LEARNING OVERCOMPLETE DICTIONARIES BASED ON PARALLEL ATOM-UPDATING

Mostafa Sadeghi[1], *Massoud Babaie-Zadeh*[1], *Christian Jutten*[2*]

[1]Electrical engineering department, Sharif University of Technology, Tehran, Iran.
m.saadeghii@gmail.com, mbzadeh@yahoo.com
[2]GIPSA-Lab, University of Grenoble, and the Institut Universitaire de France, France.
christian.jutten@gipsa-lab.grenoble-inp.fr

ABSTRACT

In this paper we propose a fast and efficient algorithm for learning overcomplete dictionaries. The proposed algorithm is indeed an alternative to the well-known K-Singular Value Decomposition (K-SVD) algorithm. The main drawback of K-SVD is its high computational load especially in high-dimensional problems. This is due to the fact that in the dictionary update stage of this algorithm an SVD is performed to update each column of the dictionary. Our proposed algorithm avoids performing SVD and instead uses a special form of alternating minimization. In this way, as our simulations on both synthetic and real data show, our algorithm outperforms K-SVD in both computational load and the quality of the results.

Index Terms— Sparse approximation, compressive sensing, dictionary learning, alternative minimization

1. INTRODUCTION

1.1. Sparse signal approximation

Sparse approximation of signals has received a lot of attention during the last decade [1]. This is due to its capability in various applications such as Compressive Sensing (CS) [2], image processing tasks (e.g. denoising, compression, inpainting, zooming) [3], and linear regression and variable selection [4]. The sparse approximation problem consists in approximating a given signal as a linear combination of as few as possible basis functions. Each basis function is called an *atom* and the collection of them is called *dictionary* [5]. This dictionary is overcomplete, i.e. the number of atoms is (much) more than the dimension of each atom. Specifically, let $\mathbf{y} \in \mathbb{R}^n$ be the signal whose sparsest approximation is going to be found in $\mathbf{D} \in \mathbb{R}^{n \times K}$, with $K > n$. This amounts to solve the following problem:

$$P_0: \quad \min_{\mathbf{x}} \|\mathbf{x}\|_0 \quad \text{subject to} \quad \|\mathbf{y} - \mathbf{Dx}\|_2 \leq \epsilon, \tag{1}$$

where $\|.\|_2$ is the ℓ_2 norm, $\|.\|_0$ stands for the so-called ℓ_0 (pseudo) norm, which counts the number of non-zero elements, and ϵ is a small positive constant. The above problem needs a combinatorial search and is generally NP-hard [6]. So, alternative methods are used to solve it [1, 7]. One of the most successful ideas is to replace the non-convex sparsity measure $\|.\|_0$ with its best convex approximation based on ℓ_1 norm [8]. This leads to the following convex problem:

$$P_1: \quad \min_{\mathbf{x}} \|\mathbf{x}\|_1 \quad \text{subject to} \quad \|\mathbf{y} - \mathbf{Dx}\|_2 \leq \epsilon. \tag{2}$$

Many algorithms have been introduced to solve the problem of finding the sparsest approximation of a signal in a given overcomplete dictionary (for a good review see [9]). These algorithms can be classified into two general categories, *greedy methods* such as Orthogonal Matching Pursuit (OMP) [10] and Compressive Sampling Matching Pursuit (CoSaMP) [11], and *relaxation methods*, which replace the combinatorial P_0 problem with a tractable one, e.g. P_1 problem. Iterative Shrinkage-Thresholding (IST) [12], Iteratively Re-weighted Least Squares (IRLS) [13], and Smoothed ℓ_0 (SL0) [7] 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. Theses algorithms benefit from high speed, but their accuracy is usually less than that of the second category.

1.2. Learning overcomplete dictionaries

For a given class of signals, e.g. class of natural facial images, the dictionary should have the capability of sparsely approximating the signals. In some applications there is a pre-defined and universal dictionary that is 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

*This work has been partially funded by the Iran National Science Foundation (INSF) under contract number 91004600, and by the European project ERC-2012-AdG320684-CHESS.

because of their simplicity and in some cases their fast computations. However, *learning* the atoms from some 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 [14], classification tasks [15], and so on.

