Proximal Methods for Sparse Hierarchical Dictionary Learning

Rodolphe Jenatton[1] RODOLPHE.JENATTON@INRIA.FR
Julien Mairal[1] JULIEN.MAIRAL@INRIA.FR
Guillaume Obozinski GUILLAUME.OBOZINSKI@INRIA.FR
Francis Bach FRANCIS.BACH@INRIA.FR
INRIA - WILLOW Project, Laboratoire d'Informatique de l'Ecole Normale Supérieure (INRIA/ENS/CNRS UMR 8548)
23, avenue d'Italie, 75214 Paris. France

Abstract

We propose to combine two approaches for modeling data admitting sparse representations: on the one hand, dictionary learning has proven effective for various signal processing tasks. On the other hand, recent work on structured sparsity provides a natural framework for modeling dependencies between dictionary elements. We thus consider a tree-structured sparse regularization to learn dictionaries embedded in a hierarchy. The involved proximal operator is computable exactly via a primal-dual method, allowing the use of accelerated gradient techniques. Experiments show that for natural image patches, learned dictionary elements organize themselves in such a hierarchical structure, leading to an improved performance for restoration tasks. When applied to text documents, our method learns hierarchies of topics, thus providing a competitive alternative to probabilistic topic models.

1. Introduction

Learned sparse representations, initially introduced by Olshausen & Field (1997), have been the focus of much research in machine learning and signal processing, leading notably to state-of-the-art algorithms for several problems in image processing (Elad & Aharon, 2006). Modeling signals as a linear combination of a few "basis" vectors offers more flexibility than decompositions based on principal component analysis and its variants. Indeed, sparsity allows for overcomplete dictionaries, whose number of basis vectors are greater than the original signal dimension.

As far as we know, while much attention has been given to efficiently solving the corresponding optimization problem (Lee et al., 2007; Mairal et al., 2010), there are few attempts in the literature to make the model richer by adding structure between dictionary elements (Bengio et al., 2009; Kavukcuoglu et al., 2009). We propose to use recent work on structured sparsity (Zhao et al., 2009; Jenatton et al., 2009; Kim & Xing, 2009) to embed the dictionary elements in a hierarchy.

Hierarchies of latent variables, typically used in neural networks and deep learning architectures (see Bengio, 2009 and references therein) have emerged as a natural structure in several applications, notably to model text documents. Indeed, in the context of *topic models* (Blei et al., 2003), hierarchical models using Bayesian non-parametric methods have been proposed by Blei et al. (2010). Quite recently, hierarchies have also been considered in the context of kernel methods (Bach, 2009). Structured sparsity has been used to regularize dictionary elements by Jenatton et al. (2010), but to the best of our knowledge, it has never been used to model dependencies between them.

This paper makes three contributions:

- We propose to use a structured sparse regularization to learn a dictionary embedded in a tree.
- We show that the proximal operator for a tree-structured sparse regularization can be computed exactly in a finite number of operations using a primal-dual approach, with a complexity linear, or close to linear, in the number of variables. Accelerated gradient methods (e.g., Nesterov, 2007) can then be applied to solve tree-structured sparse decomposition problems, which may be useful in other uses of tree-structured norms (Kim & Xing, 2009; Bach, 2009).
- Our method establishes a bridge between *dictionary learning for sparse coding* and *hierarchical topic models* (Blei et al., 2010), which builds upon the interpretation of topic models as multinomial PCA (Buntine, 2002), and can learn similar hierarchies of topics. See Section 5 for a discussion.

[1]Contributed equally.

Appearing in *Proceedings of the 27^{th} International Conference on Machine Learning*, Haifa, Israel, 2010.

2. Problem Statement

2.1. Dictionary Learning

Let us consider a set $\mathbf{X} = [\mathbf{x}^1, \ldots, \mathbf{x}^n] \in \mathbb{R}^{m \times n}$ of n signals of dimension m. Dictionary learning is a matrix factorization problem that aims to represent these signals as linear combinations of *dictionary elements*, denoted here by the columns of a matrix $\mathbf{D} = [\mathbf{d}^1, \ldots, \mathbf{d}^p] \in \mathbb{R}^{m \times p}$. More precisely, the dictionary $\mathbf{D}$ is learned along with a matrix of *decomposition coefficients* $\mathbf{A} = [\boldsymbol{\alpha}^1, \ldots, \boldsymbol{\alpha}^n] \in \mathbb{R}^{p \times n}$, so that $\mathbf{x}^i \approx \mathbf{D}\boldsymbol{\alpha}^i$ for every signal $\mathbf{x}^i$, as measured by any convex loss, e.g., the square loss in this paper.

While learning simultaneously $\mathbf{D}$ and $\mathbf{A}$, one may want to encode specific prior knowledge about the task at hand, such as, for example, the positivity of the decomposition (Lee & Seung, 1999), or the sparsity of $\mathbf{A}$ (Olshausen & Field, 1997; Lee et al., 2007; Mairal et al., 2010). This leads to penalizing or constraining $(\mathbf{D}, \mathbf{A})$ and results in the following formulation:

$$\min_{\mathbf{D} \in \mathcal{D}, \mathbf{A} \in \mathcal{A}} \frac{1}{n} \sum_{i=1}^{n} \Big[\frac{1}{2} ||\mathbf{x}^i - \mathbf{D}\boldsymbol{\alpha}^i||_2^2 + \lambda \Omega(\boldsymbol{\alpha}^i) \Big], \qquad (1)$$

where $\mathcal{A}$ and $\mathcal{D}$ denote two convex sets and Ω is a regularization term, usually a norm, whose effect is controlled by the regularization parameter $\lambda > 0$. Note that $\mathcal{D}$ is assumed to be bounded to avoid any degenerate solutions of Eq. (1). For instance, the standard *sparse coding* formulation takes Ω to be the ℓ_1 norm, $\mathcal{D}$ to be the set of matrices in $\mathbb{R}^{m \times p}$ whose columns are in the unit ball of the ℓ_2 norm, with $\mathcal{A} = \mathbb{R}^{p \times n}$ (Lee et al., 2007; Mairal et al., 2010).

