LEARNING FROM SPARSE CODES

Sofia Karygianni and Pascal Frossard

Signal Processing Laboratory (LTS4)
Ecole Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland

ABSTRACT

In this paper we address the problem of learning image structures directly from sparse codes. We first model images as linear combinations of molecules, which are themselves groups of atoms from a redundant dictionary. We then formulate a new structure learning problem that learns molecules directly from image sparse codes, namely from the image representation in the atom domain. We build on a structural difference function that permits to compare molecules and we derive an algorithm that analyses sparse codes and estimates the most relevant signal structure without reconstructing the images. Experiments on both synthetic and real image datasets confirm the benefits of our new method compared to traditional learning methods.

Index Terms— sparse domain, geometric representation, representation learning

1. INTRODUCTION

Structured sparsity [1, 2, 3] has become a popular trend in signal processing since the consideration of dependencies between signal elements and not only of their number leads to very effective signal models. However, extracting the structure from the signals is quite a challenging task. Several methods have been proposed to deal with this problem like the double sparsity model in [4], the Markov Random Fiels (MRFs) models in [5, 6, 7] and various deep learning architectures [8, 9].

In this paper, we propose a new perspective on representation learning and we address the problem of estimating the image structures directly in the sparse code domain. Our goal is to provide a way to understand and analyze the sparse codes with minimal information requirements on the sparse coding algorithm and the dictionary of atoms, which are not always readily available. We consider that the signal, given by its sparse code, is actually a linear combination of a few molecule realizations. The molecule realizations are linear combinations of atoms and they resemble the molecule prototypes which define the main structures in the image. Equipped with a structural difference measure [10] that permits to compare molecules built on possibly redundant dictionaries of atoms, we propose a new learning algorithm for representing an image sparse code with molecule realizations and estimate the corresponding molecule prototypes. Our algorithm is different than traditional dictionary learning [11, 4, 12] as it does not work in the image domain, hence the possible similarities between the different code entries have to be taken into account carefully.

Our algorithm is able to recover the signal structure independently of the exact atom form and of the actual sparse coding algorithm that produced the codes. In this way, we follow the idea supported by various biological evidence that sparsity can be employed to produce representations that fully describe the corresponding signals without the need to reconstruct them. Moreover, in our scheme we directly deal with sparse data which is much more computationally efficient than working with the original signals. In the rest of the paper, we provide more details about the problem formulation and the proposed algorithm in Sections 2 and 3 respectively. Finally, in Section 4 we provide results on both synthetic and real images that verify the effectiveness of our scheme.

2. PROBLEM FORMULATION

2.1. Signal model

We formulate now the problem of learning the underlying structure of images given by their sparse codes. As with most structured sparsity priors, we assume that the occurrence of atoms in the sparse codes is not completely independent but that atoms rather tend to form typical visual patterns. In other words, there are some linear combinations of atoms that tend to appear more frequently than others. In our work, we call these underlying patterns 'molecules' and we assume that each code is a linear combinations of a few molecules. Therefore, our task is to identify these groups of atoms from sparse codes. In order to cope with the overcomplete nature of the underlying dictionary and the variability of natural signals, we distinguish between molecule prototypes and molecule realizations, which form respectively the image structures and their actual appearance in different signals. We employ the structure difference function used for structured sparse coding with pre-defined dictionaries in [10], in order to compare different sets of atoms without explicit knowledge of the underlying atoms. To be more specific, we consider a set of sparse codes $X = \{x \in \mathbb{R}^N,\ ||x||_0 \leq T_S\}$, where T_S is the maximum sparsity level. We want to learn a set of M molecule prototypes $C_\pi = [c_{\pi,1}\, c_{\pi,2} \ldots c_{\pi,M}] \in \mathbb{R}^{N\times M}$, $||c_{\pi,i}||_0 < T_A, \forall i$, where each prototype is a sparse linear combination of atoms on a base dictionary $D \in \mathbb{R}^{N\times K}$, such that each sparse code $x \in X$ can be represented well as a linear combination of a few slightly deformed versions of the molecules prototypes.