In this paper we are going to propose an efficient dictionary learning algorithm, which is indeed a good alternative to the well-known K-SVD algorithm [16]. As will be seen in the experimental results, our proposed algorithm outperforms K-SVD in both execution time and quality of the results.

The paper is organized as follows. In Section 2 we briefly review the general dictionary learning problem together with K-SVD algorithm. Then in Section 3 we detail our proposed algorithm. Section 4 presents some experimental results. Finally, Section 5 concludes the paper.

2. DICTIONARY LEARNING PROBLEM

Consider a set of L training signals in $\mathbb{R}^n$ as $\{\mathbf{y}_i\}_{i=1}^L$. 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 $\|.\|_F$ is the Frobenius norm and $\mathcal{D}$ 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-norm. Since we require that each signal has a sparse approximation, $\mathcal{X}$ is the set of all matrices $\mathbf{X}$ with sparse columns.

The general approach to solve (3) is to use alternating minimization in which the following two stages are repeated several times:

1. **Sparse approximation:** with a fixed $\mathbf{D}$, solve (3) for $\mathbf{X}$,
2. **Dictionary update:** with a fixed $\mathbf{X}$, update the dictionary to reduce the approximation error.

With a fixed $\mathbf{D}$, the minimization of (3) over $\mathbf{X}$ is equivalent to sparsely approximating the training signals over $\mathbf{D}$. Among the various sparse coding algorithms, OMP (or its variants) is very appealing. This is due to two reasons. The first reason is the high speed of OMP. The second one is its capability to be efficiently implemented in batch mode. In some applications, such as dictionary learning, in which we need to find sparse approximations of a batch of signals in the same dictionary, we can factor some common operations of OMP. This fact along with the use of Cholesky factorization results in the significant acceleration of OMP which leads to Batch-OMP algorithm [17].

Up to our best knowledge, various dictionary learning algorithms differ only in the way of updating the dictionary. Some algorithms such as K-SVD are based on updating atoms one-by-one whereas some others such as Method of Optimal Directions (MOD) [18] update the whole set of atoms at once.

2.1. K-SVD

In the dictionary update stage of K-SVD, only one atom is updated at a time. Moreover, while updating each atom, the non-zero entries in the associated row vector of $\mathbf{X}$ is also updated. In this way, only those signals participate in updating one atom that have used it. This prevents each row vector in $\mathbf{X}$ to be filled and thus violating the sparsity constraint of the coefficient matrix. In what follows, we define $\mathbf{x}(\omega)$ as a vector containing those entries of $\mathbf{x}$ that are indexed by ω, and $\mathbf{E}(:,\omega)$ as a matrix containing those columns of $\mathbf{E}$ that are indexed by ω. Assume that we want to update the ith atom, $\mathbf{d}_i$, along with the non-zero 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 $\mathbf{x}_{[i]}^T$, and $|\omega_i|$ as the cardinality of ω_i. Then the problem of updating $\mathbf{d}_i$ along with $\mathbf{x}_{[i]}^T(\omega_i)$ amounts to solve the following minimization problem:

$$\min_{\mathbf{d},\mathbf{x}_r} \|\mathbf{E}_i^r - \mathbf{d}\mathbf{x}_r^T\|_F^2 \quad \text{subject to} \quad \|\mathbf{d}\|_2 = 1, \tag{4}$$

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 $|\omega_i|$ corresponding to the non-zero entries of $\mathbf{x}_{[i]}^T$. The above problem actually finds the closest rank-1 approximation to $\mathbf{E}_i^r$ and can be easily solved via SVD. For more details refer to [16].