However, this classical setting treats each dictionary element independently from the others, and does not exploit possible relationships between them. We address this potential limitation of the ℓ_1 norm by embedding the dictionary in a tree structure, through a hierarchical norm introduced by Zhao et al. (2009) and Bach (2009), which we now present.

2.2. Hierarchical Sparsity-Inducing Norms

We organize the dictionary elements in a rooted-tree $\mathcal{T}$ composed of p nodes, one for each dictionary element $\mathbf{d}^j$, $j \in \{1, \ldots, p\}$. We will identify these indices j in $\{1, \ldots, p\}$ and the nodes of $\mathcal{T}$. We want to exploit the structure of $\mathcal{T}$ in the following sense: the decomposition of any vector $\mathbf{x}$ can involve a dictionary element $\mathbf{d}^j$ *only if the ancestors of* $\mathbf{d}^j$ *in* $\mathcal{T}$ *are themselves part of the decomposition*. Equivalently, one can say that when a dictionary element $\mathbf{d}^j$ is not involved in the decomposition of a vector $\mathbf{x}$ then *its descendants in* $\mathcal{T}$ *should not be part of the decomposition*. While these two views are equivalent, the latter leads to an intuitive penalization term.

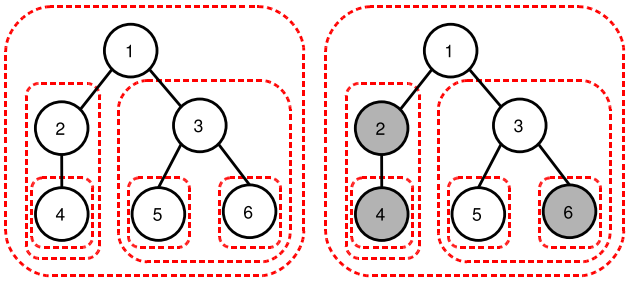

Figure 1. Left: example of a tree-structured set of groups $\mathcal{G}$ (dashed contours in red), corresponding to a tree $\mathcal{T} = \{1, \ldots, 6\}$ (in black). Right, example of a sparsity pattern: the groups $\{2, 4\}, \{4\}$ and $\{6\}$ are set to zero, so that the corresponding nodes (in gray) that form subtrees of $\mathcal{T}$ are removed. The remaining nonzero variables $\{1, 3, 5\}$ are such that, if a node is selected, the same goes for all its ancestors.

To obtain models with the desired property, one considers for all j in $\mathcal{T}$, the group $g_j \subseteq \{1, \ldots, p\}$ of dictionary elements that only contains j and all its descendants, and penalizes the number of such groups that are involved in the decomposition of $\mathbf{x}$ (a group being "involved in the decomposition" meaning here that at least one of its dictionary element is part of the decomposition). We call $\mathcal{G}$ this set of groups (Figure 1).

While this penalization is non-convex, a convex proxy has been introduced by Zhao et al. (2009) and was further considered by Bach (2009) and Kim & Xing (2009) in the context of regression. For any vector $\boldsymbol{\alpha} \in \mathbb{R}^p$, let us define

$$\Omega(\boldsymbol{\alpha}) \triangleq \sum_{g \in \mathcal{G}} w_g \|\boldsymbol{\alpha}_{|g}\|,$$

where $\boldsymbol{\alpha}_{|g}$ is the vector of size p whose coordinates are equal to those of $\boldsymbol{\alpha}$ for indices in the set g, and 0 otherwise[2]. $\|.\|$ stands either for the ℓ_∞ or ℓ_2 norm, and $(w_g)_{g \in \mathcal{G}}$ denotes some positive weights[3]. As analyzed by Zhao et al. (2009), when penalizing by Ω, some of the vectors $\boldsymbol{\alpha}_{|g}$ are set to zero for some $g \in \mathcal{G}$. Therefore, the components of $\boldsymbol{\alpha}$ corresponding to some entire subtrees of $\mathcal{T}$ are set to zero, which is exactly the desired effect (Figure 1).

Note that even though we have presented for simplicity reasons this hierarchical norm in the context of a single tree with a single element at each node, it can be extended easily to the case of forests of trees, and/or trees containing several dictionary elements at each node. More generally, this formulation can be extended with the notion of *tree-structured* groups, which we now present.

[2] Note the difference with the notation $\boldsymbol{\alpha}_g$, which is often used in works on structured sparsity, where $\boldsymbol{\alpha}_g$ is a vector of size $|g|$.

[3] For a complete definition of Ω for any ℓ_q norm, a discussion of the choice of q, and a strategy for choosing the weights w_g, see (Zhao et al., 2009; Kim & Xing, 2009).

Definition 1 (Tree-structured set of groups.) *A set of groups $\mathcal{G} = \{g\}_{g\in\mathcal{G}}$ is said to be tree-structured in $\{1,\ldots,p\}$, if $\bigcup_{g\in\mathcal{G}} g = \{1,\ldots,p\}$ and for all $g,h \in \mathcal{G}$, $(g\cap h \neq \emptyset) \Rightarrow (g \subseteq h \text{ or } h \subseteq g)$. For such a set of groups, there exists a (non-unique) total order relation $\preceq$ such that:*

$$g \preceq h \;\Rightarrow\; \big\{g \subseteq h \quad or \quad g\cap h = \emptyset\big\}.$$

Sparse hierarchical norms having been introduced, we now address the optimization dealing with such norms.

3. Optimization

Optimization for dictionary learning has already been intensively studied, and a typical scheme alternating between the variables $\mathbf{D}$ and $\mathbf{A} = [\boldsymbol{\alpha}^1,\ldots,\boldsymbol{\alpha}^n]$, i.e., minimizing over one while keeping the other one fixed, yields good results in general (Lee et al., 2007). The main difficulty of our problem lies essentially in the optimization of the vectors $\boldsymbol{\alpha}^i$, $i \in \{1,\ldots,n\}$ for $\mathbf{D}$ fixed, since n may be large, and since it requires to deal with the nonsmooth regularization term Ω. The optimization of the dictionary $\mathbf{D}$ (for $\mathbf{A}$ fixed) is in general easier, as discussed in Section 3.5.