The deformed versions of the prototypes are called molecule realizations and they are defined based on the notion of pools of atoms. The pools of atoms represent atoms in D that are similar i.e., the pool of $d_j \in D$ is defined as $P(d_j) = \{d_m, 1 \leq m \leq K,\ |\ \langle d_j, d_m\rangle > 1 - \epsilon\}$ where ϵ is a suitable chosen threshold depending on the application at hand and the coherence of the dictionary D. Then, the actual energy corresponding to the atom d_j in a sparse code x can be calculated by measuring the energy captured by the coefficients of all the atoms in the pool $P(d_j)$, i.e., $e_j(x) = \sum_{d_m \in P(d_j)} x(m)\langle d_j, d_m\rangle = S_j\, x$. The vector S_j essentially expresses the pairwise relationships between the atom d_j and the rest of the atoms in the dictionary D. It is non-zero only when

$d_m \in P(d_j)$. A molecule realization $c_{x,i}$ is then a deformation of a molecule prototype $c_{\pi,i}$ whose original atoms could be each substituted by atoms from their respective pool. As a result, a molecule realization has a similar energy as the prototype when measured on atom pools but not necessary exactly the same coefficients on the atoms. Therefore, the realizations of a molecule prototype are allowed to have non-zero values only in the union of the pools of the active atoms in the prototype, namely $\Gamma_{P_{\pi,i}} = \bigcup_{d_k \in \Gamma_{\pi,i}} P(d_k)$. By denoting $\Gamma_{x,i}$ and $\Gamma_{\pi,i}$ the set of atoms for which the $c_{x,i}$ and the $c_{\pi,i}$ are non-negative respectively, the constraint on the support of $c_{x,i}$ can be written as $\Gamma_{x,i} \subseteq \Gamma_{P_{\pi,i}}$. The structural difference Δ then computes the energy in the pools of $c_{x,i}$ and compares it to the ones in $c_{\pi,i}$, i.e.,

$$\begin{aligned}\Delta(c_{\pi,i}, c_{x,i}) &= \sum_{d_j \in \Gamma_{\pi,i}} (c_{\pi,i}(j) - e_j(c_{x,i}))^2 \\ &= ||W_i \times (c_{\pi,i} - S\, c_{x,i})||_2^2 \end{aligned} \qquad (1)$$

where $S = [S_1\, S_2\, \cdots\, S_K]$. The indicator vector W_i denotes the inclusion of dictionary atoms in the support $\Gamma_{\pi,i}$ of the molecule prototype $c_{\pi,i}$, and it is non-zero only for the atoms $d_j \in \Gamma_{\pi,i}$. The operator $\times$ stands for element-wise multiplication. Equipped with the above definitions, each sparse code $x \in X$ can finally be written as a sparse non-negative combination of molecules realizations plus some bounded noise, i.e.,

$$\begin{aligned} x = \quad & C_x a_x + \eta, \text{ with } C_x = [c_{x,1}\, c_{x,2}\, \cdots\, c_{x,Q}] \\ & \text{and} \quad \Delta(c_{\pi,i}, c_{x,i}) < T, \forall i \end{aligned} \qquad (2)$$

with the atom and molecule coefficients $a_x(k) \geq 0, \forall k$ and $c_{x,i}(k) \geq 0, \forall (k,i)$ and $||a_x||_0 \leq T_M$, $\forall x$ for some sparsity threshold T_M.

2.2. Structure learning problem

With the code representation in Eq. (2), the structure learning problem is cast as an optimization that seeks the best set of molecules prototypes $\hat{C}_\pi$ to effectively represent a set of signals given by their sparse codes X. It can be written as:

$$\hat{C}_\pi = \arg\min_{C_\pi \geq 0} \sum_{x \in X} \arg \min_{\substack{a_x \geq 0,\, c_{x,i} \geq 0 \\ \Delta(c_{\pi,i}, c_{x,i}) \leq T \\ \Gamma_{x,i} \subseteq \Gamma_x \cap \Gamma_{P_{\pi,i}},\, \forall i}} ||x - C_x a_x||_2^2$$

$$\text{such that} \quad ||a_x||_0 \leq T_M \text{ and } ||c_{\pi,i}||_0 \leq T_A, \forall i, x \qquad (3)$$

