Improving Dictionary Learning: Multiple Dictionary Updates and Coefficient Reuse

Leslie N. Smith and Michael Elad (*Fellow, IEEE*)

***Abstract*—In this paper we propose two improvements of the MOD and K-SVD dictionary learning algorithms, by modifying the two main parts of these algorithms – the dictionary update and the sparse coding stages. Our first contribution is a different dictionary-update stage that aims at finding both the dictionary and the representations *while keeping the supports intact*. The second contribution suggests to *leverage the known representations* from the previous sparse-coding in the quest for the updated representations. We demonstrate these two ideas in practice and show how they lead to faster training and better quality outcome.**

I. Introduction

Sparse and redundant representation modeling of signals is a very effective way to describe the inner-structure of signal sources [1]. This model assumes that the signal $\mathbf{x} \in \mathbb{R}^d$ can be described as $\mathbf{x} = \mathbf{D}\boldsymbol{\alpha}$, where $\mathbf{D} \in \mathbb{R}^{d \times n}$ is a dictionary (matrix), and $\boldsymbol{\alpha} \in \mathbb{R}^n$ is the signal's representation, which is assumed to be sparse. We typically consider the case $n > d$, suggesting that the representation is redundant. The number of non-zeroes in the representation, denoted as $k = \|\boldsymbol{\alpha}\|_0$, is expected to be very small, i.e., $k \ll d < n$, implying that the signal $\mathbf{x}$ is characterized as being a linear combination of few columns (also referred to as atoms) from the dictionary.

This model has drawn considerable research attention in the past decade, with contributions spanning theoretical, numerical and applicative ideas [1]. A fundamental tool for the practice of this model is dictionary learning – the construction of a dictionary that is suitable for a family of signals that are of interest. Dictionary learning algorithms use a set of signal examples, $\{\mathbf{x}_i\}_{i=1}^N$, to identify $\mathbf{D}$ that will best sparsify them, thereby leading to more effective modeling. This can be formulated as the problem

$$\underset{\mathbf{D},\mathbf{A}}{\text{Argmin}} \;\; \|\mathbf{X} - \mathbf{DA}\|_F^2 \;\; s.t. \;\; \forall \; 1 \le i \le N, \;\; \|\boldsymbol{\alpha}_i\|_0 \le k \;, \quad (1)$$

where $\mathbf{X} \in \mathbb{R}^{d \times N}$ contains all the training examples as columns, and similarly, $\mathbf{A} \in \mathbb{R}^{n \times N}$ contains all their sparse representations, $\boldsymbol{\alpha}_i$. Dictionary learning starts with the given signals and aims to find both the dictionary $\mathbf{D}$ and the representations $\mathbf{A}$. This is a very complex optimization task in general, and existing methods for its solution can only hope to bring an approximate solution.

Various numerical algorithms have been proposed for the task of dictionary learning, and the literature on this topic is fast growing – see [2], [3], [4], [5], [6], [7], [8] for representative work, from the early stages of this field to very recent contributions. Among these, two very familiar algorithms are the MOD [9], [10] and the K-SVD [11], [12]. Both these methods (and in fact, many others among the list mentioned above) divide the numerical treatment of the optimization problem in Equation (1) into two phases – a sparse-coding stage that optimizes $\mathbf{A}$ assuming the knowledge of $\mathbf{D}$, and a dictionary update stage that updates $\mathbf{D}$ using the known representations $\mathbf{A}$. These algorithms differ in the details of the pursuit (finding $\mathbf{A}$) and the update of the dictionary $\mathbf{D}$.

