ANALYSIS SIMCO: A NEW ALGORITHM FOR ANALYSIS DICTIONARY LEARNING

Jing Dong[⋆] *Wenwu Wang*[⋆] *Wei Dai*[†]

[⋆] Centre for Vision, Speech and Signal Processing, University of Surrey, UK
[†] Department of Electrical and Electronic Enginnering, Imperial College London, UK
Email: {j.dong, w.wang}@surrey.ac.uk wei.dai1@imperial.ac.uk

ABSTRACT

We consider the dictionary learning problem for the analysis model based sparse representation. A novel algorithm is proposed by adapting the synthesis model based simultaneous codeword optimisation (SimCO) algorithm to the analysis model. This algorithm assumes that the analysis dictionary contains unit ℓ_2-norm atoms and trains the dictionary by the optimisation on manifolds. This framework allows one to update multiple dictionary atoms in each iteration, leading to a computationally efficient optimisation process. We demonstrate the competitive performance of the proposed algorithm using experiments on both synthetic and real data, as compared with three baseline algorithms, Analysis K-SVD, analysis operator learning (AOL) and learning overcomplete sparsifying transforms (LOST), respectively.

Index Terms— Analysis model, SimCO, analysis dictionary learning.

1. INTRODUCTION

Dictionary design is an important problem in sparse representation. Recent studies have shown that dictionaries learned from a set of training signals have the potential to fit the signals better than the analytical dictionaries [1], [2]. Many dictionary learning algorithms, such as MOD [3], K-SVD [1] and SimCO [4], are established on the synthesis model, where a signal is represented as a linear combination of a few atoms (signal components) from the dictionary. Learning dictionaries with an analysis model where the product of the dictionary and the signal is sparse, however, has received less attention, with only a few activities emerging recently, such as [2], [5], [6], [7], [8]. Analysis K-SVD algorithm [2] was proposed based on the K-SVD algorithm [1]. SVD is applied to update the rows of the dictionary one by one. The algorithm is able to recover a high percentage of the atoms of the ground-truth dictionary used to generate the training signals. However, its computational complexity is quite high. The algorithm reported in [5], named analysis operator learning (AOL), restricts the dictionary on the uniformly normalized tight frame (UNTF) and formulates the dictionary learning problem as an ℓ_1 optimisation problem. It is computationally efficient but the constraints exclude the feasible dictionaries outside UNTF. The algorithm proposed in [6] is based on ℓ_p-norm minimization on the set of full rank dictionaries, which is addressed by a conjugate gradient method on manifolds. The objective function of this algorithm is however complicated with many parameters required to be carefully chosen. In [7], a learning overcomplete sparsifying transform (LOST) algorithm is proposed by extending the work in [8] from complete dictionary to the overcomplete case, where both the full rank and unit norm constraints are considered in the optimisation criterion. As demonstrated in Section 4.1, however, this algorithm is less effective in reaching the cosparsity pre-defined in the optimisation criterion.

Here we focus on the analysis dictionary learning (ADL) problem and propose a new algorithm which can partly address the limitations of the algorithms mentioned above. More specifically, we adapt the synthesis model based SimCO algorithm [4] to the analysis model and develop a new ADL algorithm which is referred to as Analysis SimCO. We assume that the analysis dictionary contains unit ℓ_2-norm rows. The Analysis SimCO algorithm has three important features. First, the optimisation method on manifolds modified from the framework of SimCO is applied to update the dictionary. Second, multiple dictionary atoms can be updated simultaneously which is different from the Analysis K-SVD algorithm where only one atom is allowed to be updated in each iteration. Third, the Analysis SimCO is computationally efficient due to the use of a simple optimisation process.

The remainder of the paper is organized as follows. The SimCO algorithm is introduced in Section 2. Section 3 presents the proposed Analysis SimCO algorithm in detail and discusses its computational complexity. Section 4 demonstrates the performance of the Analysis SimCO algorithm via simulations on synthetic and real data. Finally, Section 5 concludes the paper.

2. THE SIMCO ALGORITHM

SimCO [4] was proposed to train an overcomplete synthesis dictionary containing unit ℓ_2-norm atoms from a set of signals so that the signals can be best approximated by a few atoms of the dictionary.