The threshold parameters T_M and T_A control the sparsity of the representation while the parameter T controls the flexibility to deformations. All parameters are dependent on the application at hand. The support of each molecule realization $\Gamma_{x,i}$, is constrained by definition to be a subset of $\Gamma_{P_{\pi,i}}$. In order to comply with the sparse nature of the code x, we constrain it to be also a subset of the support of x, i.e., $\Gamma_{x,i} \subseteq \Gamma_x \cap \Gamma_{P_{\pi,i}}$. Note finally that the only dependence on the dictionary D lies in the structural difference $\Delta(c_{\pi,i}, c_{x,i})$ from Eq. (1) that only depends on the matrix S, representing the information about the atoms' pools, and not on the actual underlying dictionary D.

3. THE MLSC ALGORITHM

The problem in Eq. (3) is highly complicated and non-convex as it requires to solve for the code representations for all $x \in X$ as well as the molecule prototypes C_π. To solve it, we adopt the technique of alternating optimization to design our algorithm for molecule learning from sparse codes (MLSC). In particular, we divide the learning problem into two sub-problems: the representation of the sparse codes as linear combinations of molecule realizations given the molecule prototypes and the update of the molecule prototypes based on the codes' representations. We iterate over these two steps until we reach convergence. To check for convergence we use a symmetric extension of the structural difference measure in Eq. (1) suitable for molecule prototypes i.e., we set the difference between two molecule prototypes $c_{\pi,k}$ and $c_{\pi,m}$ to be $\Delta_\pi(c_{\pi,k}, c_{\pi,m}) = \frac{1}{2}(\Delta(c_{\pi,k}, c_{\pi,m}) + \Delta(c_{\pi,k}, c_{\pi,m}))$. Then, given the old and the new molecule sets, we use the Hungarian algorithm [13] to find the matching with the minimum cost i.e., minimum sum of Δ_π's over the pairs of matched prototypes. If that cost is small, the sets are similar and the algorithm has converged.

3.1. Sparse code representation

We present now our scheme for representing the sparse codes as linear combinations of molecule realizations given the molecule prototypes. In this case, the problem in Eq. (3) becomes equivalent to representing a sparse code $x \in \mathbb{R}^N$ as a set of molecule realizations $x \approx C_x a_x$ given the prototypes C_π, where C_x and a_x follow the required constraints. This is essentially a constrained sparse coding problem. However, the sparse nature of the codes in combination with the absence of the dictionary D from the data fidelity term complicates the procedure since now a code x and a molecule are similar only if they have an overlap in their non-zero entries.

To resolve this issue, we propose a greedy matching pursuit algorithm [14, 15] that adjusts the molecule prototypes to the code, i.e., it solves for the molecule realizations simultaneously with the coefficients while taking into account the atom similarities. The algorithm starts with an empty code representation and a residual code equal to the original sparse code. Then, at each iteration the molecule that best fits the residual code is picked for the code representation. The residual is then updated and the iterations continue until either the maximum number of allowed molecules in the representation T_M is reached, or the residual cannot be reduced anymore.

However, when picking the next molecule a simple inner product solution between the residual and the molecule prototype is challenged by potential non-overlaps in their support. Therefore, we have designed a new scheme for discovering the realization of a molecule prototype that best approximates a residual sparse code by taking into account the special characteristics of the codes. In particular, to project a residual sparse code r to the direction of a prototype $c_{\pi,i}$, we need to find the molecule realization of $c_{r,i}$ that best approximates r, while fulfilling the structural constraints $\Gamma_{r,i} \subseteq \Gamma_r \cap \Gamma_{P_{\pi,i}}$ and $\Delta(c_{\pi,i}, c_{r,i}) = ||W_i \times (c_{\pi,i} - S\, c_{r,i})|| \leq T$. The constraint on the support $\Gamma_{r,i}$ indicates exactly which entries should be non-zero in $c_{r,i}$, namely the common atoms between the residual code and the union of active pools in the prototype $c_{\pi,i}$. Then, we only need to decide the actual coefficients on this support. In particular, it is sufficient to decide the energies in the active pools. Therefore we can transform all the vectors to the pool level, i.e., $r_p = W_i \times (S\, r)$, $c_p = W_i \times c_{\pi,i}$ and $v_p = W_i \times (S\, c_{r,i})$ for the residual, the prototype and the realization respectively. Then, we get the following optimization problem:

$$\{\hat{b}, \hat{v}_p\} = \arg \min_{\substack{b \geq 0\; v_p \geq 0, \\ ||c_p - v_p|| \leq T}} ||r_p - v_p\, b||_2^2 \qquad (4)$$