3. PARALLEL ATOM-UPDATING DICTIONARY LEARNING

In this section we introduce our proposed algorithm. This algorithm like K-SVD is based on atom-by-atom updating. The main drawback of K-SVD is its computational load especially in high dimensions. This is due to performing SVD for atom updating. An alternative way of solving (4) is to use the idea of alternating minimization [17]. In other words, (4) 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) [17]. Although performing more alternations gives a better approximation, the average performance will not exceed the performance of the exact solution, i.e. via SVD.

We describe a different way of performing alternating minimization to update the atoms and their associated row vectors in the coefficient matrix. To this aim, consider the overall error matrix,

$$\mathbf{E} = \mathbf{Y} - (\mathbf{A}_1 + \mathbf{A}_2 + \ldots + \mathbf{A}_K),\ \forall i:\ \mathbf{A}_i = \mathbf{d}_i\mathbf{x}_{[i]}^T. \tag{5}$$

Algorithm 1 PAU-DL algorithm

```
1: Task: Learning an overcomplete dictionary to sparsely approximate $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^{L}$.
2: Initialization: Set $k = 0$, $\mathbf{D}^{(k)} = \mathbf{D}^0$.
3: The main loop: Repeat until convergence:
4: Sparse Approximation: $\mathbf{X}^{(k)} = \text{Batch-OMP}(\mathbf{Y}, \mathbf{D}^{(k)}, \tau)$.
5: Dictionary Update: Set $\mathbf{D} = \mathbf{D}^{(k)}$, $\mathbf{X} = \mathbf{X}^{(k)}$, $\mathbf{E} = \mathbf{Y} - \mathbf{D}\mathbf{X}$, and repeat the following loop $A$ times ($A = 3$ works well):
6: for $i = 1, \dots, K$ do
7:     $\mathbf{E}_i = \mathbf{E} + \mathbf{d}_i \mathbf{x}_{[i]}^T$
8:     $\mathbf{E}_i^r = \mathbf{E}_i(:, \omega_i)$ where $\omega_i = \left\{ j : \mathbf{x}_{[i]}^T(j) \neq 0 \right\}$
9:     $\mathbf{x}_r = \mathbf{x}_{[i]}(\omega_i)$
10:    $\mathbf{d}_i = \mathbf{E}_i^r \mathbf{x}_r$
11:    $\mathbf{d}_i = \mathbf{d}_i / \|\mathbf{d}_i\|_2$
12:    $\mathbf{x}_{[i]}^T(\omega_i) = \mathbf{d}_i^T \mathbf{E}_i^r$
13:    $\mathbf{E} = \mathbf{E}_i - \mathbf{d}_i \mathbf{x}_{[i]}^T$
14: end for
15: Set $k = k + 1$, $\mathbf{D}^{(k)} = \mathbf{D}$, and go back to the sparse approximation stage.
```

In K-SVD (or AK-SVD), to update (the non-zero columns of) for example $\mathbf{A}_i$, the updated versions of $\mathbf{A}_1, \dots, \mathbf{A}_{i-1}$ are used to compute $\mathbf{E}_i$; while $\mathbf{A}_{i+1}, \dots, \mathbf{A}_K$ 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 $\mathbf{x}_r$, we perform "A" alternations 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 4 suggest, parallel updating of the atoms results in further accelerating the convergence rate and even the quality of the final result.

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 $\mathbf{E} = \mathbf{Y} - \mathbf{D}\mathbf{X}$ 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 $\mathbf{E} = \mathbf{E}_i - \mathbf{A}_i^{(new)}$. In this way, there is no need to completely re-compute the error matrix for each atom, but instead it is updated in a ranked-1 manner.

Algorithm 1 gives a description of the proposed algorithm, which we have called it Parallel Atom-Updating Dictionary Learning (PAU-DL) algorithm. 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 (ϵ in (1)) or the maximum allowed number of atoms in the sparse approximation of each training signal.

