Learning Overcomplete Dictionaries Based on Atom-by-Atom Updating

Mostafa Sadeghi*, Massoud Babaie-Zadeh, *Senior Member, IEEE*, and Christian Jutten, *Fellow, IEEE*

***Abstract*—A dictionary learning algorithm aims to learn a set of atoms from some training signals in such a way that each signal can be approximated as a linear combination of only a few atoms. Most dictionary learning algorithms use a two-stage iterative procedure. The first stage is to sparsely approximate the training signals over the current dictionary. The second stage is the update of the dictionary. In this paper we develop some atom-by-atom dictionary learning algorithms, which update the atoms sequentially. Specifically, we propose an efficient alternative to the well-known K-SVD algorithm, and show by various experiments that the proposed algorithm has much less execution time compared to K-SVD while its results are better. Moreover, we propose a novel algorithm that instead of alternating between the two dictionary learning stages, performs only the second stage. While in K-SVD each atom is updated along with the *nonzero entries* of its associated row vector in the coefficient matrix (which we name it its profile), in the new algorithm, each atom is updated along with the *whole entries* of its profile. As a result, contrary to K-SVD, the support of each profile can be changed while updating the dictionary.**

To further accelerate the convergence of this algorithm and to have a control on the cardinality of the representations, we then propose its two-stage counterpart by adding the sparse approximation stage. We evaluate the performance of the proposed algorithms by performing two sets of experiments. The first set is the reconstruction of a true underlying dictionary, while the second set is designing a sparsifying dictionary for a certain class of signals. The results emphasize on the promising performance of the proposed algorithms.

***Index Terms*—Sparse approximation, dictionary learning, compressive sensing, K-SVD.**

I. Introduction

A. Sparse Signal Approximation

SPARSE decomposition of signals based on some basis functions has attracted a lot of attention during the last decade [1]. The problem consists in approximating a given signal as a linear combination of as few as possible basis functions. In this context, each basis function is called an *atom* and their collection as the columns of a matrix is called *dictionary* [2]. The dictionary may be overcomplete, i.e., the number of atoms may be (much) more than the dimension of the atoms. Specifically, let $\mathbf{y} \in \mathbb{R}^n$ be the signal which is going to be sparsely represented in the dictionary $\mathbf{D} \in \mathbb{R}^{n\times K}$ with $K > n$. This amounts to solve the following problem,

$$P_0: \quad \min_{\mathbf{x}} \ \|\mathbf{x}\|_0 \quad \text{subject to} \quad \mathbf{y} = \mathbf{D}\mathbf{x}, \tag{1}$$

where $\|.\|_0$ stands for the so-called ℓ_0 pseudo-norm which counts the number of nonzero elements. Because of the combinatorial nature of P_0 [1] that makes it hard to solve, especially in high dimensions, some relaxation methods have been introduced. A well-known relaxation approach is to use the ℓ_1 norm instead of the ℓ_0 pseudo-norm. This leads to the following convex P_1 problem which is known as Basis Pursuit (BP) [3]

$$P_1: \quad \min_{\mathbf{x}} \ \|\mathbf{x}\|_1 \quad \text{subject to} \quad \mathbf{y} = \mathbf{D}\mathbf{x}. \tag{2}$$

In practical situations instead of the exact equality $\mathbf{y} = \mathbf{D}\mathbf{x}$, the constraint $\|\mathbf{y} - \mathbf{D}\mathbf{x}\|_2 \le \epsilon$ is used, which has a denoising and stabilizing effect [4], [5]. In this case, usually the term *approximation* is used instead of *representation*.

Many algorithms have been introduced to solve the problem of finding the sparsest approximation of a signal in a given overcomplete dictionary (as a good review, see [6]). These methods can be classified into two general categories, *greedy methods* such as Orthogonal Matching Pursuit (OMP) [7], Stagewise Orthogonal Matching Pursuit (StOMP) [8], Compressive Sampling Matching Pursuit (CoSaMP) [9], Subspace Pursuit (SP) [10], and *relaxation methods*, which replace the combinatorial P_0 problem with a tractable one, e.g., P_1 problem. Iterative Shrinkage-Thresholding (IST) [11], [12], [13], Iterative Hard-Thresholding (IHT) [14], Iteratively Reweighted Least Squares (IRLS) [15], Smoothed ℓ_0 (SL0) [16], and interior-point methods [17] are some examples of the second category. Greedy algorithms successively choose the appropriate atoms of the dictionary that result in the greatest improvement in the quality of the approximation. These algorithms benefit from having high speed, but their accuracy is usually less than that of the second category.

Various image processing tasks (e.g., denoising, compression, inpainting, zooming) [18], Blind Source Separation (BSS) in underdetermined mixtures [19], Compressive Sensing (CS) [20], [21], decoding real field codes [22], linear regression and variable selection [23] are some applications where the sparse approximation approach has already been applied.

B. Learning overcomplete dictionaries

For a given class of signals, e.g., class of natural facial images, the dictionary should have the capability of sparsely

This work was supported in part by Iran National Science Foundation (INSF) under Contract 91004600, and by European project 2012-ERC-AdG-320684 CHESS.

M. Sadeghi and M. Babaie-Zadeh are with the Electrical Engineering Department, Sharif University of Technology, Tehran, Iran (e-mail: m.saadeghii@gmail.com; mbzadeh@yahoo.com).

C. Jutten is with the GIPSA-Lab, Department of Images and Signals, University of Grenoble and Institut Universitaire de France, France (e-mail: Christian.Jutten@inpg.fr).

representing the signals. To this aim, atoms of the dictionary should capture the most salient features of the signals. In some applications there are predefined and universal dictionaries which are known to be well-matched to the contents of the given class of signals, for example the overcomplete DCT dictionary for the class of natural images. These *non-adaptive* dictionaries are appealing because of their simplicity and in some cases their fast computations. However, *learning* the atoms from a set of training signals would result in dictionaries with the capability of better matching the contents of the signals. In this way, the *adaptive* dictionaries would outperform the *non-adaptive* ones in many applications such as image denoising [24], image compression [25], classification tasks [26], and so on.

Most dictionary learning algorithms are indeed a generalization of the K-means clustering algorithm [27]. While in K-means each training signal is forced to use only one atom (cluster center) as its approximation, in the dictionary learning problem each signal is allowed to use more than one atom provided that it uses as few as possible atoms. The approach of K-means to optimize a set of atoms (called codebook) is to iteratively perform two stages [28]. In the first stage, known as the clustering stage, training signals are assigned to their nearest atoms (usually in the sense of the ℓ_2 norm). The second stage is the update of the atoms in which each atom is updated as the average of the signals in its cluster. This procedure is repeated several times. As a generalization of this approach, dictionary learning algorithms iteratively perform the two stages of *sparse approximation* and *dictionary update*. In the first stage, which is actually the clustering of the signals into a union of subspaces, the sparse approximations of the signals are computed using the current dictionary. The second stage is the update of the dictionary.

Up to our best knowledge, most dictionary learning algorithms differ mainly in the way of updating the dictionary [27], [29], [30]. Some algorithms such as K-Singular Value Decomposition (K-SVD) [27] are based on updating the atoms one-by-one, while some others such as Method of Optimal Directions (MOD) [29] update the whole set of atoms at once.