This is essentially the problem of projecting the point r_p to the closed convex cone C_Ω, where Ω is the intersection of the hypersphere H defined by $||c_p - v_p|| \leq T$ and the non-negative orthant J. According to the projection theorem [16], it is a convex problem with a unique solution which could be solved by iterating over projections on the individual sets [17]. However, in our case due to the small value of T and the positivity of c_p, it usually happens that the hypersphere H lies entirely into J, i.e. $\Omega = H \cup J = H$. In this case, our problem has a closed form solution whose detailed computation is presented in [18].

3.2. Structure update

After finding the codes' representations, we need to update the set of molecule prototypes C_π. To simplify this optimization problem, we update each molecule prototype alone. However, since the molecule realizations are strongly related to the corresponding prototypes through the Δ structural constraints, the prototypes and their realizations need to be updated at the same time. In particular, for each molecule $i \in [1, M]$, we find the codes that use it, i.e., $X_i = \{x \in X,\ a_{x,i} > 0\}$. Then, for each code $x \in X_i$ we define the residual code with respect to the ith molecule, i.e., $e_x = x - \sum_{j \neq i} C_{x,j} a_{x,j}$. The goal of the optimization is then to uncover a prototype $c_{\pi,i}$ and the corresponding code representations $c_{x,i}, a_{x,i}$ for all the codes in X_i such that the e_x's are well approximated. A major challenge in the corresponding optimization problem is the existence of binary variables namely the indicator function of the support of the prototype W_i and the corresponding union of active pools $\Gamma_{P_{\pi,i}}$. As a result, we have a mixed-integer optimization problem whose exact solution can be time consuming as it may require a full search of the variable space. To overcome this difficulty, we build an approximate solution in two successive steps. In Step A, we decide on the support $\Gamma_{\pi,i}$ of the prototype $c_{\pi,i}$, and in Step B we solve for the coefficients of the prototype on the chosen support $\Gamma_{\pi,i}$.

3.2.1. *Step A: Solve for molecule support*

In this step, we solve for the support of the prototype $c_{\pi,i}$, denoted as $\Gamma_{\pi,i}$. Our solution is based on the fact that the active pools in the molecule prototype, namely $\Gamma_{P_{\pi,i}}$, should cover as many of the non-zero entries in the e_x's as possible. Otherwise, the molecule realizations cannot be non-zero in these positions and the approximation error increases. However, given the maximum number of pools in a prototype T_A, deciding which pools to pick to maximize the covered energy in the residuals e_x is an NP-hard problem [19]. To solve the problem efficiently, we propose instead a greedy solution that approximates the optimal support through iterations: the algorithm starts with an empty support set and at each iteration adds the most 'energetic' pool to it. The total energy in each pool is computed as $E_p = \sum_{x \in X_i} S\ r_x$ where S is the matrix representing the atoms' pools defined in Section 2.1 and r_x are the residual codes, initialized to e_x. At each iteration the pool with the highest value in E_p is added to the support. Then, the residuals r_x are updated by excluding the coefficients in the chosen pool, and the iterations continue until the maximum number of atoms per prototype is reached.

3.2.2. *Step B: Solve for molecule coefficients*

After computing $\Gamma_{\pi,i}$, in the second step we solve for the exact coefficients of the prototype $c_{\pi,i}$. As before, we only need to compute the energies at the pool level. Therefore, we end up with a formulation similar to the one in Eq. (4). The difference is that now multiple r_p's are available and the unknowns are both the c_p and the v_p's. Seen from a geometric perspective, the unknown now is the direction of the center of the cone C_Ω which minimizes the sum of distances between the set of r_p's and their projections on the cone. For the sake of efficiency, we have approximated the solution by picking the direction that maximizes the projections of the r_p's, i.e., their first principal component. This solution is more accurate when the value of the radius T is small, which is usually the case, as then the cone C_Ω shrinks close to its central line.