In this paper we embark from the MOD and K-SVD dictionary learning methods, and propose improvements for the two phases on these algorithms – the dictionary update and the sparse coding stages. First, we change the dictionary update stage so as to find the dictionary and the representations while keeping the supports intact. In this respect, the updates proposed by the MOD and the K-SVD are in-fact sub-optimal processes, and we replace them with a more effective updates. Second, we notice that the sparse-coding stage is performed at every iteration while disregarding previous iteration's representation. We propose an alternative pursuit that modifies the existing representations, thereby getting faster to the desired solution. We demonstrate the two modifications on image-patch training, showing that the new emerging algorithm leads to faster training, and therefore to a better quality outcome when limited number of iterations is practiced.

II. Modified Dictionary-Learning Algorithm

In this section we present the two modifications we propose to the MOD and the K-SVD algorithms. In both cases, we start by describing the existing methods, and then suggest the improvements, so as to make the discussion self-contained.

A. Improved Dictionary-update Stage

Let's recall the way the dictionary update is done in MOD and K-SVD. Our starting point is the availability of the representations $\mathbf{A}$, and our goal is now to update $\mathbf{D}$. Considering the learning problem formulation in Equation (1), MOD does the most natural thing [9], [10], by minimizing the ℓ_2 term $\|\mathbf{X} - \mathbf{DA}\|_F^2$ with respect to $\mathbf{D}$. This amounts to

$$\hat{\mathbf{D}} = \underset{\mathbf{D}}{\text{Argmin}} \;\; \|\mathbf{X} - \mathbf{DA}\|_F^2 = \mathbf{XA}^T(\mathbf{AA}^T)^{-1} = \mathbf{XA}^\dagger. \quad (2)$$

In cases where the matrix $\mathbf{AA}^T$ is very large, an iterative solver for the normal system $\mathbf{XA}^T = \mathbf{DAA}^T$ can be considered. In such a case, a small number of iterations will provide an approximate solution of this system. K-SVD takes a different approach altogether [11], [12], by updating one atom at a time in $\mathbf{D}$. However, far more important is the fact that K-SVD not only updates the atom, but also all the non-zero coefficients in

L. N. Smith is with the US Naval Research Laboratory, Optical Sciences Division, 4555 Overlook Ave., SW., Washington, DC 20375 USA (e-mail: Leslie.Smith@nrl.navy.mil). M. Elad is with the Computer Science Department, Technion – Israel Institute of Technology, Haifa 32000, Israel (e-mail: elad@cs.technion.ac.il). This work was supported in part by the European Commissions FP7-FET program, SMALL project (grant agreement no. 225913) and by the US Naval Research Laboratory.

$\mathbf{A}$ that multiply it. Thus, K-SVD provides an update of both $\mathbf{D}$ and $\mathbf{A}$ in the "dictionary-update" stage.

Bearing the above in mind, what is truly the objective of the dictionary-update stage? What is the information we would like to carry from the sparse-coding stage as the foundation for this update? Adopting the K-SVD flavor of answer, our objective seems to be to find an update of $\mathbf{D}$ and $\mathbf{A}$ such that the *s*upports (location of the non-zeros) in $\mathbf{A}$ remain intact. Put formally, this leads to the optimization task

$$\{\hat{\mathbf{D}}, \hat{\mathbf{A}}\} = \underset{\mathbf{D},\mathbf{A}}{\text{Argmin}} \ \|\mathbf{X} - \mathbf{D}\mathbf{A}\|_F^2 \quad s.t. \quad \mathbf{A} \odot \mathbf{M} = 0. \tag{3}$$

The notation $\mathbf{A} \odot \mathbf{M}$ is an entry-wise (Schur) multiplication between two equally-sized matrices. The mask matrix $\mathbf{M}$ is a patterned matrix of zeros and ones, given by $\mathbf{M} = \{|\mathbf{A}| = 0\}$, implying that $\mathbf{M}(i,j) = 1$ if $\mathbf{A}(i,j) = 0$, and zeros elsewhere. Thus, the requirement $\mathbf{A} \odot \mathbf{M} = 0$ forces all the zero entries in $\mathbf{A}$ to remain intact. While the problem posed in (3) is much easier than the overall dictionary-learning task, it is still non-convex and hard to solve. One could propose various ways to iteratively solve this constrained problem. One such option is a block-coordinate-descent approach where we fix $\mathbf{A}$ and minimize $\|\mathbf{X} - \mathbf{D}\mathbf{A}\|_F^2$ w.r.t. $\mathbf{D}$ (which is exactly MOD), followed by an update of $\mathbf{A}$ by fixing $\mathbf{D}$ and solving