In another point of view, dictionary learning algorithms are divided into two groups: *online* [31], [32], [33] and *batch-based* [29], [27], [30] algorithms. Online algorithms continuously update the dictionary using only one (or a mini batch of) training data. Because of this, they enjoy from having the capability of handling very large sets of signals, which is common in, for example, image processing tasks (see the experiments of [31] on a high-resolution image). Recursive Least Squares dictionary learning algorithm (RLS-DLA) [32] is an example of online algorithms, which can be considered as a generalization of the McQueen variant of K-means [34]. One major benefit of online algorithms is to learn gradually, which is denoted as the *scaling the past data* in [31], and by using a *forgetting factor* in [32]. Batch-based algorithms use the whole set of training data to update the dictionary. This increases the computational burden of these algorithms in high dimensions. However, an advantage of batch-based algorithms over online algorithms is their low computational loads and memory requirements in relatively low dimensions. This is due to the fact that at each sparse approximation stage, the whole set of signals are approximated in the same dictionary. So, the sparse coding algorithm can be optimized to avoid performing common operations. Batch-OMP [35] is an example of such optimized sparse coding algorithms.

C. Our contributions and the structure of the paper

The focus of this paper is on the algorithms that update atoms one-by-one. We first propose an efficient way of updating each atom along with its associated row vector in the coefficient matrix: we call this idea parallel atom-updating. From this, we propose an efficient and fast alternative for K-SVD: we call this algorithm PAU-DL, and show by various experiments that it performs better than K-SVD while its computational burden is substantially lower than it. We then propose a novel method of dictionary learning whose structure is different from the existing algorithms. Specifically, instead of alternating between the two dictionary learning stages, the proposed algorithm performs only the dictionary update stage. We call this algorithm OS-DL. In each alternation of OS-DL, each atom is updated along with the whole entries of its associated row vector in the coefficient matrix. The main differences between the new algorithm and the current atom-by-atom dictionary learning algorithms, are as follows. Firstly, the new algorithm does not explicitly perform the sparse approximation stage. Secondly, in this algorithm the support of each row vector of the coefficient matrix is allowed to change.

To further accelerate the convergence of this method and have a control on the cardinality of the representations, we propose to add the sparse approximation stage to this algorithm, and hence we derive another new algorithm, called APrU-DL. OS-DL and APrU-DL use the parallel atom-updating approach. Experimental results show the promising performance of the proposed algorithms relative to K-SVD.

The paper is organized as follows. Section II is devoted to the formulation of the dictionary learning problem. In Section III we describe our proposed algorithms. This section begins with introducing the parallel atom-updating approach; a general idea for updating the atoms of the dictionary one-by-one. Using this idea in the dictionary update stage, we derive the *parallel atom-updating dictionary learning* (PAU-DL) algorithm. We then propose the two novel atom-by-atom dictionary learning algorithms, i.e., OS-DL and APrU-DL. Section IV presents some experimental results.

D. Notation

We use the following notations. For vector and matrix valued quantities we use small and capital bold face characters, respectively. The superscript "T" denotes vector or matrix transpose. We denote the ith column vector of a matrix $\mathbf{X}$ as $\mathbf{x}_i$, and its ith row vector as $\mathbf{x}_{[i]}^T$. $\|\mathbf{X}\|_F = \sqrt{\sum_{ij} x_{ij}^2}$ and $\|\mathbf{x}\|_p = (\sum_i |x_i|^p)^{\frac{1}{p}}$ denote the Frobenius norm of the matrix $\mathbf{X}$, and the ℓ_p norm of the vector $\mathbf{x}$, respectively. In an iterative algorithm, we denote a parameter in the kth iteration with the iteration number in parenthesis, e.g., $\mathbf{d}^{(k)}$. For a matrix $\mathbf{X}$ with L columns and $\omega \subseteq \{1, 2, \dots, L\}$, we define $\mathbf{X}(:, \omega)$ as

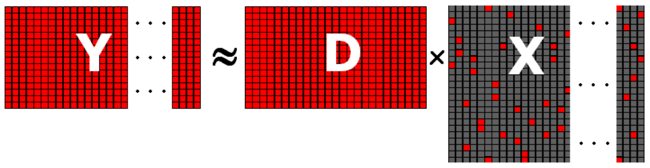

Fig. 1. A dictionary learning problem is to factorize a training data matrix $\mathbf{Y}$ as $\mathbf{Y} \simeq \mathbf{D}\mathbf{X}$ with $\mathbf{X}$ a sparse-column matrix.

a matrix containing those columns of $\mathbf{X}$ that their indices are in ω. Also, $\mathbf{x}(\omega)$ is a vector containing those entries of $\mathbf{x}$ that are indexed by ω. Finally, $|\omega|$ denotes the cardinality of ω, that is, its number of entries.

II. Dictionary Learning Problem

Let $\{\mathbf{y}_i\}_{i=1}^{L}$ be a set of L training signals in $\mathbb{R}^n$. Putting these signals as the columns of the matrix $\mathbf{Y}$, the general dictionary learning problem is then to find a sparsifying dictionary, $\mathbf{D}$, by solving the following problem

$$\min_{\mathbf{D}\in\mathcal{D},\mathbf{X}\in\mathcal{X}} \|\mathbf{Y}-\mathbf{D}\mathbf{X}\|_F^2, \tag{3}$$

where $\mathcal{D}$ and $\mathcal{X}$ are admissible sets of the dictionary and the coefficient matrix, respectively. $\mathcal{D}$ is usually defined as the set of all dictionaries with unit column-norms. Since we require that each signal has a sparse approximation, $\mathcal{X}$ is the set of all matrices $\mathbf{X}$ with sparse columns (see Fig. 1). The sparsity measure can be, for example, the non-convex ℓ_0 pseudo-norm, or the convex ℓ_1 norm. The general approach to solve (3) is to use alternating minimization over $\mathbf{X}$ and $\mathbf{D}$, i.e., by fixing $\mathbf{D}$, the objective function is minimized over $\mathbf{X}$, and vice versa.

A. Sparse Approximation

With a fixed $\mathbf{D}$, the minimization of (3) with respect to $\mathbf{X}$ is equivalent to sparsely approximating the training signals over $\mathbf{D}$. Among the various sparse coding algorithms, OMP (or its variants) is very appealing for this stage. This is due to two reasons. The first reason is the high speed of OMP compared to relaxation algorithms. The second one is its capability to be efficiently implemented in the batch mode. This fact along with the use of Cholesky factorization result in the significant acceleration of OMP, which leads to Batch-OMP algorithm [35].

Another option, which we use in this paper, is the IST algorithms. Although the speed of these algorithms is lower than that of OMP, their accuracy is better.

B. Dictionary Update

As stated previously, dictionary learning algorithms differ mainly in the way they perform the second stage, i.e., the dictionary update. In this stage, some desired properties about the dictionary may be applied, such as having bounded Frobenius norm [30], and having bounded self-coherence (i.e., pairwise similarity of the atoms) [36]. However, the constraint of having unit column-norms is usually used in this stage.

In spite of its name, in the dictionary update stage some information about the coefficient matrix may be updated, too. This information may be the nonzero coefficients of $\mathbf{X}$ (as in K-SVD and [37]) or the whole coefficients of it (as in two of our suggested algorithms). Furthermore, as stated previously, one of the differences among various dictionary learning algorithms is that they either update the whole set of atoms at once (as in MOD) or each atom separately (as in K-SVD).

C. MOD and K-SVD

MOD [29] is one of the simplest dictionary learning algorithms which firstly finds the unconstrained minimum of $\|\mathbf{Y}-\mathbf{D}\mathbf{X}\|_F^2$ and then projects the solution onto the set $\mathcal{D}$. This leads to the closed-form expression

$$\mathbf{D} = \mathbf{Y}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}, \tag{4}$$

followed by normalizing the columns of $\mathbf{D}$.

