Strategies to learn computationally efficient and compact dictionaries

Luc Le Magoarou, Rémi Gribonval

Strategies to learn computationally efficient and compact dictionaries

Luc Le Magoarou[1] and Rémi Gribonval[1]

[1] INRIA Rennes-Bretagne Atlantique, Campus de Beaulieu, 35042 Rennes, France.

***Abstract*— Dictionary learning is a branch of signal processing and machine learning that aims at expressing some given training data matrix as the multiplication of two factors: one dense matrix called dictionary and one sparse matrix being the representation of the data in the dictionary. The sparser the representation, the better the dictionary. However, manipulating the dictionary as a dense matrix can be computationally costly both in the learning process and later in the usage of this dictionary, thus limiting dictionary learning to relatively small-scale problems. In this paper we consider a general structure of dictionary allowing faster manipulation, and give an algorithm to learn such dictionaries over training data, as well as preliminary results showing the interest of our approach.**

1 Introduction

The use of a dictionary to sparsely represent a certain type of data goes back to almost two centuries with the Fourier transform that was designed to sparsely represent heat flow at that time [6]. The Fourier and the Hadamard transforms, as well as the wavelets to cite just a few rely on a "simple" mathematical formula that has been shown to yield fast algorithms [5, 9].

On the other hand, more recently have arisen algorithms that learn a dictionary directly over training data, without using an analytical formula (see [11] and references therein for a survey on the topic). They consider some data matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$, which is the collection of n training vectors $\mathbf{x}_i \in \mathbb{R}^d$, and can be approximated sparsely in a certain dictionary:

$$\mathbf{X} \approx \mathbf{D}\mathbf{\Gamma}, \tag{1}$$

$\mathbf{\Gamma}$ having sparse columns. Such algorithms provide dictionaries that are well adapted to the data but they do not lead to fast algorithms because of the lack of structure of the dictionary, and storing $\mathbf{D}$ as a dense matrix may be impractical.

Bridging the gap between analytical dictionaries that are not necessarily well adapted to the data at hand but lead to fast algorithms, and learnt dictionaries that are very well adapted to the data but do not lead to fast algorithms is an important challenge. The dictionary learning community started recently to tackle the problem, mainly in [12] and [4], where the authors introduced new dictionary structures. The model we introduce below generalizes these works, as we will see in Sec.2.

In this paper we build on the very simple observation that the fast algorithms used to apply analytical transforms like the FFT or the DWT can be seen as consecutive multiplications of the input vector by sparse matrices. This fact implies that the analytical dictionaries associated with such fast transforms can be expressed as a product of sparse matrices, that is[1]:

$$\mathbf{D} = \prod_{j=1}^{M} \mathbf{S}_j. \tag{2}$$

This structure is precisely what makes possible the appearance of fast algorithms (such as the butterfly FFT for example). Knowing that, it makes sense to learn a dictionary that has this form and is the product of several sparse matrices, making it intrinsically fast and easier to store. The problem at hand is unfortunately highly non-convex and the sparsity constraint is non-smooth, but recent advances in optimization like the algorithm proposed in [2] can be adapted to such problems with convergence guarantees as we will show in the next sections.

[1] The product being taken from left to right: $\prod_{i=1}^{N} \mathbf{A}_i = \mathbf{A}_1 \cdots \mathbf{A}_N$

2 Model and optimization problem

Notation Throughout this paper, matrices are denoted by bold uppercase letters: $\mathbf{A}$. Vectors are denoted by bold lowercase letters: $\mathbf{a}$. The ith column of a matrix $\mathbf{A}$ is denoted by: $\mathbf{a}_i$. Sets are denoted by calligraphical symbols: $\mathcal{A}$.

As stated in the introduction, we propose a new dictionary structure that intrinsically leads to fast algorithms although the dictionary is learnt over some training data. Our goal is to find a dictionary that sparsely represent the data, being itself a multiplication of sparse matrices.

2.1 The matrix factorization point of view

Let $\mathbf{D} \in \mathbb{R}^{d\times a}$ be our dictionary with a atoms and $\mathbf{\Gamma} \in \mathbb{R}^{a\times n}$ the corresponding sparse representation matrix such that $\mathbf{X} \approx \mathbf{D}\mathbf{\Gamma}$. In order to meet the requirements and be intrinsically fast, $\mathbf{D}$ must take the form of equation (2), where the $\mathbf{S}_j$s are sparse matrices in $\mathbb{R}^{a_j \times a_{j+1}}$ with $a_1 = d$ and $a_{M+1} = a$. Now if we say that $\mathbf{\Gamma}$ is now called $\mathbf{S}_{M+1}$ for ease of notation, our goal is to find the sparse factors $\mathbf{S}_j$s such that:

$$\mathbf{X} \approx \prod_{j=1}^{M+1} \mathbf{S}_j. \tag{3}$$

We see with this equation that our structured dictionary learning problem amounts to a factorization of the data matrix into $M+1$ sparse factors. Actually, this model is quite general and can encompass those introduced in [12] and [4] as we will see. Such a multifactor representation of a data matrix has been introduced in the NMF framework in [8], for deep learning in [1, 10] and statistics in [7, 3].