$$\hat{\mathbf{A}} = \underset{\mathbf{A}}{\text{Argmin}} \ \|\mathbf{X} - \mathbf{D}\mathbf{A}\|_F^2 \quad s.t. \quad \mathbf{A} \odot \mathbf{M} = 0. \tag{4}$$

This second optimization w.r.t. $\mathbf{A}$ has a closed-form solution, easily obtained by handling each column in $\mathbf{A}$ separately, and targeting only the non-zeros in these columns, leaving the zeros intact. The new formulation becomes

$$\hat{\boldsymbol{\alpha}}_i = \underset{\boldsymbol{\alpha}_i}{\text{Argmin}} \ \|\mathbf{x}_i - \mathbf{D}_i \boldsymbol{\alpha}_i\|_2^2 = \mathbf{D}_i^\dagger \mathbf{x}_i, \tag{5}$$

where $\mathbf{D}_i$ is a sub-matrix of $\mathbf{D}$ containing only the atoms in the support of this representation, and $\hat{\boldsymbol{\alpha}}_i$ is the non-zero portion of the i-th column in $\mathbf{A}$.

Iterating the above algorithm – updating $\mathbf{D}$ and then $\mathbf{A}$ as described above – will lead to an approximation of the solution for the problem we have posed in Equation (3). Again, we have noticed that the MOD is thus merely half an iteration of this process, and as such constitutes a weaker approximation of the overall desired "dictionary-update" stage.

Another option for handling the "dictionary-update" problem posed in Equation (3) aligns better with the K-SVD approach. Breaking the term $\mathbf{DA}$ to the sum of rank-1 outer-products, our problem becomes

$$\{\hat{\mathbf{D}}, \hat{\mathbf{A}}\} = \underset{\mathbf{D},\mathbf{A}}{\text{Argmin}} \ \|\mathbf{X} - \sum_{j=1}^{n} \mathbf{d}_j \mathbf{a}_j^T\|_F^2 \tag{6}$$
$$s.t. \ \ \forall\, 1 \le j \le n, \ \mathbf{m}_j \odot \mathbf{a}_j = 0.$$

In this formulation $\mathbf{d}_j$ is the j^{th} dictionary atom (column), $\mathbf{a}_j^T$ is the j^{th} row in $\mathbf{A}$ (as opposed to the representations $\boldsymbol{\alpha}_i$ that are columns of this matrix), and $\mathbf{m}_j^T$ is the j^{th} row in $\mathbf{M}$ that forces the zeros in the proper locations in $\mathbf{a}_j$. We can again use a block-coordinate-descent approach, but this time optimize sequentially the pairs $(\mathbf{d}_j, \mathbf{a}_j)$ for $j = 1, 2, \ldots, n$. This optimization is done by an SVD operation performed on the matrices $\mathbf{E}_j = (\mathbf{X} - \sum_{i \neq j} \mathbf{d}_i \mathbf{a}_i^T) \odot (\mathbf{1}_d \cdot \mathbf{m}_j^T)$. The mask matrix, $\mathbf{1}_d \cdot \mathbf{m}_j^T$ is a rank-1 matrix of size $d \times N$ that replicates the row $\mathbf{m}_j^T$ d times. This mask matrix effectively removes from $\mathbf{X} - \sum_{i \neq j} \mathbf{d}_i \mathbf{a}_i^T$ all the columns corresponding to examples that do not use the j^{th} atom.