Within the context of regression, several optimization methods to cope with Ω have already been proposed. A boosting-like technique with a path-following strategy is used by Zhao et al. (2009). Kim & Xing (2009) uses a reweighted least-squares scheme when $\|.\|$ is the ℓ_2 norm. The same approach is considered by Bach (2009), but built upon an active set strategy. In this paper, we propose to perform the updates of the vectors $\boldsymbol{\alpha}^i$ based on a proximal approach which we now introduce.

3.1. Proximal Operator for the Norm Ω

Proximal methods have drawn increasing attention in the machine learning community (e.g., Ji & Ye, 2009 and references therein), especially because of their convergence rates (optimal for the class of first-order techniques) and their ability to deal with large nonsmooth convex problems (e.g., Nesterov, 2007; Beck & Teboulle, 2009). In a nutshell, these methods can be seen as a natural extension of gradient-based techniques when the objective function to minimize has a nonsmooth part. In our context, when the dictionary $\mathbf{D}$ is fixed and $\mathcal{A} = \mathbb{R}^p$, we minimize for each signal $\mathbf{x}$ the following convex nonsmooth objective function w.r.t. $\boldsymbol{\alpha} \in \mathbb{R}^p$:

$$f(\boldsymbol{\alpha}) + \lambda\Omega(\boldsymbol{\alpha}),$$

where $f(\boldsymbol{\alpha})$ stands for the data-fitting term $\frac{1}{2}\|\mathbf{x} - \mathbf{D}\boldsymbol{\alpha}\|_2^2$. At each iteration of the proximal algorithm, f is linearized around the current estimate $\hat{\boldsymbol{\alpha}}$, and the current value of $\boldsymbol{\alpha}$ is updated as the solution of the *proximal* problem:

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^p} f(\hat{\boldsymbol{\alpha}}) + (\boldsymbol{\alpha}-\hat{\boldsymbol{\alpha}})^\top \nabla f(\hat{\boldsymbol{\alpha}}) + \lambda\Omega(\boldsymbol{\alpha}) + \tfrac{L}{2}\|\boldsymbol{\alpha}-\hat{\boldsymbol{\alpha}}\|_2^2.$$

The quadratic term keeps the update in a neighborhood where f is close to its linear approximation, and $L > 0$ is a parameter. This problem can be rewritten as,

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^p} \frac{1}{2}\|\boldsymbol{\alpha} - \big(\hat{\boldsymbol{\alpha}} - \tfrac{1}{L}\nabla f(\hat{\boldsymbol{\alpha}})\big)\|_2^2 + \tfrac{\lambda}{L}\Omega(\boldsymbol{\alpha}).$$

Solving *efficiently* and *exactly* this problem is crucial to enjoy the fast convergence rates of proximal methods. In addition, when the nonsmooth term Ω is not present, the previous proximal problem exactly leads to the standard gradient update rule. More generally, the *proximal operator* associated with our regularization term $\lambda\Omega$, is the function that maps a vector $\mathbf{u} \in \mathbb{R}^p$ to the (unique) solution of

$$\min_{\mathbf{v}\in\mathbb{R}^p} \frac{1}{2}\|\mathbf{u}-\mathbf{v}\|_2^2 + \lambda\Omega(\mathbf{v}). \tag{2}$$

In the simpler setting where $\mathcal{G}$ is the set of singletons, Ω is the ℓ_1 norm, and the proximal operator is the (elementwise) soft-thresholding operator $\mathbf{u}_j \mapsto \text{sign}(\mathbf{u}_j)\max(|\mathbf{u}_j| - \lambda, 0)$, $j \in \{1,\ldots,p\}$. Similarly, when the groups in $\mathcal{G}$ form a *partition* of the set of variables, we have a group Lasso like penalty, and the proximal operator can be computed in closed-form (see Bengio et al., 2009 and references therein). This is a priori not possible anymore as soon as some groups in $\mathcal{G}$ overlap, which is always the case in our hierarchical setting with tree-structured groups.

3.2. Primal-Dual Interpretation

We now show that Eq. (2) can be solved with a primal-dual approach. The procedure solves a dual formulation of Eq. (2) involving the dual norm[4] of $\|.\|$, denoted by $\|.\|_*$, and defined by $\|\boldsymbol{\kappa}\|_* = \max_{\|\mathbf{z}\|\leq 1} \mathbf{z}^\top\boldsymbol{\kappa}$ for any vector $\boldsymbol{\kappa}$ in $\mathbb{R}^p$. The formulation is described in the following lemma that relies on conic duality (Boyd & Vandenberghe, 2004):

Lemma 1 (Dual of the proximal problem) *Let $\mathbf{u} \in \mathbb{R}^p$ and let us consider the problem*

$$\max_{\boldsymbol{\xi}\in\mathbb{R}^{p\times|\mathcal{G}|}} -\frac{1}{2}\big(\|\mathbf{u} - \sum_{g\in\mathcal{G}}\boldsymbol{\xi}^g\|_2^2 - \|\mathbf{u}\|_2^2\big) \quad \text{s.t. } \forall g\in\mathcal{G},\ \|\boldsymbol{\xi}^g\|_* \leq \lambda w_g \text{ and } \boldsymbol{\xi}^g_j = 0 \text{ if } j\notin g, \tag{3}$$

where $\boldsymbol{\xi} = (\boldsymbol{\xi}^g)_{g\in\mathcal{G}}$ and $\boldsymbol{\xi}^g_j$ denotes the j-th coordinate of the vector $\boldsymbol{\xi}^g$ in $\mathbb{R}^p$. Then, problems (2) and (3) are dual to each other and strong duality holds. In addition, the pair of primal-dual variables $\{\mathbf{v},\boldsymbol{\xi}\}$ is optimal if and only if $\boldsymbol{\xi}$ is a feasible point of the optimization problem (3), and

$$\mathbf{v} = \mathbf{u} - \textstyle\sum_{g\in\mathcal{G}}\boldsymbol{\xi}^g,$$
$$\forall g\in\mathcal{G},\quad \begin{cases} \mathbf{v}_{|g}^\top\boldsymbol{\xi}^g = \|\mathbf{v}_{|g}\|\,\|\boldsymbol{\xi}^g\|_* \quad \text{and} \quad \|\boldsymbol{\xi}^g\|_* = \lambda w_g, \\ \text{or } \mathbf{v}_{|g} = 0. \end{cases}$$

[4] It is easy to show that the dual norm of the ℓ_2 norm is the ℓ_2 norm itself. The dual norm of the ℓ_∞ is the ℓ_1 norm.