Let $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_n] \in \mathbb{R}^{m \times n}$ denote the training signals, where each column of $\mathbf{Y}$ is one training signal. In SimCO, the dictionary learning problem is formulated as

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

$$\inf_{\mathbf{D}\in\mathcal{D}} f(\mathbf{D}) = \inf_{\mathbf{D}\in\mathcal{D}} \underbrace{\inf_{\mathbf{A}\in\mathcal{A}} \|\mathbf{Y}-\mathbf{D}\mathbf{A}\|_F^2}_{f(\mathbf{D})} \tag{1}$$

where $\|\cdot\|_F$ is the Frobenius norm. $\mathbf{A}\in\mathbb{R}^{d\times n}$ contains the representation vectors and $\mathbf{D}\in\mathbb{R}^{m\times d}$ is the dictionary we seek. $\mathcal{D}$ represents the set of all feasible matrices that contain unit ℓ_2-norm columns. $\mathcal{A}$ represents the set of all feasible coefficient matrices with a sparsity pattern which contains the indices of all the non-zero elements in $\mathbf{A}$.

To solve the optimisation problem (1), SimCO follows the conventional two-stage optimisation process – sparse coding and dictionary update. The sparse coding stage is to calculate the sparse representations $\mathbf{A}$ of the signals $\mathbf{Y}$ for a given $\mathbf{D}$. Sparse coding algorithms such as Orthogonal Matching Pursuit [9] can be used in this stage. In the dictionary update stage, SimCO uses the optimisation methods on manifolds to update the dictionary $\mathbf{D}$ and $\mathbf{A}$ simultaneously while keeping the sparsity pattern of $\mathbf{A}$ unchanged. This framework is able to update multiple atoms of $\mathbf{D}$ simultaneously in each iteration and guarantee the columns of $\mathbf{D}$ to have unit ℓ_2-norm.

3. THE ANALYSIS SIMCO ALGORITHM

Here we address the problem of dictionary learning for the analysis model. Given a set of training signals $\mathbf{Y}$, ADL aims to learn a dictionary $\mathbf{\Omega}\in\mathbb{R}^{p\times m}$ with which the analysis representations of $\mathbf{Y}$ are sparse. This problem can be cast as

$$\min_{\mathbf{X},\mathbf{\Omega}} \|\mathbf{X}-\mathbf{\Omega}\mathbf{Y}\|_F^2 \quad \text{s.t.} \quad \forall i,\ \|\mathbf{X}_{:,i}\|_0 = p-l \tag{2}$$

where $\mathbf{X}_{:,i}$ is the ith column of $\mathbf{X}$. The ℓ_0 quasi-norm $\|\cdot\|_0$ counts the number of non-zeros of a vector and l is the cosparsity. However, the problem (2) has many trivial solutions, for example, $\mathbf{\Omega}=\mathbf{0}$. In order to exclude such trivial solutions, we assume the rows of $\mathbf{\Omega}$ to have unit ℓ_2-norm. In this case, the ADL problem can be rewritten as

$$\begin{aligned} &\min_{\mathbf{X},\mathbf{\Omega}} \|\mathbf{X}-\mathbf{\Omega}\mathbf{Y}\|_F^2 \\ \text{s.t.}\quad &\forall i,\ \|\mathbf{X}_{:,i}\|_0 = p-l, \\ &\forall j,\ \|\mathbf{\Omega}_{j,:}\|_2 = 1 \end{aligned} \tag{3}$$

where $\mathbf{\Omega}_{j,:}$ denotes the jth row of $\mathbf{\Omega}$.

The problem (3) can be viewed as two separate optimisation problems on $\mathbf{X}$ and $\mathbf{\Omega}$ respectively by keeping one fixed and changing the other. Inspired by the dictionary learning algorithms based on the synthesis model, ADL can also alternate between two stages: analysis sparse coding and dictionary update, initializing with a random and normalized $\mathbf{\Omega}$. The first stage calculates $\mathbf{X}$ for the given dictionary $\mathbf{\Omega}$. In the dictionary update stage, $\mathbf{\Omega}$ is updated assuming known and fixed $\mathbf{X}$. We attempt to develop the Analysis SimCO based on SimCO for the update of $\mathbf{\Omega}$, observing the similarity between (3) and (1). We use the name Analysis SimCO since the atoms are updated simultaneously using optimization on manifolds like SimCO.

Algorithm 1 Analysis SimCO

Input: $\mathbf{Y}$, p, l
Output: $\hat{\mathbf{\Omega}} = \mathbf{\Omega}^{k+1}$
Initialization:
Initialize the iteration counter $k=1$ and the analysis dictionary $\mathbf{\Omega}^k$. Perform the following steps.
Main Iterations:
1. Calculate $\mathbf{X}^k$ according to equation (4): $\mathbf{X}^k = \mathbf{\Omega}^k\mathbf{Y}$
2. Apply hard thresholding operation according to equation (5): $\hat{\mathbf{X}}^k = HT_l(\mathbf{X}^k)$
3. Update the analysis dictionary according to equations (7), (8), and (9): $\mathbf{\Omega}^{k+1} \leftarrow \mathbf{\Omega}^k$
4. Increase the iteration counter: $k = k+1$
5. If the stopping criterion is satisfied, quit the iteration. Otherwise, go back to step 1.