While the K-SVD dictionary update stage performs exactly one such round of updates for $j = 1, 2, \ldots, N$, the algorithm we describe here should repeat these rounds several times for better approximating the overall solution of (3). Again, and just like in the MOD case, it appears that the K-SVD performs a first step towards the desired solution, but does not proceed to exhaust the potential in the "dictionary-update" stage.

To summarize this part, an effective dictionary-update stage should refresh both the dictionary and the non-zero coefficients in the representations, such that the representation error is minimized and the supports are maintained. We have described two iterative methods for approximating this goal. In both cases, MOD or K-SVD are merely the first part of the computational process, and the algorithms proceed iteratively, performing what we refer to hereafter as several Dictionary Update Cycles (DUC), in order to more effectively minimize the original dictionary learning objective in Equation (1).

As for computational complexity, both methods described above are roughly equivalent. While they add to the complexity of the dictionary-update stage, this additional complexity is negligible in the overall dictionary-learning process. This is because in most cases the learning task is heavily dominated by the sparse coding stage, thus implying that with the additional computational effort that we have brought here, the overall run-time remains almost unchanged.

B. *Coefficient Reuse OMP (CoefROMP)*

Given an updated dictionary, the MOD and K-SVD algorithms proceed by searching for the sparsest representations for the training data. The goal is the solution of the optimization problem posed in Equation (1) with respect to $\mathbf{A}$, while keeping $\mathbf{D}$ fixed. If we choose the previously computed representations[1], the objective function remains at the same height. This suggests that we may use the given representations as a warm-start for the pursuit stage. Note, however, that this does not imply a guaranteed improvement.

The idea to initialize the coefficients this way in the pursuit algorithm can be easily incorporated into various relaxation or greedy sparse coding methods. In this paper we focus on a specific variant of a greedy pursuit algorithm that builds on the work of CoSaMP [13] and Subspace Pursuit (SP) [14]. However, unlike these methods, our algorithm is initialized with the largest $k/3$ coefficients[2] from the previous pursuit stage, then proceeds with a coefficient augmentation and pruning process like CoSAMP and SP. The proposed algorithm is termed the *Coefficient Reuse OMP (CoefROMP)*, and it is described in Algorithm 1.

We should note that we tested the warm-start idea with several relaxation and greedy pursuit methods. A warm-start for the relaxation methods gave some speed improvement but

[1]Either from the previous iteration, or from the modified dictionary-update stage as described in Section II.A

[2]This choice was found empirically to perform well.

not any improvement in RMSE. Initializing OMP with the largest $k/3$ coefficients from the previous iteration and running standard OMP to find the remaining $2k/3$ coefficients led to modest improvements in RMSE and run-time. It appears that without the augmentation and pruning steps of CoefROMP, the warm-start idea does not lead to the significant improvements that will be described in the Results section.

Two versions of CoefROMP were developed – the first that targets a desired cardinality k, similar to CoSaMP [13] and SP [14], and a second that performs the sparse coding with an error stopping criteria by accepting only coefficients above a threshold. Also, the CoefROMP algorithm can be made far more efficient when adopting a Batch-OMP architecture [15]. Under the assumption that our pursuit algorithm is to operate on a large group of different signals using a fixed dictionary, various pre-computations can be done so as to reduce substantially the amount of operations in the coefficient computations. This is exactly the scenario we encounter in typical dictionary learning. Such a Batch-CoefROMP algorithm has been developed and shown to lead to a factor of $2-3$ savings in run-time. Due to space limitations, we present in Algorithm 1 only the core CoefROMP method.