2.2 Optimization objective

The proposed structured dictionary learning problem can be expressed as a constrained optimization problem. In its most general form, it can be stated as follows:

$$\begin{aligned} \underset{\mathbf{S}_j, \forall j \in \{1\cdots M+1\}}{\text{Minimize}} \quad & \left\| \mathbf{X} - \prod_{j=1}^{M+1} \mathbf{S}_j \right\|_F^2 \\ \text{Subject to} \quad & \mathbf{S}_j \in \mathcal{U}_j, \forall j \in \{1 \cdots M+1\}, \end{aligned} \tag{4}$$

where the $\mathcal{U}_j$s are the sets in which each factor should lie.

In [12], the authors propose to constrain each atom of the dictionary to be a sparse linear combination of the atoms of a

so-called *base dictionary*. This base dictionary is assumed to be asociated with a fast algorithm (it takes the form of equation (2)), thus leading to fast manipulation of the whole learnt dictionary. It is actually equivalent to solving the problem (4) with the $M-1$ leftmost factors being known (the base dictionary), $\mathbf{S}_M$ and $\mathbf{S}_{M+1} = \mathbf{\Gamma}$ to estimate, and $\mathcal{U}_M$ and $\mathcal{U}_{M+1}$ denoting sparsity constraints. They give an algorithm to estimate jointly the dictionary and the representation. However, such a structure constrains the dictionary to be close to the base dictionary and thus does not provide full adaptability.

In [4], the authors constrain each atom in the dictionary to be the composition of several circular convolutions with sparse kernels, thus leading again to fast manipulation of the learnt dictionary. They give an algorithm to estimate the dictionary knowing the representation and the support of each kernel. It is actually equivalent to solving the problem (4) with $\mathbf{S}_{M+1}$ being known, $\mathcal{U}_i, \forall i \in \{1 \cdots M\}$ denoting the intersection of sparsity constraints with the set of circulant matrices, and the support of each factor being known.

We propose to handle problem (4) in a more general form, namely we will not assume that any factor is known, and we will consider any $\mathcal{U}_i$s provided that we are able to perform the projection onto it (note that in practice, only the $\mathcal{U}_i$s that are included in the set of sparse matrices will be of interest).

2.3 Algorithm

We propose to apply an algorithm stemming from recent advances in non-convex and non-smooth optimization to problem (4). The algorithm is introduced in [2] and called PALM (Proximal Alternating Linearized Minimization). PALM is made to handle objective functions that take the form:

$$\Psi(\mathbf{x}_1, \ldots, \mathbf{x}_p) := \sum_{i=1}^{p} f_i(\mathbf{x}_i) + H(\mathbf{x}_1, \ldots, \mathbf{x}_p), \qquad (5)$$

where each $\mathbf{x}_i \in \mathbb{R}^{n_i}$ is a block of variables, H is only assumed to be C^1 and the f_is are only assumed to be proper and lower-semicontinuous (possibly indicator functions of constraint sets). The idea behind PALM is to alternate updates between different blocks of variables (Gauss-Seidel), performing a proximal gradient step for each block. PALM is guaranteed to converge to a critical point of the objective function.

Problem (4) is obviously non-convex and thus the algorithm introduced in this paper is not guaranteed to converge toward the global minimum. Adapting PALM to the structured dictionary learning problem we just introduced is quite straightforward. One option is to take each $\mathbf{S}_j$ as a block of variables, so that H is the objective function of problem (4) and the f_is are the indicator functions of the sets $\mathcal{U}_j$. The proximal operator of the indicator function of a set $\mathcal{X}$ reducing to the projection operator onto this set $P_{\mathcal{X}}(.)$, the adaptation of PALM to our problem is given in Algorithm 1, where c_j^i is a stepsize depending on the Lipschitz modulus of the gradient $\nabla_{\mathbf{S}_j} H$ (see [2] for more details).

Algorithm 1 PALM for structured dictionary learning

for $i \in \{1 \cdots Niter\}$ **do**
  for $j \in \{1 \cdots M+1\}$ **do**
    Set $\mathbf{S}_j^{i+1} = P_{\mathcal{U}_j}\left(\mathbf{S}_j^i - \frac{1}{c_j^i} \nabla_{\mathbf{S}_j} H\left(\mathbf{S}_1^{i+1} \ldots \mathbf{S}_j^i \ldots \mathbf{S}_{M+1}^i\right)\right)$
  end for
end for

3 One practical example

Choosing the sets of constraints $\mathcal{U}_j$ is crucial in order to avoid as much as possible local minima, but in order to have a fast dictionary, they have to be subsets of the following (set of k_j-sparse matrices in $\mathbb{R}^{a_j \times a_{j+1}}$):

$$\mathcal{S}_{k_j}^{a_j \times a_{j+1}} = \{\mathbf{U} \in \mathbb{R}^{a_j \times a_{j+1}} : \sum_{i=1}^{a_{j+1}} \|\mathbf{u}_i\|_0^0 \leq k_j\}. \qquad (6)$$

In further work, we intend to set constraints such as the combination of sparsity and orthogonality, or some sort of structured sparsity, because we observed that the factorizations of the most famous analytical dictionaries are of this type.