K-SVD [27] is one of the most successful dictionary learning algorithms. In its dictionary update stage, only one atom is updated at a time. Moreover, while updating each atom, the nonzero entries in the associated row vector of $\mathbf{X}$ are also updated. In other words, only those signals that have used a specific atom participate in updating that atom. This is in accordance with the approach of K-means in which each atom is updated using its own cluster signals. As stated in [27], this also prevents each row vector in $\mathbf{X}$ to be filled and thus violating the sparsity constraint of the coefficient matrix.

Assume that we want to update the ith atom, $\mathbf{d}_i$, along with the nonzero entries of $\mathbf{x}_{[i]}^T$, the ith row of $\mathbf{X}$. We define $\omega_i = \left\{j : \mathbf{x}_{[i]}^T(j) \neq 0\right\}$ as the support of $\mathbf{x}_{[i]}^T$. Then the problem of updating $\mathbf{d}_i$ along with $\mathbf{x}_{[i]}^T(\omega_i)$ amounts to solve the following minimization problem

$$\min_{\mathbf{d},\mathbf{x}_r} \|\mathbf{E}_i^r - \mathbf{d}\mathbf{x}_r^T\|_F^2 \quad \text{subject to} \quad \|\mathbf{d}\|_2^2 = 1, \tag{5}$$

where $\mathbf{E}_i^r = \mathbf{E}_i(:,\omega_i)$, in which $\mathbf{E}_i = \mathbf{Y} - \sum_{j\neq i}\mathbf{d}_j\mathbf{x}_{[j]}^T$ denotes the approximation error matrix when $\mathbf{d}_i$ is removed, and $\mathbf{x}_r^T$ is a row vector of length $|\omega_i|$. The above problem is in fact equivalent to finding the closest rank-1 approximation to $\mathbf{E}_i^r$, which can be easily solved via SVD of $\mathbf{E}_i^r$. For more details refer to [27].

III. Atom-by-Atom dictionary update

In this section we propose three algorithms that update the dictionary atom-by-atom. We call the ith row vector of the coefficient matrix, the *profile* of the ith atom, because this vector indicates which signals use this atom in their representations. We first introduce the idea of *parallel atom-updating* and from which we then propose the *parallel atom-updating dictionary learning* (PAU-DL) algorithm, which is introduced to overcome the computational burden of K-SVD. We then proceed with introducing a novel method for dictionary learning which is based on performing only the second stage of the general dictionary learning procedure, and thus we name it *One-Stage Dictionary Learning* (OS-DL). In OS-DL, the support of each profile is allowed to change during the update of the dictionary. To further accelerate the convergence rate of OS-DL (as will be shown in the simulations) and control the sparsity level of the representations, we propose an algorithm, which we call *Atom-Profile Updating Dictionary Learning* (APrU-DL), which in fact adds the first stage of dictionary learning to OS-DL. In all of these algorithms, we choose $\mathcal{D}$ to be the set of all unit column-norm dictionaries in $\mathbb{R}^{n\times K}$.

A. Parallel Atom-Updating Dictionary Learning (PAU-DL)

The main drawback of K-SVD is its computational burden especially in high dimensions. This is due to performing SVD for atom updating. An alternative way of solving (5) is to use the idea of alternating minimization [35]. In other words, (5) is alternatively minimized over $\mathbf{d}$ and $\mathbf{x}_r$. A few (e.g., 3) alternations give a fast approximation to SVD. The resulting algorithm is known as the Approximate K-SVD (AK-SVD) [35]. Although performing more alternations gives a better approximation, the average performance will not exceed the performance of the exact solution, i.e., via SVD.

In this subsection we describe a different way of performing alternating minimization to update the atoms and their profiles. To this aim, consider the overall error matrix,

$$\mathbf{E} = \mathbf{Y} - (\mathbf{A}_1 + \mathbf{A}_2 + \ldots + \mathbf{A}_K),\ \forall i:\ \ \mathbf{A}_i = \mathbf{d}_i\mathbf{x}_{[i]}^T. \quad (6)$$

In K-SVD (or AK-SVD), in order to update (the nonzero columns of) for example $\mathbf{A}_i$, the updated versions of $\mathbf{A}_1, \ldots, \mathbf{A}_{i-1}$ are used to compute $\mathbf{E}_i$, while $\mathbf{A}_{i+1}, \ldots, \mathbf{A}_K$ have not been yet updated. Keeping this point in mind, we propose to update the atoms in parallel. In other words, instead of fully updating each $\mathbf{A}_i$ by performing "A" alternations between $\mathbf{d}_i$ and $\mathbf{x}_r$, "A" alternations are performed in such a way that in each alternation all of $\mathbf{A}_i$'s are partially updated (using only one alternation). In this way, in the subsequent alternations, all $\mathbf{A}_i$'s have been partially updated. As our experimental results in Section IV suggest, parallel updating of the atoms may result in further accelerating the convergence rate and the quality of the final results. In other words, the new algorithm outperforms K-SVD, which is based on exact solving of the rank-1 approximation problem.

To update each $\mathbf{A}_i$, we need to compute the error matrix $\mathbf{E}_i$. It can be easily seen that this matrix can be updated as follows. The overall error matrix is firstly computed as $\mathbf{E} = \mathbf{Y} - \mathbf{DX}$ using the current dictionary and coefficient matrix. Then $\mathbf{E}_i = \mathbf{E} + \mathbf{A}_i$ and after updating $\mathbf{A}_i$ to $\mathbf{A}_i^{(new)}$, the error matrix $\mathbf{E}$ is updated as $\mathbf{E} = \mathbf{E}_i - \mathbf{A}_i^{(new)}$.

Algorithm 1 gives a description of the dictionary update based on parallel atom-updating. PAU-DL is an alternative to K-SVD that uses this atom-updating procedure. Algorithm 2 gives a complete description of PAU-DL and AK-SVD. By Batch-OMP$(\mathbf{Y}, \mathbf{D}, \tau)$ we mean the sparse approximation of $\mathbf{Y}$ in $\mathbf{D}$ and with threshold τ. Depending on the application at hand, τ may be the threshold on the approximation error or the maximum allowed number of atoms in the sparse approximation of each training signal.

Considering Algorithm 2, we see that in K-SVD: $A = 1$, with a large B, in AK-SVD: $A = 1$, $B = 3$, and in PAU-DL: $A = 3$, $B = 1$. Here the reader may suggest to use $A = 3$, $B = 3$. However, as we saw in our simulations, the results are similar to those of PAU-DL, yet with a higher computational load.