Algorithm 1 COEFROMP

```
1: Input: $\mathbf{D}$, $\mathbf{x}$, $\boldsymbol{\alpha}_0$ (warm-start), and $k$ (target cardinality)
2: Output: Updated sparse representation $\boldsymbol{\alpha}$
3: Initialization (n=0): (i) $T_0 :=$ sort $(|\boldsymbol{\alpha}_0|, k/3)$ (take the leading $k/3$ elements from $\boldsymbol{\alpha}_0$); (ii) $\mathbf{r}_0 := \mathbf{x} - \mathbf{D}_{T_0}\boldsymbol{\alpha}_{T_0}$; and (iii) $\epsilon_0 := \|\mathbf{r}_0\|^2$
4: for n=1:1:max-iter do
5:     $S_n :=$ sort $\left(|\mathbf{D}^T\mathbf{r}_{n-1}|, k/3\right)$, take the leading $k/3$ elements from the projected residual
6:     $\tilde{T}_n := (T_{n-1}, S_n)$, merge the supports
7:     $\tilde{\boldsymbol{\alpha}}_n := (\mathbf{D}_{\tilde{T}_n})^+\mathbf{r}_n$, compute the representation by LS
8:     $T_n =$ sort $(|\tilde{\boldsymbol{\alpha}}_n|, k)$, take the leading $k$ (or less) elements from $\tilde{\boldsymbol{\alpha}}_n$
9:     $\boldsymbol{\alpha}_n := (\mathbf{D}_{T_n})^+\mathbf{x}$, update the representation by LS
10:    $\mathbf{r}_n := \mathbf{x} - \mathbf{D}_{T_n}\boldsymbol{\alpha}_n$ and $\epsilon_n := \|\mathbf{r}_n\|^2$, update the residual
11:    If $\epsilon_n > \epsilon_{n-1}$ quit
12: end for
```

III. Results

This section illustrates the improvements obtained by multiple Dictionary Update Cycles (DUC) and a Coefficient Reuse OMP (CoefROMP). We present experiments that demonstrate each of these ideas separately, and then combine them in an image denoising scheme. As the performance gain is dependent on the choice of various parameters, we provide illustrative examples accompanied by a discussion of how the performance changes with parameters.

A. Multiple Dictionary Update Cycles (DUC)

In the following experiments we use a collection of seventeen well-known standard images, including barbara, cameraman, jetplane, lena, mandril, and peppers. We extract $60,000$ patches from these images, $50,000$ of which are used for training and the remaining $10,000$ are used to test the reconstruction accuracy of the trained dictionary and the generalization performance. We subtract the mean from all patches, so as to eliminate the need to use a coefficient on the constant offset of the data vector. The quality of the representation is dependent on the number of atoms, n, and the number of non-zero coefficients, k. In our experiments, these two parameters are chosen as related to the signal dimension d by $n = 3d$ and $k = round(d/10)$. Also, the dictionary is initialized with samples from the training data.

Figure 1 shows plots of the RMSE versus iterations[3] during the dictionary learning for both the training and the testing signals. This figure illustrates the improvement in convergence when two or four cycles of the dictionary updates are performed in each iteration. The comparisons are based on the original K-SVD code[4]. In this example, 8×8 image patches are used (thus $d = 64$). A 64×192 dictionary is trained with a representation cardinality set to $k = 6$. As can be seen, both training and testing data benefit from the several update cycles, with a small computational cost. Much of the gain occurs in the first several iterations and this continues to carry forward through the remaining iterations. Note that running the algorithms with many iterations leads to nearly the same results – this suggests that using several DUCs does not lead to a different "steady-state" solution, but rather to a faster convergence to this solution.

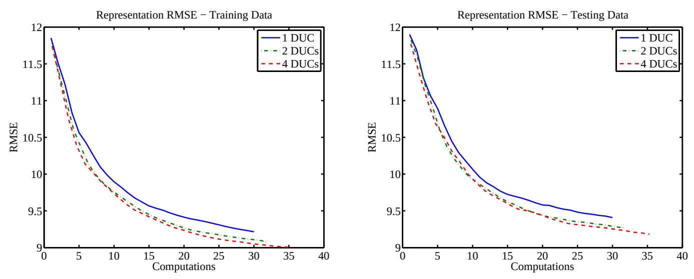

Figure 1. Training (left) and testing (right)representation's RMSE versus computations for the K-SVD algorithm with multiple DUCs.