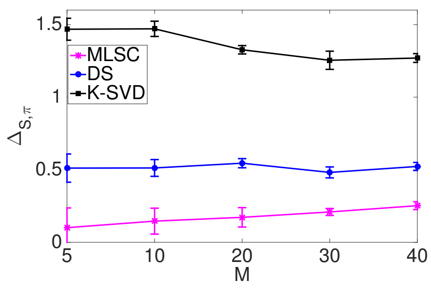
Once the prototype $c_{\pi,i}$ has been computed through steps A and B, we complete the solution by finding the molecule realizations $c_{x,i}$ that best fit the residual codes $e_x, \forall x \in X_i$ with our corresponding algorithm described in Section 3.1.

4. EXPERIMENTAL RESULTS

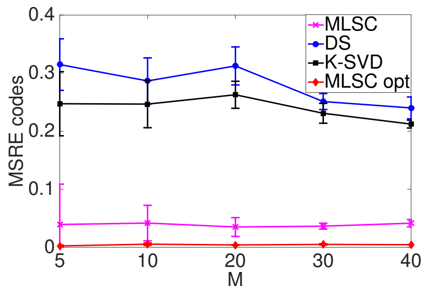
We have experimented with both synthetic and real images to check the performance of our learning algorithm, namely MLSC, when learning the structure of a set of sparse codes. We have used an underlying dictionary of gaussian anisotropic atoms i.e., $D = \{\phi_u : u = (\tau_x, \tau_y, r, \sigma) \in U\}$ where $\phi(x, y) = A \exp(-(x/h)^2 - y^2)$ is the gaussian mother function and $\phi_u(x, y) = \phi(x', y')$ with $x' = \cos r\ (x - \tau_x) + \sin r(y - \tau_y), y' = (1/s)(-\sin r\ (x - \tau_x) + \cos r(y - \tau_y))$ is the transformation between the mother function and an atom ϕ_u. In this definition, h stands for the anisotropy, r for rotation, τ_x and τ_y for translations in x and y directions, and σ for scale. We compare our results with those of the double sparsity algorithm (DS) [4] and K-SVD [11]. The input to the algorithms in all cases is the sparse codes of the data. The performance is measured with the structural difference for molecule sets introduced in Section 3 when the ground truth is known and with the mean square reconstruction error (MSRE) of the testing set in both the sparse code and the image domain.

4.1. Synthetic data

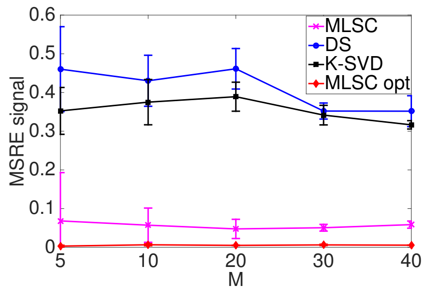
For our synthetic data, the atoms in D are of size 14×14 and anisotropy $h = 2$. We have sampled the image plane for the scales 0.5 and 1 with a step size of 2 for translation and $\frac{\pi}{6}$ for rotation. Each molecule has been randomly constructed to contain 2, 3 or 4 atoms of equal energy. To produce a molecule realization, each non-zero atom in the molecule prototype is substituted by a few atoms in its pool. The results of the experiments, averaged over 5 different structure instances for each M, are shown in Figure 1. From the first plot in the Figure 1, we can verify that our scheme manages to uncover the structure that is the most relevant one in terms of structural difference. In the same plot, we observe that K-SVD unsurprisingly performs very poorly in terms of this measure, since it does not take into account the sparse nature of the molecules. Double sparsity on the other hand, performs better than K-SVD but still worse than our scheme. In the next two plots, we plot the MSRE for the testing set in both the sparse code and image domain. For reference, we also plot the MSRE achieved when the optimal structure is used in the coding, denoted as 'MLSC, opt'. The qualitative behavior in both domains is the same for all schemes with our scheme being the closest to the performance of the optimal structure. The interesting twist however is that the MSRE achieved by the structure learned with K-SVD is a bit better than that of the double sparsity scheme. This means that the non-sparse nature of the molecules of K-SVD that is completely wrong in terms of structure, permits a better approximation perfor-