Algorithm 1 Parallel Atom-Updating

```
1: $\mathbf{E} = \mathbf{Y} - \mathbf{DX}$
2: for $a = 1, \ldots, A$ do
3:     for $i = 1, \ldots, K$ do
4:         $\mathbf{E}_i = \mathbf{E} + \mathbf{d}_i\mathbf{x}_{[i]}^T$
5:         Update $\mathbf{x}_{[i]}^T$
6:         Update $\mathbf{d}_i$
7:         $\mathbf{E} = \mathbf{E}_i - \mathbf{d}_i\mathbf{x}_{[i]}^T$
8:     end for
9: end for
```

Algorithm 2 AK-SVD ($A = 1$, $B = 3$) and PAU-DL ($A = 3$, $B = 1$)

```
1: Task: Learning a dictionary for $\mathbf{Y}$
2: Initialization: $\mathbf{D} = \mathbf{D}^{(0)}$
3: Repeat:
4: Sparse Approximation: $\mathbf{X}$ = Batch-OMP$(\mathbf{Y}, \mathbf{D}, \tau)$
5: Dictionary Update: set $\mathbf{E} = \mathbf{Y} - \mathbf{DX}$
6: for $a = 1, \ldots, A$ do
7:     for $i = 1, \ldots, K$ do
8:         $\mathbf{E}_i = \mathbf{E} + \mathbf{d}_i\mathbf{x}_{[i]}^T$
9:         $\mathbf{E}_i^r = \mathbf{E}_i(:, \omega_i)$ where $\omega_i = \left\{j:\ \mathbf{x}_{[i]}^T(j) \neq 0\right\}$
10:        for $b = 1, \ldots, B$ do
11:            $\mathbf{d}_i = \mathbf{E}_i^r\mathbf{x}_{[i]}^T(\omega_i)$
12:            $\mathbf{d}_i = \mathbf{d}_i/\|\mathbf{d}_i\|_2$
13:            $\mathbf{x}_{[i]}^T(\omega_i) = \mathbf{d}_i^T\mathbf{E}_i^r$
14:        end for
15:        $\mathbf{E} = \mathbf{E}_i - \mathbf{d}_i\mathbf{x}_{[i]}^T$
16:    end for
17: end for
18: Until convergence
```

At first glance, one may think that PAU-DL differs from AK-SVD by simply changing the operation orders. This, however, is not just a simple rescheduling. As explained earlier, the main idea behind PAU-DL is to *partially* update each atom before moving to the next atom. In this way, we have in our disposal more reliable updates of $\mathbf{A}_i$'s in order to compute the error matrix associated with the atom we are going to update; see (6). Moreover, as will be seen in Section IV, PAU-DL has a sufficiently better performance than AK-SVD and even original K-SVD.

B. One-stage Dictionary Learning (OS-DL)

In [27], after developing the main steps of K-SVD algorithm, the authors asserted that "*Here one might be tempted to suggest skipping the step of sparse coding and using only updates of columns in* $\mathbf{D}$*, along with their coefficients, applied in a cyclic fashion, again and again. This, however, will not work well, as the support of the representations will never be changed, and such an algorithm will necessarily fall into a local minimum trap.*"

In this subsection, we describe OS-DL algorithm. This algorithm ignores the sparse approximation stage and perform dictionary learning by updating atoms along with their profiles in a cyclic fashion. To prevent the above mentioned problem, a sparsity constraint on the profiles of the atoms is introduced, which at the same time allows the support of each profile to change (and probably not getting trapped into a local minimum) and prevents each profile to be filled. Allowing the support of each profile to change lets each atom adaptively find its cluster members (see Subsection III-C). Each atom and its profile are updated by solving the following minimization problem

$$\forall i: \ \left\{\mathbf{d}_i, \mathbf{x}_{[i]}\right\} = \underset{\mathbf{d},\mathbf{z}}{\operatorname{argmin}} \ \frac{1}{2}\|\mathbf{E}_i - \mathbf{d}\mathbf{z}^T\|_F^2 + \lambda\|\mathbf{z}\|_1, \tag{7}$$

subject to the constraint $\|\mathbf{d}_i\|_2^2 = 1$. Note that if we set $\lambda = 0$ and restrict $\mathbf{E}_i$ and $\mathbf{x}_{[i]}^T$ to those training data that have used $\mathbf{d}_i$ in their approximations, then equation (7) is exactly the one used in K-SVD. Note also that the above problem is indeed a regularized rank-1 approximation of $\mathbf{E}_i$. It is worth mentioning that in OS-DL there is no control on the cardinalities of the representations, and indeed each one may have a different cardinality. However, in the next algorithm described in Subsection III-C, this problem is avoided by performing the sparse approximation stage.

To solve (7), we use alternating minimization. At the first stage, we update $\mathbf{z}$ with a fixed $\mathbf{d}$, and at the second stage, we update $\mathbf{d}$ using the previously updated $\mathbf{z}$. The update formula for $\mathbf{z}$ is obtained by solving (7) with $\mathbf{d}$ being fixed. Interestingly, this leads to a simple closed-form formula [38]. To see this, note that (7) can be de-coupled for each entry of $\mathbf{z}$. Then, we should solve L scalar problems of the form

$$\min_z \ f(z) = \frac{1}{2}\|\mathbf{e} - z\mathbf{d}\|_2^2 + \lambda|z|, \tag{8}$$

where $\mathbf{e}$ is the corresponding column of $\mathbf{E}_i$. Since the last term in $f(z)$ is non-differentiable, we require that zero is included in the subgradient of $f(z)$, i.e., $0 \in \partial f(z)$. This results in

$$0 = -\mathbf{d}^T(\mathbf{e} - z\mathbf{d}) + \lambda \mathrm{sgn}(z). \tag{9}$$

Now, considering the assumption $\|\mathbf{d}\|_2^2 = 1$, the final solution is found using the soft-thresholding operation

$$z = \mathrm{sgn}(\mathbf{e}^T\mathbf{d}).\max(0, |\mathbf{e}^T\mathbf{d}| - \lambda) = \mathcal{S}_\lambda(\mathbf{e}^T\mathbf{d}), \tag{10}$$

where $\mathcal{S}_\lambda(a)$ is the well-known soft-thresholding function [38] defined as follows

$$\mathcal{S}_\lambda(a) \triangleq \begin{cases} a - \lambda & \text{if } a > \lambda \\ 0 & \text{if } |a| \leq \lambda \\ a + \lambda & \text{if } a < -\lambda \end{cases}. \tag{11}$$

The final solution is $\mathbf{x}_{[i]} = \mathcal{S}_\lambda(\mathbf{E}_i^T\mathbf{d}_i)$, where $\mathcal{S}_\lambda(.)$ acts component-wise.

The update formula for $\mathbf{d}$ is also of the simple form

$$\mathbf{d}_i = \mathbf{E}_i\mathbf{x}_{[i]}, \tag{12}$$

followed by a normalization. Like PAU-DL, in OS-DL the atoms are updated in parallel. A few iterations ($A = 3$ in our experiments) of the alternating minimization is sufficient to obtain $\mathbf{x}_{[i]}^T$ and $\mathbf{d}_i$. This process is repeated several iterations until a stopping condition is satisfied. The final algorithm is summarized in Algorithm 3.

Algorithm 3 OS-DL

```
1:  Task: Learning a dictionary for $\mathbf{Y}$
2:  Initialization: $\mathbf{D} = \mathbf{D}^{(0)}$ and $\mathbf{X} = \mathbf{X}^{(0)}$
3:  The main loop: Set $\mathbf{E} = \mathbf{Y} - \mathbf{D}\mathbf{X}$
4:  for $a = 1, \dots, A$ do
5:      for $i = 1, \dots, K$ do
6:          $\mathbf{E}_i = \mathbf{E} + \mathbf{d}_i\mathbf{x}_{[i]}^T$
7:          $\mathbf{x}_{[i]} = \mathcal{S}_\lambda(\mathbf{E}_i^T\mathbf{d}_i)$
8:          $\mathbf{d}_i = \mathbf{E}_i\mathbf{x}_{[i]}$
9:          $\mathbf{d}_i = \mathbf{d}_i/\|\mathbf{d}_i\|_2$
10:         $\mathbf{E} = \mathbf{E}_i - \mathbf{d}_i\mathbf{x}_{[i]}^T$
11:     end for
12: end for
```

How does it work?