Now we describe briefly how this improvement depends on the primary parameters: Starting with a 2D-DCT initial dictionary leads to improved convergence and a slightly smaller gain by multiple DUCs. In addition, there is a slight reduction in the relative amount of improvement from multiple DUCs as signal length, cardinality and dictionary size increases. However, the gain from multiple DUCs increases when there are fewer training samples, possibly because the effect on the dictionary training becomes more significant. When applying the multiple dictionary update cycles via the MOD approach (see Section 2), a similar gain is seen, and thus we chose to present only the K-SVD form.

[3]In Figures 1 and 2 the x-axis is "computation-normalized", i.e., the reference method of choice is shown versus iterations, while the modified algorithm is shown with a scaled x-axis to reflect the relative change in computations, measured by counting multiplications.

[4]www.cs.technion.ac.il/˜ronrubin/software.html

B. Coefficient Reuse OMP

To illustrate CoefROMP, a 15×15 image patch size was used, which corresponds to vectors of length 225. The signal dimension was increased from 64 to 225 in order to better demonstrate the advantages of CoefROMP. A 225×675 dictionary was trained, the cardinality was set $k = 23$ and $50,000$ training patches were used in the dictionary learning. Figure 2 illustrates the significant improvement in the RMSE (and run-time) from CoefROMP as compared to OMP for the training and the testing signals, for the first 30 iterations of the dictionary learning process. The CoefROMP is started after the first iteration since it needs previous iteration's results. In this experiment, CoefROMP has used an average of 6 iterations when the augmentation is in groups of 8, rather than 23 iterations required by OMP. The figure shows that training the dictionary with CoefROMP leads to a similar improvement for both the training and test data.

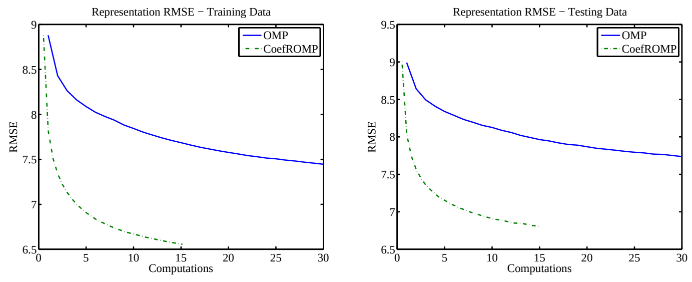

Figure 2. Training (left) and testing (right) representation's RMSE versus computations for the K-SVD algorithm with OMP and CoefROMP.

C. Image denoising

We turn to show the improvement achieved by applying both multiple DUCs and CoefROMP together, and this is demonstrated through the application of image denoising. The experiments here follow the methodology given in [12]. In this application there is no distinction between training and testing data, as we train the dictionary on patches extracted from the noisy image itself. Also, the pursuit is done with an error stopping criterion ($\epsilon = 1.1\sigma$), as in [12], implying that we use the CoefROMP version suited for this mode. We show the results obtained for the image Barbara contaminated by an additive Gaussian noise with $\sigma = 25$. Other tests with other images lead to similar results, and are thus omitted.

Figure 3 shows the PSNR – the denoising results – of the standard K-SVD image denoising algorithm as a function of the iteration. We show the results versus iterations, since the complexity evaluations are harder to obtain. That said, we remind the reader that using several DUCs hardly change the computational load, while using CoefROMP leads to substantial computational saving. We tested two patch sizes – 8×8 (left) and 16×16 (right) pixels. The training was done on all overlapping patches in the image, using $\lambda = 30/\sigma$, and building a 4-times redundant dictionary, initialized with the redundant DCT. This Figure illustrates that the shift from the OMP to CoefROMP is worthwhile and leads to a substantial improvement in convergence speed. Similarly, using more than one round of updates for the dictionary atoms (in this test we implemented the K-SVD approach, but the MOD alternative works comparably well) is leading to a further improvement. Figure 3 demonstrates that most of the benefit in the training/denoising is obtained in fewer iterations for CoefROMP than with OMP.