For space limitation reasons, we have omitted all the detailed proofs from this section. They will be available in a longer version of this paper. Note that we focus here on specific tree-structured groups, but the previous lemma is valid regardless of the nature of $\mathcal{G}$.

The structure of the dual problem of Eq. (3), i.e., the separability of the (convex) constraints for each vector $\boldsymbol{\xi}^g$, $g \in \mathcal{G}$, makes it possible to use block coordinate ascent (Bertsekas, 1999). Such a procedure is presented in Algorithm 1. It optimizes sequentially Eq. (3) with respect to the variable $\boldsymbol{\xi}^g$, while keeping fixed the other variables $\boldsymbol{\xi}^h$, for $h \neq g$. It is easy to see from Eq. (3) that such an update for a group g in $\mathcal{G}$ amounts to the orthogonal projection of the vector $\mathbf{u}_{|g} - \sum_{h \neq g} \xi^h_{|g}$ onto the ball of radius λw_g of the dual norm $\|.\|_*$. We denote this projection $\Pi^*_{\lambda w_g}$.

Algorithm 1 Block coordinate ascent in the dual

Inputs: $\mathbf{u} \in \mathbb{R}^p$ and set of groups $\mathcal{G}$.
Outputs: $(\mathbf{v}, \boldsymbol{\xi})$ (primal-dual solutions).
Initialization: $\mathbf{v} = \mathbf{u}$, $\boldsymbol{\xi} = 0$.
while (*maximum number of iterations not reached*) **do**
  for $g \in \mathcal{G}$ **do**
    $\mathbf{v} \leftarrow \mathbf{u} - \sum_{h \neq g} \boldsymbol{\xi}^h$.
    $\boldsymbol{\xi}^g \leftarrow \Pi^*_{\lambda w_g}(\mathbf{v}_{|g})$.
  end for
end while
$\mathbf{v} \leftarrow \mathbf{u} - \sum_{g \in \mathcal{G}} \boldsymbol{\xi}^g$.

3.3. Convergence in One Pass

In general, Algorithm 1 is not guaranteed to solve exactly Eq. (2) in a finite number of iterations. However, when $\|.\|$ is the ℓ_2 or ℓ_∞ norm, and provided that the groups in $\mathcal{G}$ are appropriately ordered, we now prove that only *one pass* of Algorithm 1, i.e., only one iteration over all groups, is sufficient to obtain the exact solution of Eq. (2). This result constitutes the main technical contribution of the paper.

Before stating this result, we need to introduce a key lemma that shows that, given two nested groups g, h such that $g \subseteq h \subseteq \{1, \ldots, p\}$, if $\boldsymbol{\xi}^g$ is updated before $\boldsymbol{\xi}^h$ in Algorithm 1, then the optimality condition for $\boldsymbol{\xi}^g$ is not perturbed by the update of $\boldsymbol{\xi}^h$.

Lemma 2 (Projections with nested groups)
Let $\|.\|$ denote either the ℓ_2 or ℓ_∞ norm, and g and h be two nested groups—that is, $g \subseteq h \subseteq \{1, \ldots, p\}$. Let $\mathbf{v}$ be a vector in $\mathbb{R}^p$, and let us consider the successive projections

$$\boldsymbol{\kappa}^g \triangleq \Pi^*_{t_g}(\mathbf{v}_{|g}) \text{ and } \boldsymbol{\kappa}^h \triangleq \Pi^*_{t_h}(\mathbf{v}_{|h} - \boldsymbol{\kappa}^g)$$

with $t_g, t_h > 0$. Then, we have as well $\boldsymbol{\kappa}^g = \Pi^*_{t_g}(\mathbf{v}_{|g} - \boldsymbol{\kappa}^h_{|g})$.

The previous lemma establishes the convergence in one pass of Algorithm 1 in the case where $\mathcal{G}$ contains only two nested groups $g \subseteq h$, provided that $\boldsymbol{\xi}^g$ is computed before $\boldsymbol{\xi}^h$. In the following proposition, this lemma is extended to general tree-structured sets of groups $\mathcal{G}$:

Proposition 1 (Convergence in one pass) *Suppose that the groups in $\mathcal{G}$ are ordered according to $\preceq$ and that the norm $\|.\|$ is either the ℓ_2 or ℓ_∞ norm[5]. Then, after initializing $\boldsymbol{\xi}$ to 0, **one pass** of Algorithm 1 with the order $\preceq$ gives the solution of Eq. (2).*

Proof sketch. The proof relies on Lemma 2. We proceed by induction, by showing that we keep the optimality conditions of Eq. (3) satisfied after each update in Algorithm 1. The induction is initialized by the leaves. Once the induction reaches the last group, i.e., after one complete pass over $\mathcal{G}$, the dual variable $\boldsymbol{\xi}$ satisfies the optimality conditions for Eq. (3), which implies that $\{\mathbf{v}, \boldsymbol{\xi}\}$ is optimal. Since strong duality holds, $\mathbf{v}$ is the solution of Eq. (2). ■

3.4. Efficient Computation of the Proximal Operator

Since one pass of Algorithm 1 involves $|\mathcal{G}| = p$ projections onto the ball of the dual norm (respectively the ℓ_2 and the ℓ_1 norms) of vectors in $\mathbb{R}^p$, a naive implementation leads to a complexity in $O(p^2)$, since each of these projections can be obtained in $O(p)$ operations (see Mairal et al., 2010, and references therein). However, the primal solution $\mathbf{v}$, which is the quantity of interest, can be obtained with a better complexity, as exposed below:

Proposition 2 (Complexity of the procedure)
i) For the ℓ_2 norm, the primal solution $\mathbf{v}$ of Algorithm 1 can be obtained in $O(p)$ operations.
ii) For the ℓ_∞ norm, $\mathbf{v}$ can be obtained in $O(pd)$ operations, where d is the depth of the tree.