4. SIMULATIONS

We evaluate the efficiency of our proposed algorithm and K-SVD[1] with three sets of experiments. In the first set we aim to evaluate the capability of the two algorithms in recovery of a known dictionary. The second experiment is on an autoregressive (AR) signal, where there is no underlying dictionary, and we just evaluate the capability of the algorithms in learning a good (i.e. sparsifying) dictionary form the training signals. To further evaluate the advantage of PAU-DL over K-SVD, we consider as the third experiment the problem of image denoising over a learned overcomplete dictionary as in [14].

It is worthwhile mentioning that, as previously stated, although AK-SVD has much smaller computational load compared to original K-SVD algorithm, but as we saw in our simulations, its quality of results is inferior to that of K-SVD. So, we have reported here the results of simulating K-SVD; not AK-SVD. Also, the computational complexities of AK-SVD and PAU-DL are nearly the same.

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.

4.1. Reconstruction of a known dictionary

We generated the underlying 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 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. We applied the two algorithms onto this noisy training signals, and compared the resulting recovered dictionaries to the generating dictionary 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 [16]. The percentage of the correct recovery was used as the measure of successfully reconstructing the generative dictionary. We performed 100 alternations between sparse approximation and dictionary update stages for both algorithms. The initial dictionary was made by randomly choosing different columns of the training signals followed by a normalization.

The average percentage of successfully recovery of the underlying atoms (APSRA) is shown in Table 1. To see the

[1]For K-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`

Table 1. Percentage of successful recovery.

SNR (dB)	Algorithms	$s = 3$	$s = 4$	$s = 5$	$s = 6$
10	K-SVD	87.40	78.20	4.47	0
	PAU-DL	87.43	79.37	18.47	0
20	K-SVD	93.87	94.4	85.47	0
	PAU-DL	93.88	95.2	91.13	13.6
30	K-SVD	94.40	92.87	88.4	2
	PAU-DL	94.47	94.20	93	43.13
100	K-SVD	94	94.2	90.27	1.8
	PAU-DL	94	95.13	92.93	40.93

convergence behaviour of the two algorithms, the improvement of APSRA along the alternation number for SNR $=$ 20 dB and $s = 4, 5$ is shown in Fig. 1. The average execution times of K-SVD and PAU-DL were 27.68 and 5.41 seconds, respectively. With these results in mind, we deduce the following observations:

- PAU-DL has a better APSRA compared to K-SVD in average. This is especially evident at $s = 5$ and $s = 6$.
- The convergence rate of PAU-DL is faster than K-SVD.

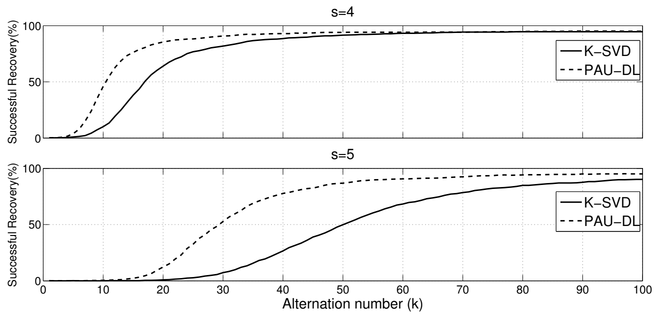

Fig. 1. Convergence behaviour of PAU-DL and K-SVD in recovery of a known dictionary (SNR $=$ 20 dB).

4.2. AR(1) signal

In this experiment, we consider an AR(1) signal (according to [19]) that is generated as $v(k) = 0.95v(k-1) + e(k)$, where $e(k)$ is Gaussian noise with zero mean and unit variance. 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 $m = 40$, and $s = 5$ atoms were used to approximate each training vector. For both algorithms 100 alternations were done. As in [19], we computed SNR as $\text{SNR} = 10 \log \|\mathbf{Y}\|_F^2 / \|\mathbf{Y} - \mathbf{DX}\|_F^2$.

SNR versus alternation number is plotted in Fig. 2. From this figure we see that PAU-DL has reached a higher SNR value compared to K-SVD. The average execution times of K-SVD and PAU-DL were 15.88 and 2.79 seconds, respectively.