According to (10), only those columns of $\mathbf{E}_i$ are represented by $\mathbf{d}_i$ that are sufficiently similar to it. After finding these signals, the cluster members of $\mathbf{d}_i$ are actually found. Similar to K-means, each atom is updated as the (weighted) average of its cluster members, followed by a normalization. Indeed, each atom is updated as a sparse linear combination of the columns of the error matrix, see (12). In OS-DL, *a sparse coding-like stage* is implicitly performed[1]. This is because after updating the whole row vectors of the coefficient matrix, its columns are also updated. This way of sparse coding is very similar to an ordinary sparse coding algorithm that is based on updating the coefficients sequentially using a coordinate descent algorithm [39], [38].

C. Atom-Profile Updating Dictionary Learning (APrU-DL)

Recall that the idea of K-SVD to update each atom is indeed a generalization of K-means clustering algorithm. In other words, each atom is updated using only its cluster members, i.e., those signals that have used it in their representations. However, there is a main difference between these two algorithms. As stated in [27], since in K-means each training data uses only one atom, the atom updating problems are decoupled, while this is not the case in K-SVD. In other words, each training data belongs possibly to multiple atom clusters, thus the atom update problems are indeed coupled. So, we believe that restricting the support of the profiles to be fixed during atom updates, as done in K-SVD, is not well justified.

To overcome this problem, we propose to allow the support of each profile to change during the atom updating procedure. In this way, each atom adaptively finds its cluster members. This idea is used in OS-DL algorithm described in the previous section. In this section, we describe the two-stage counterpart of this algorithm by performing the sparse approximation stage, too. Contrary to OS-DL, by this method, the cardinality of the representations can be controlled. Moreover, as will be seen in the simulations, the new algorithm has a better

[1]Note that this differs from the usual *sparse approximation stage* performed in a typical dictionary learning algorithm. Actually, in (7), the sparsity constraint (i.e., the ℓ_1 norm penalty) is on the *rows* of the coefficient matrix, not on its columns. As explained in the text, we include this constraint to prevent each profile to be filled during the atom updating procedure.

Algorithm 4 Batch-FISTA($\mathbf{Y}$,$\mathbf{D}$,$\mathbf{X}^{(0)}$,λ)

```
1: Require: $\mathbf{Y}$, $\mathbf{D}$, $\mathbf{X}^{(0)} \in \mathbb{R}^{K\times L}$, $\lambda$
2: Initialization: $\mathbf{Z}^{(0)} = \mathbf{X}^{(0)}$, $c \leq 1/(2\lambda_{\max}(\mathbf{D}^T\mathbf{D}))$, $\mathbf{A} = \mathbf{D}^T\mathbf{D}$, $\mathbf{B} = \mathbf{D}^T\mathbf{Y}$, $\theta_0 = 1$
3: for $k = 0, 1, 2 \ldots$ do
4:     $\mathbf{X}^{(k+1)} = \mathcal{S}_{c.\lambda}(\mathbf{Z}^{(k)} - c\mathbf{A}\mathbf{Z}^{(k)} + c\mathbf{B})$
5:     $\theta_{k+1} = (1 + \sqrt{1 + 4\theta_k^2})/2$
6:     $\beta_k = (\theta_k - 1)/\theta_{k+1}$
7:     $\mathbf{Z}^{(k+1)} = \mathbf{X}^{(k+1)} + \beta_k(\mathbf{X}^{(k+1)} - \mathbf{X}^{(k)})$
8: end for
```

performance, both in convergence rate and quality of the results.

For the sparse approximation stage any sparse coding algorithm can be used. However, the relaxation-based methods have a better performance compared to greedy ones (to see this, we conduct an experiment in the next section). Among the relaxation-based methods, the soft-thresholding ones are probably more attractive, for both their simplicity and good performance. These methods target the following problem

$$\min_{\mathbf{D}\in\mathcal{D},\mathbf{X}} \frac{1}{2}\|\mathbf{Y} - \mathbf{D}\mathbf{X}\|_F^2 + \lambda_s\|\mathbf{X}\|_1, \tag{13}$$

where λ_s is a regularization constant. Here, we use the Fast Iterative Shrinkage-Thresholding (FISTA) algorithm [40] whose global convergence rate is much better than the ordinary IST algorithms, while preserving their simplicity. The batch-mode version of FISTA is shown in Algorithm 4, in which, $\lambda_{\max}(\mathbf{X})$ denotes the largest eigenvalue of $\mathbf{X}$. For initialization of the coefficient matrix in each alternation, we use its final estimate at the previous alternation. Once the support of each column of the coefficient matrix is found, we project the associated training signal onto the subspace spanned by the corresponding atoms of the dictionary. This process is called *debiasing*, and it is known to improve the results [41].

In order to update the dictionary in the second stage, we follow the same approach as in OS-DL algorithm. In other words, the problem of updating each atom together with its profile amounts to solve (7). Algorithm 5 gives a description of APrU-DL algorithm.

IV. Simulations

We evaluate the efficiency of our proposed algorithms with two sets of experiments. The first set of experiments is on synthetic data, where we aim to evaluate the capability of our algorithms in recovery of a known dictionary. To this aim, we generate a set of training signals, each as a linear combination using a different set of atoms from an underlying dictionary. We then give these training signals (after adding a certain amount of Gaussian noise) to each algorithm and compare the output dictionary to the original one. In this way, a dictionary learning algorithm should have the ability to extract the common features of the set of signals, which are actually the generative atoms. The second set of experiments is on an autoregressive (AR) signal, where there is no underlying dictionary, and we just evaluate the capability of the algorithms

Algorithm 5 APrU-DL

```
1: Task: Learning a dictionary for $\mathbf{Y}$
2: Initialization: $\mathbf{D} = \mathbf{D}^{(0)}$ and $\mathbf{X} = \mathbf{X}^{(0)}$
3: Repeat:
4: Sparse Approximation: $\mathbf{X}$ =Batch-FISTA$(\mathbf{Y}, \mathbf{D}, \mathbf{X}, \lambda_s)$
5: Dictionary Update: set $\mathbf{E} = \mathbf{Y} - \mathbf{D}\mathbf{X}$
6: for $a = 1, \ldots, A$ do
7:     for $i = 1, \ldots, K$ do
8:         $\mathbf{E}_i = \mathbf{E} + \mathbf{d}_i\mathbf{x}_{[i]}^T$
9:         $\mathbf{x}_{[i]} = \mathcal{S}_\lambda(\mathbf{E}_i^T\mathbf{d}_i)$
10:        $\mathbf{d}_i = \mathbf{E}_i\mathbf{x}_{[i]}$
11:        $\mathbf{d}_i = \mathbf{d}_i/\|\mathbf{d}_i\|_2$
12:        $\mathbf{E} = \mathbf{E}_i - \mathbf{d}_i\mathbf{x}_{[i]}^T$
13:    end for
14: end for
15: Until convergence
```

in learning a good (i.e., sparsifying) dictionary, or extracting a set of good features. We consider an AR(1) signal as in [32], and generate the training signals by chopping this signal into a number of blocks. We compare the performance of our proposed algorithms to those of K-SVD and AK-SVD[2,3]. As it was said in Section III-A, for PAU-DL: $A = 3$, $B = 1$, and for AK-SVD: $A = 1$, $B = 3$.