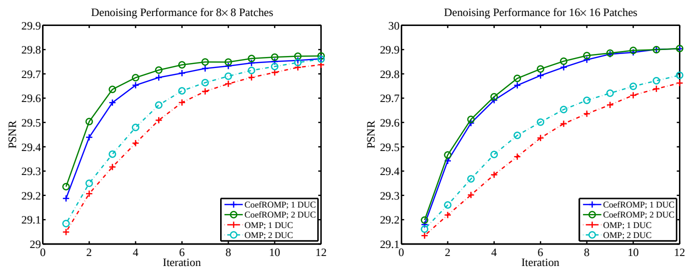

Figure 3. Denoising performance with the K-SVD algorithm, demonstrating OMP versus CoefROMP, and using 1 or 2 dictionary update stages. We consider patches of size 8×8 (left) and 16×16 (right) pixels.

IV. Conclusion

Dictionary learning is a central step in employing a sparsity-based model for various data processing tasks. Therefore, the speed of such learning algorithms is key in making many algorithms more efficient and thus more practical. In this paper we propose two simple yet effective modifications for the K-SVD and the MOD learning algorithms, both shown to lead to speedup in the convergence of the learning process. We believe that the proposed modifications are of relevance and importance to many other dictionary learning techniques.

References

[1] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*, Springer, 2010.
[2] B.A. Olshausen and D.J. Field, Sparse coding with an overcomplete basis: A strategy employed by V1? Vision Research, 37(23):3311–3325, 1997.
[3] M.S. Lewicki and T.J. Sejnowski, Learning overcomplete representations, Neural Computation, 12(2):337–365, 2000.
[4] S. Lesage, R. Gribonval, F. Bimbot, and L. Benaroya, Learning unions of orthonormal bases with thresholded singular value decomposition, ICASSP, 5:293–296, 2005.
[5] M. Yaghoobi, T. Blumensath, and M.E. Davies, Dictionary learning for sparse approximations with the majorization method, IEEE Trans. on Signal Processing, 57(6):2178–2191, 2009.
[6] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, Online learning for matrix factorization and sparse coding, JMLR, 11:19–60, 2010.
[7] K. Skretting and K. Engan, Recursive least squares dictionary learning algorithm, IEEE Trans. on Signal Processing, 58(4):2121–2130, 2010.
[8] I. Tosic and P. Frossard, Dictionary learning: what is the right representation for my signal? IEEE Signal Proc. Magazine, 28(2):27–38, 2011.
[9] K. Engan, S.O. Aase, and J.H. Hakon-Husoy, Method of optimal directions for frame design, ICASSP, 5:2443–2446, 1999.
[10] K. Engan, K. Skretting, J. Hakon-Husoy, Family of iterative LS-based dictionary learning algorithms, ILS-DLA, for sparse signal representation, Digital Signal Processing, Vol. 17(1), pp. 32–49, 2007.
[11] M. Aharon, M. Elad, A. Bruckstein, K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation, IEEE Trans. on Signal Processing, Vol. 54(11):4311–4322, 2006.
[12] M. Elad and M. Aharon, Image denoising via sparse and redundant representations over learned dictionaries, IEEE Trans. on Image Proc., 15(12):3736–3745, 2006.
[13] D. Needell and J.A. Tropp, CoSAMP: Iterative signal recovery from incomplete and inaccurate samples, ACHA, 26, 2008.
[14] W. Dai and O. Milenkovic, Subspace pursuit for compressive sensing, IEEE Trans. on Information Theory, 55(5):2230–2249, 2009.
[15] R. Rubinstein, M. Zibulevsky, M. Elad, Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit, CS Technical Report, Technion - Israel Institute of Technology, 2009.