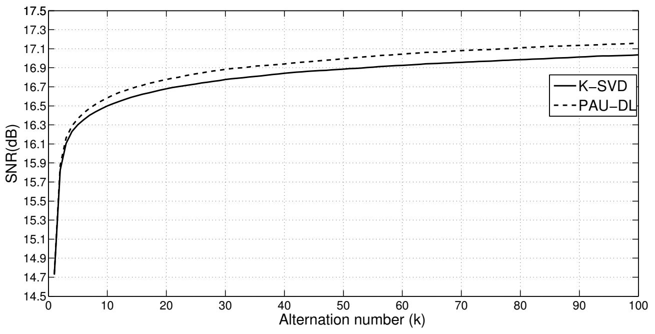

Fig. 2. SNR in dB is plotted versus the alternation number during the learning process.

4.3. Image denoising over learned dictionaries

We evaluate here the efficiency of PAU-DL relative to K-SVD in an image denoising experiment. We aim to find an estimate of a clean image contaminated with an additive white Gaussian noise with zero mean. The approach of sparse approximation is to firstly divide the noisy image into some small blocks (typically of size 8×8), then learn an overcomplete dictionary using these blocks as the training signals, and finally denoise each block over the learned dictionary [14]. As in [14], we used Peak Signal to Noise Ratio (PSNR), as the criterion for comparison of the denoising quality. We considered four test images, all of size 256×256. To learn a dictionary for each image, we randomly selected 30000 blocks of it as the training signals. We repeated each experiment (corresponding to a certain level of noise) 5 times and averaged the results.

The final PSNR values are shown in Table 2, in which σ is the standard deviation of noise. As can be seen, the results are similar for the two algorithms (with PAU-DL slightly better). However, the average execution times of K-SVD and PAU-DL were approximately 90 and 30 seconds, respectively.

As a conclusion, these results together with those in the previous experiments all suggest the advantage of PAU-DL compared to K-SVD.

5. CONCLUSION

We proposed a fast and efficient alternative to K-SVD. Our algorithm is based on updating the atoms of the dictionary in parallel using the idea of alternating minimization. As the simulations results on both synthetic and real data showed, our algorithm outperforms K-SVD in both average execution time and quality of the results.

Table 2. Image denoising PSNR results in dB. In each cell two results are reported. Top: results of K-SVD, and Bottom: results of PAU-DL. The average execution times of K-SVD and PAU-DL were 90 and 30 seconds, respectively.

σ, PSNR	House	Boat	Lena	Barb.	Avg.
2, 42.11	44.48	43.66	44.50	43.96	44.15
	44.50	43.67	44.53	43.98	44.15
5, 34.16	39.44	37.41	38.86	38.13	38.46
	39.46	37.43	38.89	38.18	38.50
10, 28.11	36.08	33.32	35.02	34.10	34.63
	36.14	33.37	35.10	34.15	34.69
15, 24.62	34.44	31.04	32.85	31.83	32.54
	34.51	31.08	32.91	31.86	32.59
20, 22.11	33.27	29.54	30.26	30.17	30.81
	33.31	29.57	30.28	30.19	30.84
25, 20.16	32.19	28.33	30.00	29.02	29.89
	32.21	28.35	30.04	29.04	29.91
50, 14.15	28.07	24.81	26.14	25.33	26.09
	28.08	24.85	26.16	25.34	26.11
100, 8.12	23.68	21.80	22.57	21.90	22.49
	23.69	21.82	22.60	21.91	22.51

6. REFERENCES

[1] M. Elad, *Sparse and Redundant Representations*, Springer, 2010.

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

[3] 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.

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

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

[6] 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.

[7] 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.

[8] S. S. Chen, D. D. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Rev.*, vol. 43, pp. 129–159, 2001.

[9] 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.

[10] 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.

[11] 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.

[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] R. Chartrand and W. Yin, "Iteratively reweighted algorithms for compressive sensing," in *IEEE ICASSP*, 2008.

[14] 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.

[15] 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.

[16] 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.

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

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

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