The linear complexity in the ℓ_2 norm case results from a recursive implementation. It exploits the fact that each projection amounts to a scaling, whose factor can be found without explicitly performing the full projection at each iteration. As for the ℓ_∞ norm, since all the groups at a depth $k \in \{1, \ldots, d\}$ do not overlap, the cost for performing all the projections at this depth k is $O(p)$, which leads to a total complexity of $O(dp)$. Note that d could depend on p as well. For instance, in an unbalanced case, the worse case could be $d = O(p)$, in a balanced tree, one could have $d = O(\log(p))$. In practice, the structures we have considered are relatively flat, with a depth not exceeding $d = 5$. Details will be provided in a longer version of this paper.

3.5. Learning the Dictionary

We alternate between the updates of $\mathbf{D}$ and $\mathbf{A}$, minimizing over one while keeping the other variable fixed.

[5] Interestingly, we have observed that this was not true in general when $\|.\|$ is an ℓ_q norm, for $q \neq 2$ and $q \neq \infty$.

Updating D. We have chosen to follow the matrix-inversion free procedure of Mairal et al. (2010) for updating the dictionary. This method consists in a block-coordinate scheme over the columns of $\mathbf{D}$. Specifically, we assume that the domain set $\mathcal{D}$ has the form

$$\mathcal{D}_\mu \triangleq \{\mathbf{D} \in \mathbb{R}^{m\times p},\ \mu\|\mathbf{d}^j\|_1 + (1-\mu)\|\mathbf{d}^j\|_2^2 \leq 1\}, \quad (4)$$

or $\mathcal{D}_\mu^+ \triangleq \mathcal{D}_\mu \cap \mathbb{R}_+^{m\times p}$, with $\mu \in [0,1]$. The choice for these particular domain sets is motivated by the experiments of Section 4. For natural image patches, the dictionary elements are usually constrained to be in the unit ℓ_2 norm ball (i.e., $\mathcal{D} = \mathcal{D}_0$), while for topic modeling, the dictionary elements are distributions of words and therefore belong to the simplex (i.e., $\mathcal{D} = \mathcal{D}_1^+$). The update of each dictionary element amounts to performing a Euclidean projection, which can be computed efficiently (Mairal et al., 2010). Concerning the stopping criterion, we follow the strategy from the same authors and go over the columns of $\mathbf{D}$ only a few times, typically 5 in our experiments.

Updating the vectors $\boldsymbol{\alpha}^i$. The procedure for updating the columns of $\mathbf{A}$ is built upon the results derived in Section 3.2. We have shown that the proximal operator from Eq. (2) can be computed exactly and efficiently. It makes it possible to use fast proximal techniques, suited to non-smooth convex optimization.

Specifically, we have tried the accelerated scheme from both Nesterov (2007) and Beck & Teboulle (2009), and finally opted for the latter since, for a comparable level of precision, fewer calls of the proximal operator are required. The procedure from Beck & Teboulle (2009) basically follows Section 3.1, except that the proximal operator is not directly applied on the current estimate, but on an auxiliary sequence of points that linearly combines past estimates. This algorithm has an optimal convergence rate in the class of first-order techniques, and also allows warm restarts, which is crucial in our alternating scheme. Furthermore, positivity constraints can be added on the domain of $\mathbf{A}$, by noticing that for our norm Ω and any $\mathbf{u} \in \mathbb{R}^p$, adding these constraints when computing the proximal operator is equivalent to solving

$$\min_{\mathbf{v}\in\mathbb{R}^p} \frac{1}{2}\|[\mathbf{u}]_+ - \mathbf{v}\|_2^2 + \lambda\Omega(\mathbf{v}),$$

with $([\mathbf{u}]_+)_j \triangleq \max\{\mathbf{u}_j, 0\}$. We will indeed use positive decompositions to model text corpora in Section 4.

Finally, we monitor the convergence of the algorithm by checking the relative decrease in the cost function. We also investigated the derivation of a duality gap, but this implies the computation of the dual norm Ω^* for which no closed-form is available; computing approximations of Ω^* based on bounds from Jenatton et al. (2009) turned out to be too slow for our experiments.

4. Experiments

4.1. Natural Image Patches

This experiment studies whether a hierarchical structure can help dictionaries for denoising natural image patches, and in which noise regime the potential gain is significant. We aim at reconstructing *corrupted* patches from a test set, after having learned dictionaries on a training set of *non-corrupted* patches. Though not typical in machine learning, this setting is reasonable in the context of images, where lots of non-corrupted patches are easily available.[6]

We have extracted $100,000$ patches of size $m = 8\times 8$ pixels from the Berkeley segmentation database of natural images[7], which contains a high variability of scenes. We have then split this dataset into a training set $\mathbf{X}_{tr}$, a validation set $\mathbf{X}_{val}$, and a test set $\mathbf{X}_{te}$, respectively of size $50,000$, $25,000$, and $25,000$ patches. All the patches are centered and normalized to have unit ℓ_2 norm.