Our simulations were performed in MATLAB R2010b environment on a system with 3.8 GHz CPU and 8 GB RAM, under Microsoft Windows 7 operating system. As a rough measure of complexity, we will mention the run times of the algorithms.

A. Synthetic Data

We generated a dictionary by normalizing a random matrix of size 20×50, with zero-mean and unit-variance independent and identically distributed (i.i.d.) Gaussian entries. A collection of 2000 training signals $\{\mathbf{y}_i\}_{i=1}^{2000}$ were produced, each as a linear combination of s different columns of the dictionary, with zero-mean and unit-variance i.i.d. Gaussian coefficients in uniformly random and independent positions. We varied s from 3 to 6. We then added white Gaussian noise with Signal to Noise Ratio (SNR) levels of 10, 20, 30, and 100 dB. The exact value of s was given to PAU-DL, K-SVD, AK-SVD, and APrU-DL (with OMP). For OS-DL we used a fixed value of $\lambda = 0.3$. For APrU-DL (with FISTA) we used $\lambda = 0.3$ and $\lambda_s = 0.6$. We applied all algorithms onto these noisy training signals, and compared the resulting recovered dictionaries to the generating one as follows. Assume that $\mathbf{d}_i$ is a generating atom and $\bar{\mathbf{d}}_i$ is the atom in the recovered dictionary that best matches $\mathbf{d}_i$ among the others. We say that the recovery is successful if $|\mathbf{d}_i^T\bar{\mathbf{d}}_i|$ is above 0.99 [27]. The percentage of the correct recovery

[2]As pointed out in the previous works, e.g., [32], the performances of MOD and K-SVD are very similar in these experiments. So, we omitted MOD from the simulations.

[3]For K-SVD, AK-SVD and OMP we have used K-SVD-Box v10 and OMP-Box v10 available at http://www.cs.technion.ac.il/~ronrubin/software.html

was used as the measure of successfully reconstructing the generating dictionary. We performed 100 alternations between sparse approximation and dictionary update stages for all algorithms. The initial dictionary was made by randomly choosing different columns of the training set followed by a normalization. We repeated each experiment (corresponding to a certain value of s and a certain noise level) 50 times and reported the averaged results.

The average percentage of successfully recovering the underlying atoms is shown in Table I. The average execution times of the algorithms for this experiment is shown in Fig. 2. To see the convergence behaviour of K-SVD, AK-SVD, PAU-DL, and APrU-DL, the improvement of recovery ratio along the alternation number is shown in Fig. 3. As we saw in our simulations, the results of AK-SVD were very close to those of K-SVD (as can be seen from Fig. 3). So, AK-SVD has been omitted from Table I. Also, the average execution time of AK-SVD is nearly the same as PAU-DL. The convergence behaviour of OS-DL and APrU-DL with OMP are shown in Fig. 4, where the results of APrU-DL with FISTA are also shown for comparison. With these results in mind, we deduce the following observations:

- PAU-DL has a better successful recovery results compared to both AK-SVD and K-SVD in average. This is especially observable at $s = 5$ and $s = 6$. The average execution time of PAU-DL is also much smaller than that of K-SVD.
- APrU-DL has the best results in average. Also, the results of APrU-DL with FISTA are better than those with OMP. However, the average runtime of FISTA is higher than that of OMP[4].
- OS-DL outperforms PAU-DL and K-SVD, both in convergence rate and the final success rate (this is especially observable at low SNRs and $s = 5$ and $s = 6$). This shows the promising performance of one-stage dictionary learning.
- The convergence rate of PAU-DL is better than K-SVD and AK-SVD, while that of APrU-DL is the best. The convergence behaviour of APrU-DL and OS-DL is approximately the same for different values of s, while those of the other three algorithms deteriorate by increasing s.

B. AR(1) signal

In this experiment, we consider an AR(1) signal (according to [32]), which is generated as $v(k) = 0.95v(k-1) + e(k)$, where $e(k)$ is a zero-mean and unit-variance Gaussian noise. A collection of $L = 2000$ training signals were made by chopping this signal into vectors of length $n = 20$. Number of atoms was set to $m = 40$. As in [32], we computed SNR as SNR $= 10 \log \|\mathbf{Y}\|_F^2 / \|\mathbf{Y} - \mathbf{DX}\|_F^2$. For the sparse approximation stage of APrU-DL we have used OMP to have an exact control on the cardinality of the representations. A number of $s = 5$ atoms were used to approximate each

[4]Note that the average execution times for all algorithms have been calculated for 100 alternations. As can be seen from Fig. 3, APrU-DL has converged in about 20 alternations. So, considering the convergence times, APrU-DL is fast enough compared to K-SVD.

TABLE I
PERCENTAGE OF SUCCESSFUL RECOVERY.

SNR (dB)	Algorithms	$s=3$	$s=4$	$s=5$	$s=6$
10	K-SVD	88.06	87.20	17.80	0
	PAU-DL	88.80	88.67	45.60	0
	OS-DL	92.27	91.47	90.80	74.67
	APrU-DL	94.13	96	90.67	77.87
20	K-SVD	93.61	94.13	88.80	9.34
	PAU-DL	94.13	94.20	92.20	58.13
	OS-DL	93.60	92.53	93.87	91.83
	APrU-DL	95.47	96.53	96.93	96.67
30	K-SVD	94.33	95.87	90.73	18.24
	PAU-DL	94.60	96.13	95.60	82.80
	OS-DL	93.87	93.20	91.84	91.86
	APrU-DL	96.53	96.80	97.87	97.07
100	K-SVD	95.20	94.70	94.80	17.87
	PAU-DL	95.40	94.73	95.60	77.47
	OS-DL	93.20	94.40	95.07	93.20
	APrU-DL	95.33	96.93	97.07	97.47

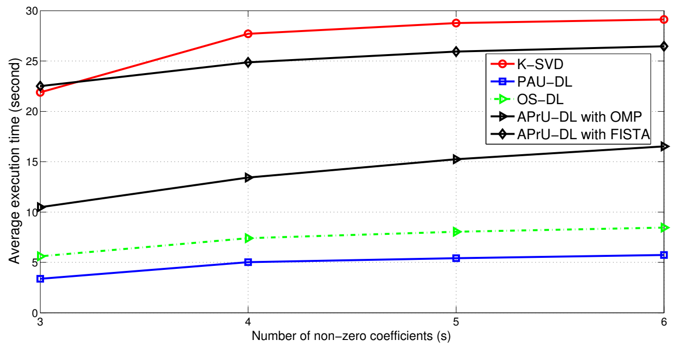

Fig. 2. Average execution times (in second) versus number of nonzero coefficients.

training vector in the sparse approximation stage of PAU-DL, AK-SVD, K-SVD, and APrU-DL. In order to compute SNR in OS-DL, after each update of the dictionary the sparse approximations of the signals have been computed using OMP. For both APrU-DL and OS-DL a value of $\lambda = 0.05$ found to yield promising results. For all algorithms 100 alternations were done.

SNR versus alternation number (averaged over 50 trials) is plotted in Fig. 5. Again, the result of AK-SVD was very similar to that of K-SVD, and thus has been omitted from the figure. This figure shows that the SNR is improved during the learning process for all algorithms. The final values of SNR as well as the average execution times of the algorithms are reported in Table II. Based on these results, we deduce the following observations:

- PAU-DL has reached a higher SNR value compared to K-SVD, while its average execution time is much less than that of K-SVD.
- OS-DL outperforms PAU-DL. This observation again indicates the promising performance of one-stage dictionary learning algorithms.
- APrU-DL has the best results, in the sense of the final value of SNR and the rate of convergence.