(a) Structural difference



(b) MSRE is sparse code



(c) MSRE in signal domain

Fig. 1. The evaluation of the structures learned by the different schemes over the number of molecules M in the dictionary for the case of synthetic data. In (a) we plot the structural difference between the learned models and the optimal structure and in (b) and (c) the MSRE in both the sparse code and image domain for the testing set.

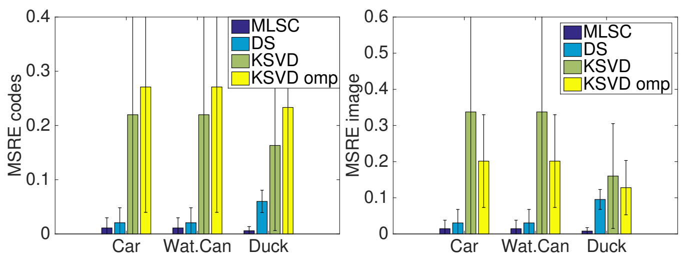


Fig. 2. Comparison of the MSRE on ALOI dataset in both the sparse code and image domain. The 30 molecule prototypes are extracted with 3 different schemes namely MLSC, DS and K-SVD and the coding is performed with our scheme for MLSC and DS and with both our scheme and the OMP for the K-SVD.

mance of the signals while the sparse and strict molecules of the DS scheme are more accurate but perform worse in the approximation. Therefore, the two performance measures are not equivalent and satisfying both tasks, namely correct structure and good approximation performance, seems challenging. Interestingly enough, our scheme performs well on both aspects.

4.2. Object images

We have also experimented with images of objects from the Amsterdam Library of Object images (ALOI) [20] which is a color image collection of small objects where each object is recorded under different viewing angles and illumination settings. In our experiments, we have used the grayscale version of the images downsampled to 35×35 pixels. For the sparse representation of the images we have used the dictionary D with the anisotropy h set to 1 and 4 and the scale s to $2^{[0:0.5:4]}$ as in [21]. The translation parameters τ_x, τ_y are sampled with a step size of 0.5 and the rotation parameter r with $\frac{\pi}{8}$.

We compare the approximation performance of the molecules learned with the different schemes when we set the values of the MLSC parameters as $T_A = 10, T_M = 5, T = 0.2, T_E = 0$ and $M = 30$. Since the number of objects per category is small in the ALOI dataset, approximately 100 instances per object, we use all the instances for training and we report the MSRE for these images in Figure 2. For the K-SVD algorithm, we plot both cases of coding with our algorithm and the classical OMP as the nature of the molecules extracted by KSVD is not always proper for our scheme

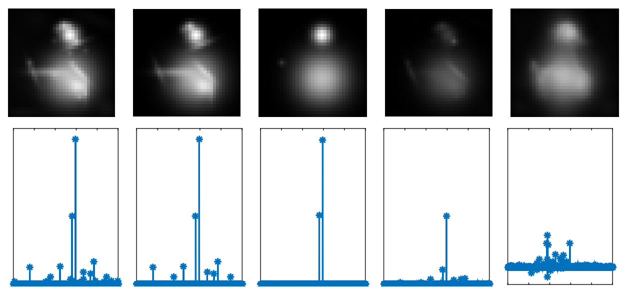


Fig. 3. Approximation example for an instance of the object duck. On the bottom line, from left to right we have: the initial code and the codes retrieved with molecules learned with the MLSC, DS and K-SVD (for KSVD molecules coding with both our coding scheme and OMP) respectively. For better visualization, on the top line we also provide the corresponding images.

(negative values, non-sparse). From the Figure 2, we can verify that the molecules learned with our scheme, approximate the signals better than the molecules learned with the competing schemes in both domains. This can also be verified from the Figure 3 where we show the approximation of a 'Duck' instance from all algorithms in both the image and the sparse domain.