For the first experiment, the dictionary $\mathbf{D}$ is learned on $\mathbf{X}_{tr}$ using the formulation of Eq. (1), with $\mu = 0$ for $\mathcal{D}_\mu$ defined in Eq. (4). The validation and test sets are corrupted by removing a certain percentage of pixels, the task being to reconstruct the missing pixels from the known pixels. We thus introduce for each element $\mathbf{x}$ of the validation/test set, a vector $\tilde{\mathbf{x}}$, equal to $\mathbf{x}$ for the known pixel values and 0 otherwise. In the same way, we define $\tilde{\mathbf{D}}$ as the matrix equal to $\mathbf{D}$, except for the rows corresponding to missing pixel values, which are set to 0. By decomposing $\tilde{\mathbf{x}}$ on $\tilde{\mathbf{D}}$, we obtain a sparse code $\boldsymbol{\alpha}$, and the estimate of the reconstructed patch is defined as $\mathbf{D}\boldsymbol{\alpha}$. Note that this procedure assumes that we know which pixel is missing and which is not for every element $\mathbf{x}$.

The parameters of the experiment are the regularization parameter λ_{tr} used during the train step, the regularization parameter λ_{te} used during the validation/test step, and the structure of the tree. For every reported result, these parameters have been selected by taking the ones offering the best performance on the *validation* set, before reporting any result from the *test* set. The values for the regularization parameters $\lambda_{tr}, \lambda_{te}$ were tested on a logarithmic scale $\{2^{-10}, 2^{-9}, \ldots, 2^2\}$, and then further refined on a finer logarithmic scale of factor $2^{-1/4}$. For simplicity reasons, we have chosen arbitrarily to use the ℓ_∞-norm in the structured norm Ω, with all the weights equal to one. We have tested 21 balanced tree structures of depth 3 and 4, with different *branching factors* $p_1, p_2, \ldots, p_{d-1}$, where d is the depth of the tree and p_k, $k \in \{1, \ldots, d-1\}$

[6] Note that we study the ability of the model to reconstruct independent patches, and additional work is required to apply our framework to a full image processing task, where patches usually overlap (Elad & Aharon, 2006).

[7] www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/

Table 1. Quantitative results of the reconstruction task on natural image patches. First row: percentage of missing pixels. Second and third rows: mean square error multiplied by 100, respectively for classical sparse coding, and tree-structured sparse coding.

noise	50 %	60 %	70 %	80 %	90 %
flat	19.3 ± 0.1	26.8 ± 0.1	36.7 ± 0.1	50.6 ± 0.0	72.1 ± 0.0
tree	18.6 ± 0.1	25.7 ± 0.1	35.0 ± 0.1	48.0 ± 0.0	65.9 ± 0.3

is the number of children for the nodes at depth k. The branching factors tested for the trees of depth 3 where $p_1 \in \{5, 10, 20, 40, 60, 80, 100\}$, $p_2 \in \{2, 3\}$, and for trees of depth 4, $p_1 \in \{5, 10, 20, 40\}$, $p_2 \in \{2, 3\}$ and $p_3 = 2$, giving 21 possible structures associated with dictionaries with at most 401 elements. For each tree structure, we evaluated the performance obtained with the tree-structured dictionary along with the non-structured dictionary containing the same number of elements. These experiments were carried out four times, each time with a different initialization, and with a different noise realization. Quantitative results are reported on Table 1. For every number of missing pixels, the tree-structured dictionary outperforms the "unstructured one", and the most significant improvement is obtained in the noisiest setting. Note that having more dictionary elements is worthwhile when using the tree structure. To study the influence of the chosen structure, we have reported on Figure 2 the results obtained by the 14 tested structures of depth 3, along with those obtained with the unstructured dictionaries containing the same number of elements, when 90% of the pixels are missing. For every number of dictionary elements, the tree-structured dictionary significantly outperforms the unstructured ones. An example of a learned tree-structured dictionary is presented on Figure 3. Dictionary elements naturally organize in groups of patches, with often low frequencies near the root of the tree, and high frequencies near the leaves. Dictionary elements tend to be highly correlated with their parents.

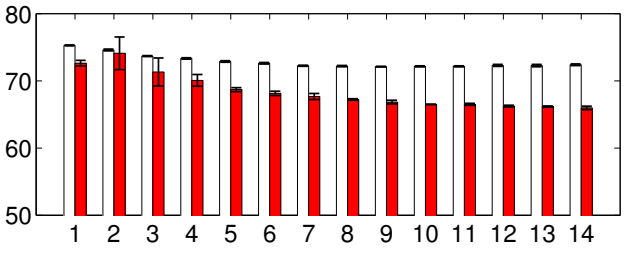

Figure 2. Mean square error multiplied by 100 obtained with 14 structures with error bars, sorted by number of dictionary elements. Red plain bars represents the tree-structured dictionaries. White bars correspond to the flat dictionary model containing the same number of dictionary as the tree-structured one. For readability purpose, the y-axis of the graph starts at 50.

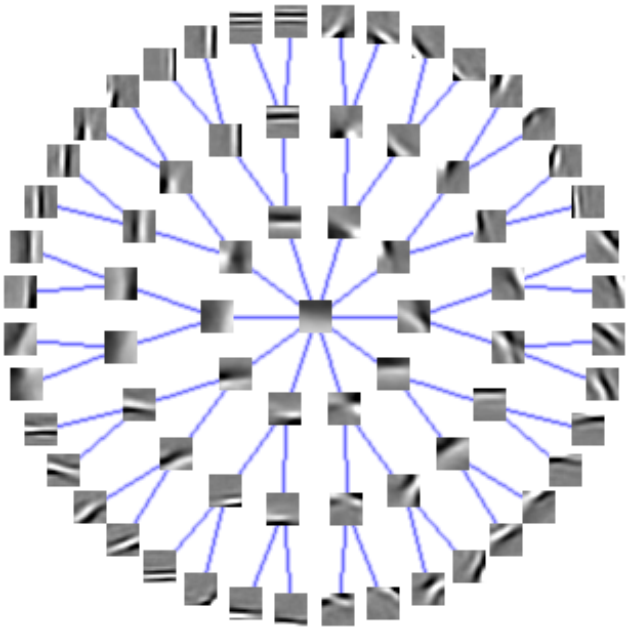

Figure 3. Learned dictionary with tree structure of depth 4. The root of the tree is in the middle of the figure. The branching factors are $p_1 = 10$, $p_2 = 2$, $p_3 = 2$. The dictionary is learned on $50,000$ patches of size 16×16 pixels.