As a conclusion, taking into account the performance and the execution time, PAU-DL seems to be the best compromise.

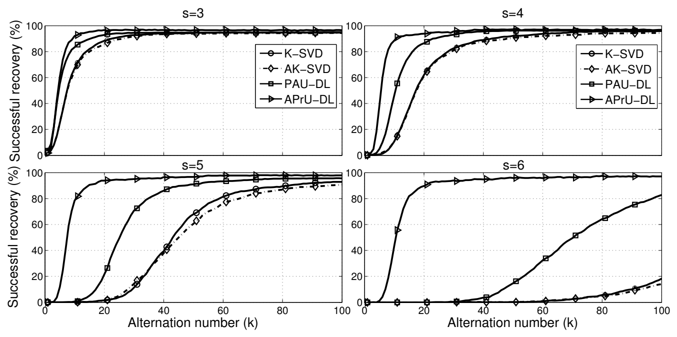

Fig. 3. Convergence behaviour of K-SVD, AK-SVD, PAU-DL, and APrU-DL (with FISTA) in reconstruction of a known dictionary. Each figure corresponds to a certain amount of sparsity (number of nonzero coefficients). Noise level is SNR = 30 dB.

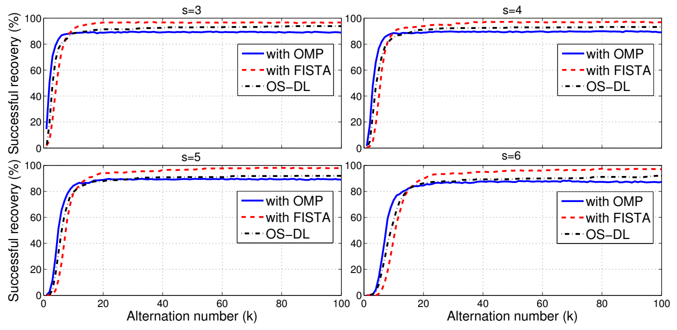

Fig. 4. Convergence behaviour of OS-DL and APrU-DL with (Batch) OMP and (Batch) FISTA as the sparse approximation stage. The setup is the same as in Fig. 3.

V. Conclusion

In this paper, we addressed the dictionary update stage in a general dictionary learning problem. We especially considered the atom-by-atom dictionary update procedure. We introduced the idea of parallel atom-updating and based on this, we derived PAU-DL, an efficient alternative for the well-known K-SVD algorithm which provides atom sequential updating, i.e., fully updating of one atom before moving to the next one. Conversely, the main idea of parallel atom-updating is to update the atoms in parallel. In this way, before moving to update each atom, previous atoms have been partially updated. So, we have more reliable updates of the previous atoms to be used for updating the next atom. We then proposed a novel dictionary learning algorithm which we called OS-DL. This algorithm sequentially updates each atom along with the *whole entries* of its corresponding row vector in the coefficient matrix. OS-DL differs from the existing dictionary learning algorithms in the sense that it does not perform the first stage, i.e., the sparse approximation stage. This is because, here, we constrain the sparsity criterion on the *rows* of the coefficient matrix, not on its columns. To have a better control on the sparsity level of the representations, we then proposed the two-stage counterpart of OS-DL, which is obtained by adding the sparse approximation stage to OS-DL. In this way, as it was seen in Section IV, the convergence rate of OS-DL is increased. Our simulations on recovery of a known dictionary, as well as designing a sparsifying dictionary for an AR(1) signal indicate that our algorithms outperform K-SVD.

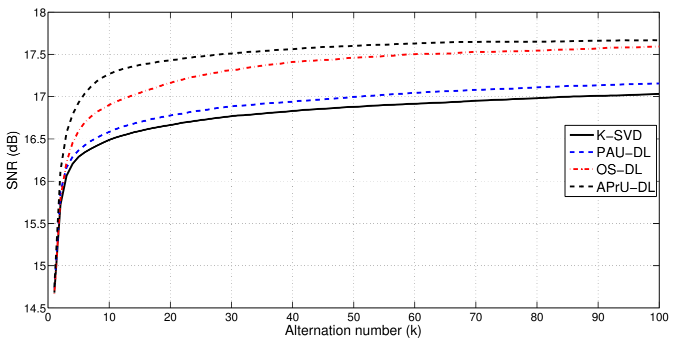

Fig. 5. SNR in dB is plotted versus the alternation number during the learning process.

TABLE II
Final SNR in dB and average execution times (AET) in seconds, for various algorithms.

Algorithm	K-SVD	PAU-DL	OS-DL	APrU-DL
SNR(dB)	17.03	17.16	17.59	17.67
AET(s)	14.57	2.53	12.93	21

References

[1] M. Elad, *Sparse and Redundant Representations*, Springer, 2010.
[2] S. Mallat and Z. Zhang, "Matching pursuits with time-frequency dictionaries," *IEEE Trans. on Signal Proc.*, vol. 41, no. 12, pp. 3397–3415, 1993.
[3] S. S. Chen, D. D. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Rev.*, vol. 43, pp. 129–159, 2001.
[4] D. L. Donoho, M. Elad, and V. Temlyakov, "Stable recovery of sparse overcomplete representations in the presence of noise," *IEEE Trans. Info. Theory*, vol. 52, no. 1, pp. 6–18, 2006.
[5] M. Babaie-Zadeh and C. Jutten, "On the stable recovery of the sparsest overcomplete representations in presence of noise," *IEEE Transactions on Signal Processing*, vol. 58, no. 10, pp. 5396–5400, 2010.
[6] J. A. Tropp and S. J. Wright, "Computational methods for sparse solution of linear inverse problems," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 948–958, 2010.
[7] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," in *In Proc. Asilomar Conf. Signal Syst. Comput.*, 1993.
[8] D. L. Donoho, Y. Tsaig, I. Drori, and J. L. Starck, "Sparse solution of underdetermined systems of linear equations by stagewise orthogonal matching pursuit," *IEEE Trans. on Information Theory*, vol. 58, no. 2, pp. 1094–1121, 2012.
[9] D. Needell and J. A. Tropp, "Cosamp: Iterative signal recovery from in-complete and inaccurate samples," *Appl. Comput. Harmon. Anal.*, vol. 26, no. 3, pp. 301–321, 2009.
[10] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *IEEE Trans. on Information Theory*, vol. 55, no. 5, pp. 2230–2249, 2009.
[11] I. Daubechies, M. Defrise, and C. De-Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Comm. Pure Appl. Math.*, vol. 57, no. 11, pp. 1413–1457, 2004.
[12] M. Elad, B. Matalon, J. Shtok, and M. Zibulevsky, "A wide-angle view at iterated shrinkage algorithms," in *in Proc. SPIE (Wavelet XII)*, 2007, pp. 26–29.
[13] S. Becker, J. Bobin, and E. J. Candès, "NESTA: a fast and accurate first-order method for sparse recovery," *SIAM Journal on Imaging Sciences*, vol. 4, no. 1, pp. 1–39, 2011.
[14] T. Blumensath, "Accelerated iterative hard threshoding," *Signal Processing*, vol. 92, no. 3, pp. 752–756, 2012.
[15] R. Chartrand and W. Yin, "Iteratively reweighted algorithms for compressive sensing," in *IEEE ICASSP*, 2008.
[16] H. Mohimani, M. Babaie-Zadeh, and Ch. Jutten, "A fast approach for overcomplete sparse decomposition based on smoothed ℓ^0 norm," *IEEE Trans. on Signal Processing*, vol. 57, pp. 289–301, 2009.

