
19. Zubair, S., Wang, W.: Tensor dictionary learning with sparse tucker decomposition. In: 2013 18th International Conference on Digital Signal Processing (DSP). pp. 1–6 (2013)