4.2. Text Documents

This second experimental section shows that our approach can also be applied to model text corpora. The goal of probabilistic topic models is to find a low-dimensional representation of a collection of documents, where the representation should provide a semantic description of the collection. Within a parametric Bayesian framework, latent Dirichlet allocation (LDA) (Blei et al., 2003) models documents as a mixture of a predefined number of latent topics that are distributions over a fixed vocabulary. When one marginalizes over the topic random variable, one gets multinomial PCA (Buntine, 2002). The number of topics is usually small compared to the size of the vocabulary (e.g., 100 against 10,000), so that the topic proportions of each document give a compact representation of the corpus. For instance, these new features can be used to feed a classifier in a subsequent classification task. We will similarly use our dictionary learning approach to find low-dimensional representations of text corpora.

Suppose that the signals $\mathbf{X} = [\mathbf{x}^1, \ldots, \mathbf{x}^n] \in \mathbb{R}^{m \times n}$ are n documents over a vocabulary of m words, the k-th component of $\mathbf{x}^i$ standing for the frequency of the k-th word in the document i. If we further assume that the entries of $\mathbf{D}$ and $\mathbf{A}$ are nonnegative, and that the dictionary elements $\mathbf{d}^j$ have unit ℓ_1 norm, the decomposition $\mathbf{DA}$ can be seen as a mixture of p topics. The regularization Ω organizes these topics in a tree, so that, if a document involves a certain topic, then all its ancestors in the tree are also present in

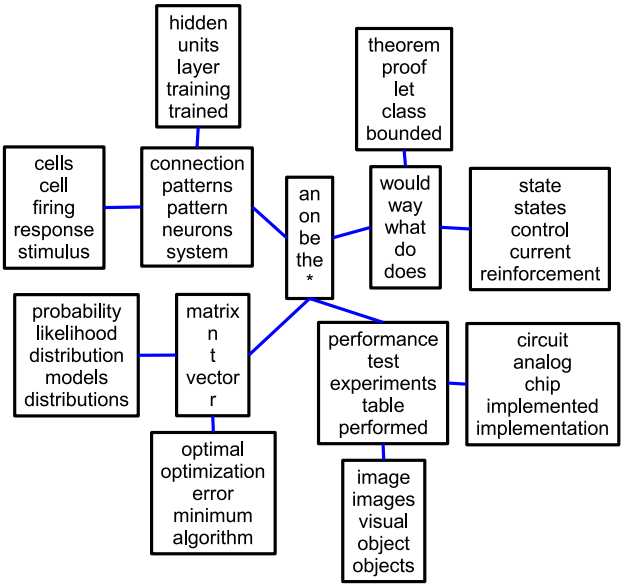

Figure 4. Example of a topic hierarchy estimated from 1714 NIPS proceedings papers (from 1988 through 1999). Each node corresponds to a topic whose 5 most important words are displayed. Single characters such as n, t, r are part of the vocabulary and often appear in NIPS papers, and their place in the hierarchy is semantically relevant to children topics.

the topic decomposition. Since the hierarchy is shared by all documents, the topics located at the top of the tree will be part of every decomposition, and should therefore correspond to topics common to all documents. Conversely, the deeper the topics in the tree, the more specific they should be. It is worth mentioning the extension of LDA that considers hierarchies of topics from a non-parametric Bayesian viewpoint (Blei et al., 2010). We plan to compare to this model in future work.

Visualization of NIPS proceedings. We first qualitatively illustrate our dictionary learning approach on the NIPS proceedings papers from 1988 through 1999[8]. After removing words appearing fewer than 10 times, the dataset is composed of 1714 articles, with a vocabulary of 8274 words. As explained above, we consider $\mathcal{D}_1^+$ and take $\mathcal{A}$ to be $\mathbb{R}_+^{p\times n}$. Figure 4 displays an example of a learned dictionary with 13 topics, obtained by using the ℓ_∞ norm in Ω and selecting manually $\lambda = 2^{-15}$. Similarly to Blei et al. (2010), we interestingly capture the stopwords at the root of the tree, and the different subdomains of the conference such as neuroscience, optimization or learning theory.

Posting classification. We now consider a binary classification task of postings from the 20 Newsgroups data set[9]. We classify the postings from the two newsgroups *alt.atheism* and *talk.religion.misc*, following the setting of

[8] http://psiexp.ss.uci.edu/research/programs_data/toolbox.htm

[9] See http://people.csail.mit.edu/jrennie/20Newsgroups/

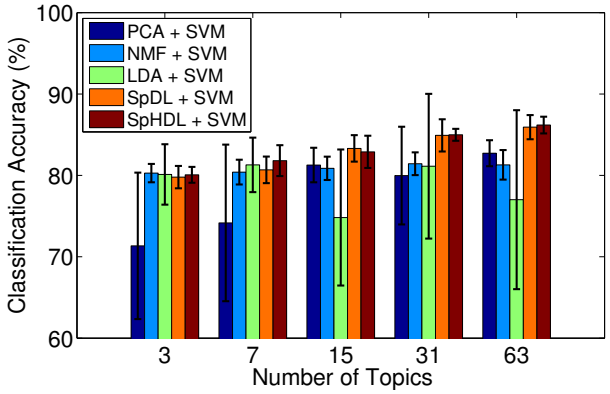

Figure 5. Binary classification of two newsgroups: classification accuracy for different dimensionality reduction techniques coupled with a linear SVM classifier. The bars and the errors are respectively the mean and the standard deviation, based on 10 random split of the data set. Best seen in color.