5. CONCLUSIONS

We have presented a new algorithm for learning image structures directly from sparse codes. In order to deal efficiently with the resulting complex optimization problem, we have alternated between steps of finding the representation of the codes based on the current molecule structure and then updating the structure based on the codes' representation. Our scheme requires only minimal knowledge about the underlying dictionary, namely the 'correlation' matrix S of the atoms' pools. We have tested our scheme with both synthetic and object images and we have verified the superior performance of our scheme compared to other existing learning techniques that are however not designed explicitly for the sparse domain.

6. REFERENCES

[1] M. Yuan and Y. Lin, "Model selection and estimation in regression with grouped variables," *Journal of the Royal Statistical Society*, vol. 68, no. 1, pp. 49–67, 2006.

[2] P. Zhao, G. Rocha, and B. Yu, "The composite absolute penalties family for grouped and hierarchical variable selection," *Annals of Statistics*, pp. 3468–3497, 2009.

[3] R. Jenatton, J.-Y. Audibert, and F. Bach, "Structured variable selection with sparsity-inducing norms," *The Journal of Machine Learning Research*, vol. 12, pp. 2777–2824, 2011.

[4] R. Rubinstein, M. Zibulevsky, and M. Elad, "Double sparsity: Learning sparse dictionaries for sparse signal approximation," *IEEE Trans. on Signal Processing*, vol. 58, no. 3, pp. 1553–1564, 2010.

[5] T. Peleg, Y. C. Eldar, and M. Elad, "Exploiting statistical dependencies in sparse representations for signal recovery," *IEEE Trans. on Signal Processing*, vol. 60, no. 5, pp. 2286–2303, 2012.

[6] P. J. Garrigues and B. A. Olshausen, "Learning horizontal connections in a sparse coding model of natural images," in *Advances in Neural Information Processing Systems (NIPS)*, 2008, pp. 505–512.

[7] V. Cevher, M. F. Duarte, C. Hegde, and R. G. Baraniuk, "Sparse signal recovery using markov random fields," in *Advances in Neural Information Processing Systems (NIPS)*, 2008, pp. 257–264.

[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012, pp. 1097–1105.

[9] Q. V. Le, "Building high-level features using large scale unsupervised learning," in *IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 8595–8598.

[10] S. Karygianni and P. Frossard, "Sparse molecular image representation," *Journal of Visual Communication and Image Representation*, vol. 36, pp. 213 – 228, 2016.

[11] M. Aharon, M. Elad, and A. Bruckstein, "K-svd: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.

[12] R. Jenatton, J. Mairal, F. R. Bach, and G. R. Obozinski, "Proximal methods for sparse hierarchical dictionary learning," in *Int. Conf. on Machine Learning (ICML)*, 2010, pp. 487–494.

[13] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval research logistics quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[14] S. G. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. on Signal Processing*, vol. 41, no. 12, pp. 3397–3415, 1993.

[15] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Journal on Scientific Computing*, vol. 20, no. 1, pp. 33–61, 1998.

[16] S. Boyd and L. Vandenberghe, *Convex optimization*, Cambridge university press, 2004.

[17] S.-P. Han, "A successive projection method," *Mathematical Programming*, vol. 40, no. 1-3, pp. 1–14, 1988.

[18] S. Karygianni, *Signal structure: from manifolds to molecules and structured sparsity*, Ph.D. thesis, EPFL, 2015.

[19] C. Stein, T. Cormen, R. Rivest, and C. Leiserson, *Introduction to algorithms*, vol. 3, MIT Press Cambridge, MA, 2001.

[20] J.-M. Geusebroek, G. J. Burghouts, and A. WM. Smeulders, "The amsterdam library of object images," *International Journal of Computer Vision*, vol. 61, no. 1, pp. 103–112, 2005.

[21] A. Fawzi and P. Frossard, "Image registration with sparse approximations in parametric dictionaries," *SIAM Journal on Imaging Sciences*, vol. 6, no. 4, pp. 2370–2403, 2013.