[17] S. J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, "An interior point method for large-scale ℓ_1-regularized least squares," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 606–617, 2007.

[18] M. Elad, M. A. T. Figueiredo, and Y. Ma, "On the role of sparse and redundant representations in image processing," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 972–982, 2010.

[19] P. Bofill and M. Zibulevsky, "Underdetermined blind source separation using sparse representations," *Signal Processing*, vol. 81, pp. 2353–2362, 2001.

[20] D. L. Donoho, "Compressed sensing," *IEEE Trans. on Information Theory*, vol. 52, no. 4, pp. 1289–1306, April 2006.

[21] R. G. Baraniuk, "Compressive sensing," *IEEE Signal Proc. Magazine*, vol. 24, no. 4, pp. 118–121, 2007.

[22] E. J. Candès and T. Tao, "Decoding by linear programming," *IEEE Transactions on Information Theory*, vol. 51, no. 12, pp. 4203–4215, 2005.

[23] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. Royal. Statist. Soc B.*, vol. 58, no. 1, pp. 267–288, 1996.

[24] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *IEEE Trans. on Image Processing*, vol. 15, no. 12, pp. 3736 – 3745, 2006.

[25] O. Bryt and M. Elad, "Compression of facial images using the K-SVD algorithm," *Journal of Visual Communication and Image Representation*, vol. 19, no. 4, pp. 270–283, 2008.

[26] J. Mairal, F. Bach, and J. Ponce, "Task-driven dictionary learning," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 34, no. 4, pp. 791–804, 2012.

[27] 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.

[28] A. Gersho and R. M. Gray, *Vector Quantization and Signal Compression*, Springer, 1992.

[29] K. Engan, S. O. Aase, and J. Hakon Husoy, "Method of optimal directions for frame design," in *Proceedings of IEEE ICASSP*, 1999, vol. 5.

[30] M. Yaghoobi, T. Blumensath, and M. E. Davies, "Dictionary learning for sparse approximations with the majorization method," *IEEE Trans. on Signal Processing*, vol. 57, no. 6, pp. 2178 – 2191, 2009.

[31] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *Journal of Machine Learning Research*, vol. 11, pp. 19–60, 2010.

[32] K. Skretting and K. Engan, "Recursive least squares dictionary learning algorithm," *IEEE Trans. on Signal Processing*, vol. 58, pp. 2121 – 2130, 2010.

[33] K. Labusch, E. Barth, and T. Martinetz, "Robust and fast learning of sparse codes with stochastic gradient descent," *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 5, pp. 1048–1060, 2011.

[34] J. MacQueen, "Some methods for classification and analysis of multivariate observations," in *Proc. 5th Berkeley Symp. Math. Statist. Probabil.*, 1967, vol. I, pp. 281–296.

[35] R. Rubinstein, M. Zibulevsky, and M. Elad, "Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit," Tech. Rep., Technion University, 2008.

[36] C. D. Sigg, T. Dikk, and J. M. Buhmann, "Learning dictionaries with bounded self-coherence," *IEEE Signal Processing Letters*, vol. 19, no. 12, pp. 861–864, 2012.

[37] W. Dai, T. Xu, and W. Wang, "Simultaneous codeword optimization (SimCO) for dictionary update and learning," *IEEE Trans. on Signal Proc.*, vol. 60, no. 12, pp. 6340–6353, 2012.

[38] M. Elad, "Why simple shrinkage is still relevant for redundant representations?," *IEEE Trans. on Information Theory,*, vol. 52, no. 12, pp. 5559–5569, 2006.

[39] J. Friedman, T. Hastie, H. Höfling, and R. Tibshirani, "Pathwise coordinate optimization," *Annals of Applied Statistics*, vol. 1, no. 2, pp. 302–332, 2007.

[40] A. Beck and M. Teboulle, "Fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM Journal on Imaging Sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[41] M. A. T. Figueiredo, R. D. Nowak, and S. J. Wright, "Gradient projection for sparse reconstruction: Application to compressed sensing and other inverse problems," *IEEE Journal of Selected Topics in Signal Processing*, vol. 1, no. 4, pp. 586–597, 2007.

Mostafa Sadeghi received the B.S. degree in electrical engineering from Ferdowsi University of Mashhad, Mashhad, Iran in 2010, and the M.S degree in electrical engineering from Sharif University of Technology, Tehran, Iran, in 2012. He is now working toward his Ph.D. degree in the electrical engineering department, Sharif University of Technology.

His main research areas are Sparse Signal Processing, Dictionary Learning for Sparse Representation, Statistical Signal Processing, and Machine Learning for Signal Processing.

Massoud Babaie-Zadeh (M04-SM09) received the B.S. degree in electrical engineering from Isfahan University of Technology, Isfahan, Iran in 1994, and the M.S degree in electrical engineering from Sharif University of Technology, Tehran, Iran, in 1996, and the Ph.D. degree in Signal Processing from Institute National Polytechnique of Grenoble (INPG), Grenoble, France, in 2002.

Since 2003, he has been a faculty member of the Electrical Engineering Department of Sharif University of Technology, Tehran, IRAN, firstly as an assistant professor and since 2008 as an associate professor. His main research areas are Blind Source Separation (BSS) and Independent Component Analysis (ICA), Sparse Signal Processing, and Statistical Signal Processing.

Dr. Babaie-Zadeh received the best Ph.D. thesis award of INPG for his Ph.D. dissertation.

Christian Jutten (AM92-M03-SM06-F08) received Ph.D. and Doctor es Sciences degrees in signal processing from Grenoble Institute of Technology (GIT), France, in 1981 and 1987, respectively. From 1982, he was an Associate Professor at GIT), before being Full Professor at University Joseph Fourier of Grenoble, in 1989. For 30 years, his research interests have been machine learning and source separation, including theory (separability, source separation in nonlinear mixtures, sparsity, multimodality) and applications (brain and hyperspectral imaging, chemical sensor array, speech). He is author or coauthor of more than 75 papers in international journals, 4 books, 24 keynote plenary talks, and 170 communications in international conferences.

He has been visiting professor at Swiss Federal Polytechnic Institute (Lausanne, Switzerland, 1989), at Riken labs (Japan, 1996) and at Campinas University (Brazil, 2010). He was director or deputy director of his lab from 1993 to 2010, especially head of the signal processing department (120 people) and deputy director of GIPSA-lab (300 people) from 2007 to 2010). He was a scientific advisor for signal and images processing at the French Ministry of Research (19961998) and for the French National Research Center (20032006). Since May 2012, he is deputy director at the Institute for Information Sciences at French National Center of Research (CNRS) in charge of signal and image processing.

Christian Jutten was organization or program chairs of many international conferences, especially of the 1st International Conference on Blind Signal Separation and Independent Component Analysis in 1999. He has been a member of a few IEEE Technical Committees, and currently in SP Theory and Methods of the IEEE Signal Processing society. He received best paper awards of EURASIP (1992) and of IEEE GRSS (2012), and Medal Blondel (1997) from the French Electrical Engineering society for his contributions in source separation and independent component analysis. He is IEEE fellow (2008) and EURASIP fellow (2013). He is a Senior Member of the Institut Universitaire de France since 2008, with renewal in 2013. He is the recipient of a 2012 ERC Advanced Grant for a project on challenges in extraction and separation of sources (CHESS).