Zhu et al. (2009). After removing words appearing fewer than 10 times and standard stopwords, these postings form a data set of 1425 documents over a vocabulary of 13312 words. We compare different dimensionality reduction techniques that we use to feed a linear SVM classifier, i.e., we consider (i) LDA (with the code from Blei et al., 2003), (ii) principal component analysis (PCA), (iii) non-negative matrix factorization (NMF), (iv) standard sparse dictionary learning (denoted by SpDL) and (v) our sparse hierarchical approach (denoted by SpHDL). Both SpDL and SpHDL are optimized over $\mathcal{D}_1^+$ and $\mathcal{A} = \mathbb{R}_+^{p\times n}$, with the weights w_g equal to 1. We proceed as follows: given a random split into a training/test set of 1000/425 postings, and given a number of topics p (also the number of components for PCA, NMF, SpDL and SpHDL), we train a SVM classifier based on the low-dimensional representation of the postings. This is performed on the training set of 1000 postings, where the parameters, $\lambda \in \{2^{-26}, \ldots, 2^{-5}\}$ and/or $C_{\text{svm}} \in \{4^{-3}, \ldots, 4^{1}\}$ are selected by 5-fold cross-validation. We report in Figure 5 the average classification scores on the test set of 425 postings, based on 10 random splits, for different number of topics. Unlike the experiment on the image patches, we consider only one tree structure, namely complete binary trees with depths in $\{1, \ldots, 5\}$. The results from Figure 5 show that SpDL and SpHDL perform better than the other dimensionality reduction techniques on this task. As a baseline, the SVM classifier applied directly to the raw data (the 13312 words) obtains a score of 90.9 ± 1.1, which is better than all the tested methods, but without dimensionality reduction (as already reported by Blei et al., 2003). Moreover, the error bars indicate that, though nonconvex, SpDL and SpHDL do not seem to suffer much from instability issues. Even if SpDL and SpHDL perform similarly, SpHDL has the advantage to give a more interpretable topic mixture in terms of hierarchy, which standard unstructured sparse coding cannot.

5. Discussion

We have shown in this paper that tree-structured sparse decomposition problems can be solved at the same computational cost as addressing classical decomposition based on the ℓ_1 norm. We have used this approach to learn dictionaries embedded in trees, with application to representation of natural image patches and text documents.

We believe that the connection established between sparse methods and probabilistic topic models should prove fruitful as the two lines of work have focused on different aspects of the same unsupervised learning problem: our approach is based on convex optimization tools, and provides experimentally more stable data representations. Moreover, it can be easily extended with the same tools to other types of structures corresponding to other norms (Jenatton et al., 2009; Jacob et al., 2009). However, it is not able to learn elegantly and automatically model parameters such as dictionary size of tree topology, which Bayesian methods can. Finally, another interesting common line of research to pursue is the supervised design of dictionaries, which has been proved useful in the two frameworks (Mairal et al., 2009; Blei & McAuliffe., 2008).

Acknowledgments

This paper was partially supported by grants from the Agence Nationale de la Recherche (MGA Project) and from the European Research Council (SIERRA Project).

References

Bach, F. High-dimensional non-linear variable selection through hierarchical kernel learning. Technical report, arXiv:0909.0844, 2009.

Beck, A. and Teboulle, M. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imag. Sci.*, 2(1):183–202, 2009.

Bengio, S., Pereira, F., Singer, Y., and Strelow, D. Group sparse coding. In *Adv. NIPS*, 2009.

Bengio, Y. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1), 2009.

Bertsekas, D. P. *Nonlinear programming*. Athena Scientific Belmont, 1999.

Blei, D., Ng, A., and Jordan, M. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.

Blei, D. and McAuliffe, J. Supervised topic models. In *Adv. NIPS*, 2008.

Blei, D., Griffiths, T., and Jordan, M. The nested chinese restaurant process and Bayesian nonparametric inference of topic hierarchies. *Journal of the ACM*, 2010.

Boyd, S. P. and Vandenberghe, L. *Convex optimization*. Cambridge University Press, 2004.

Buntine, W.L. Variational Extensions to EM and Multinomial PCA. In *Proc. ECML*, 2002.

Elad, M. and Aharon, M. Image denoising via sparse and redundant representations over learned dictionaries. *IEEE Trans. Image Process.*, 54(12):3736–3745, 2006.

Jacob, L., Obozinski, G., and Vert, J.-P. Group Lasso with overlap and graph Lasso. In *Proc. ICML*, 2009.

Jenatton, R., Audibert, J.-Y., and Bach, F. Structured variable selection with sparsity-inducing norms. Technical report, arXiv:0904.3523, 2009.

Jenatton, R., Obozinski, G., and Bach, F. Structured sparse principal component analysis. In *Proc. AISTATS*, 2010.

Ji, S., and Ye, J. An accelerated gradient method for trace norm minimization. In *Proc. ICML*, 2009.

Kavukcuoglu, K., Ranzato, M., Fergus, R., and LeCun, Y. Learning invariant features through topographic filter maps. In *Proc. CVPR*, 2009.

Kim, S. and Xing, E. P. Tree-guided group lasso for multi-task regression with structured sparsity. Technical report, arXiv:0909.1373, 2009.

Lee, D. D. and Seung, H. S. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755): 788–791, 1999.

Lee, H., Battle, A., Raina, R., and Ng, A. Y. Efficient sparse coding algorithms. In *Adv. NIPS*, 2007.

Mairal, J., Bach, F., Ponce, J., Sapiro, G., and Zisserman, A. Supervised Dictionary Learning. In *Adv. NIPS*, 2009.

Mairal, J., Bach, F., Ponce, J., and Sapiro, G. Online learning for matrix factorization and sparse coding. *J. Mach. Learn. Res.*, 11, 19–60, 2010.

Nesterov, Y. Gradient methods for minimizing composite objective function. Technical report, CORE, 2007.

Olshausen, B. A. and Field, D. J. Sparse coding with an overcomplete basis set: A strategy employed by V1? *Vision Research*, 37:3311–3325, 1997.

Zhao, P., Rocha, G., and Yu, B. The composite absolute penalties family for grouped and hierarchical variable selection. *Ann. Stat.*, 37(6A):3468–3497, 2009.

Zhu, J., Ahmed, A., and Xing, E. P. MedLDA: maximum margin supervised topic models for regression and classification. In *Proc. ICML*, 2009.