However in this paper we will consider the simplest constraints, namely we just want the matrices $\mathbf{S}_j$s to have at most k_j nonzero entries. This means that we will have $\mathcal{U}_j = \mathcal{S}_{k_j}^{a_j \times a_{j+1}}$. In this simple configuration, the projection operator $P_{\mathcal{U}_j}(.)$ is simply the hard thresholding operator that sets all but the k_j greatest entries (in absolute value) to zero.

Complexity savings In the setting we just described, the multiplication of a vector by the dictionary or its adjoint (and its storage) would cost $\mathcal{O}(\sum_{j=1}^{M} k_j)$ operations instead of $\mathcal{O}(da)$ operations for a dictionary without structure. The interest here is twofold: first at the learning stage since this algorithm requires multiplications by the dictionary at each iteration, second at the usage stage since it involves again multiplications by the dictionary and its adjoint. It is thus necessary to tune M and the k_js in order to have a significant complexity gain, namely $\sum_{j=1}^{M} k_j \ll da$. For example in the FFT case with $d = a = 2^P$, $P \in \mathbb{N}$, the dictionary is factorized into $M = \log_2 d$ factors that are all $2d$-sparse, thus we perform multiplication in $\mathcal{O}(2d.\log_2 d)$ operations instead of $\mathcal{O}(d^2)$.

Preliminary result We are still testing different settings for the algorithm, but it shows promising results in tasks such as denoising. To illustrate this, we present an example of result with $d = 8$, $a = 20$, $n = 100$, $M = 3$, $k_4 = 100$, $k_1 = k_2 = k_3 = 20$ and $Niter$ = 1000. We generated randomly some data matrix $\mathbf{X}_0$ following the model of equation (3) and we added white Gaussian noise $\mathbf{B}$ with a signal-to-noise ratio of 6dB to obtain $\mathbf{X} = \mathbf{X}_0 + \mathbf{B}$. We ran Algorithm 1 in order to obtain $\hat{\mathbf{X}}_0$. We repeated the experiment 100 times and got an output signal-to-noise ratio of 7.7dB in average. We also made the interesting though expected observation that in approximation task (when we just want to factorize some matrix in sparse factors), there is a trade-off between sparsity of the factors and expressiveness of their product: the more non-zero entries we allow, the better the performances are, but the complexity is also increased.

4 Conclusion

In this abstract, we first presented the dictionary learning problem and the growing concern in the community to make it more computationally efficient. We then introduced a new model that is a generalization of two previously existing ones and that leads to intrinsically fast dictionaries. We presented an algorithm with convergence guarantees to a stationary point that is able to learn such type of dictionary over some training data. We finished by showing briefly some preliminary results of our approach. In the future, we intend to set new configurations for this method. We could for example modify the constraints in order to avoid as much as possible the numerous local minima inherent to the problem.

References

[1] S. Arora, A. Bhaskara, R. Ge, and T. Ma. Provable bounds for learning some deep representations. *CoRR*, abs/1310.6343, 2013.

[2] J. Bolte, S. Sabach, and M. Teboulle. Proximal alternating linearized minimization for nonconvex and nonsmooth problems. *Mathematical Programming*, pages 1–36, 2013.

[3] G. Cao, L. Bachega, and C. Bouman. The sparse matrix transform for covariance estimation and analysis of high dimensional signals. *Image Processing, IEEE Transactions on*, 20(3):625–640, 2011.

[4] O. Chabiron, F. Malgouyres, J.-Y. Tourneret, and N. Dobigeon. Toward Fast Transform Learning. Technical report, Nov. 2013.

[5] J. Cooley and J. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301, 1965.

[6] J. Fourier. *Théorie analytique de la chaleur*. 1822.

[7] A. B. Lee, B. Nadler, and L. Wasserman. Treelets - an adaptive multi-scale basis for sparse unordered data. *The Annals of Applied Statistics*, 2(2):435–471, July 2008.

[8] S. Lyu and X. Wang. On algorithms for sparse multi-factor nmf. In *Advances in Neural Information Processing Systems 26*, pages 602–610. 2013.

[9] S. Mallat. A theory for multiresolution signal decomposition : the wavelet representation. *IEEE Transaction on Pattern Analysis and Machine Intelligence*, 11:674–693, June 1989.

[10] B. Neyshabur and R. Panigrahy. Sparse matrix factorization. *CoRR*, abs/1311.3315, 2013.

[11] R. Rubinstein, A. Bruckstein, and M. Elad. Dictionaries for Sparse Representation Modeling. *Proceedings of the IEEE*, 98(6):1045 –1057, 2010.

[12] R. Rubinstein, M. Zibulevsky, and M. Elad. Double sparsity: learning sparse dictionaries for sparse signal approximation. *IEEE Transactions on Signal Processing*, 58(3):1553–1564, Mar. 2010.