The procedure of our proposed Analysis SimCO algorithm is presented in Algorithm 1. The details are given in the following subsections.

3.1. Analysis Sparse Coding

The goal of analysis sparse coding is to get the sparse representations of the training signals based on a given dictionary. Unlike the corresponding part of the synthesis model, here the exact representations $\mathbf{X}$ can be calculated directly by simply multiplying the signals by the dictionary, that is

$$\mathbf{X} = \mathbf{\Omega}\mathbf{Y} \tag{4}$$

However, the representations obtained by equation (4) may not be sparse since the initial dictionary is an arbitrary one.

A hard thresholding operation is therefore applied to enforce the cosparsity

$$\hat{\mathbf{X}} = HT_l(\mathbf{X}) \tag{5}$$

where $HT_l(\mathbf{X})$ is an operator that sets the smallest l elements (in magnitude) of each column of $\mathbf{X}$ to 0. In doing so, the sparsity constraint can be enforced.

3.2. Dictionary Update

The dictionary update stage aims to find the solution of the optimisation problem

$$\min_{\mathbf{\Omega}} \|\mathbf{X}-\mathbf{\Omega}\mathbf{Y}\|_F^2 \quad \text{s.t.} \quad \forall j, \|\mathbf{\Omega}_{j,:}\|_2 = 1. \tag{6}$$

The Stiefel manifold $\mathcal{U}_{m,1}$ is defined as $\mathcal{U}_{m,1} = \{\mathbf{u}\in\mathbb{R}^m : \mathbf{u}^T\mathbf{u}=1\}$ [4]. Based on this definition, the transpose of each row in $\mathbf{\Omega}$ is one element in $\mathcal{U}_{m,1}$, while in SimCO each column in the dictionary $\mathbf{D}$ is one element in $\mathcal{U}_{m,1}$ [4]. Realizing this similarity, we can solve the optimisation problem (6) by applying a similar line search path on manifolds as in SimCO.

Here we use the first order optimisation procedures of SimCO, i.e. the gradient descent line search method. We explain below the key points of this method including search direction, line search path and step size respectively.

3.2.1. Search Direction

Using the gradient descent method, the search direction $\mathbf{H}$ is the negative gradient of the cost function (6) with respect to $\mathbf{\Omega}$, that is

$$\mathbf{H} = -\frac{\partial \|\mathbf{X} - \mathbf{\Omega}\mathbf{Y}\|_F^2}{\partial \mathbf{\Omega}} = 2\mathbf{X}\mathbf{Y}^T - 2\mathbf{\Omega}\mathbf{Y}\mathbf{Y}^T \tag{7}$$

3.2.2. Line Search Path

Let $\mathbf{h}_j$ be the jth row of $\mathbf{H}$. Define

$$\bar{\mathbf{h}}_j = \mathbf{h}_j - \mathbf{h}_j \mathbf{\Omega}_{j,:}^T \mathbf{\Omega}_{j,:}, \ \forall j \in \{1, 2, ..., p\} \tag{8}$$

so that $\bar{\mathbf{h}}_j \mathbf{\Omega}_{j,:}^T = \mathbf{\Omega}_{j,:} \bar{\mathbf{h}}_j^T = 0$. The line search path for dictionary update, say $\mathbf{\Omega}(t)$, where $t \geq 0$ denotes step size, is given by [4]

$$\begin{cases} \mathbf{\Omega}_{j,:}(t) = \mathbf{\Omega}_{j,:} & \text{if } \|\bar{\mathbf{h}}_j\|_2 = 0, \\ \mathbf{\Omega}_{j,:}(t) = \mathbf{\Omega}_{j,:} \cos(\|\bar{\mathbf{h}}_j\|_2 t) + (\bar{\mathbf{h}}_j / \|\bar{\mathbf{h}}_j\|_2) \sin(\|\bar{\mathbf{h}}_j\|_2 t) & \\ \qquad\qquad \text{if } \|\bar{\mathbf{h}}_j\|_2 \neq 0 \end{cases} \tag{9}$$

Using equation (9) to update the rows of $\mathbf{\Omega}$, the constraint that the dictionary $\mathbf{\Omega}$ only contains unit ℓ_2-norm rows can always be satisfied.

3.2.3. Step Size

We use the method of golden section search, same as that in SimCO [4], to find a proper step size t.

3.3. Computational Complexity

The time complexity of the sparse coding stage is dominated by the calculation of $\mathbf{\Omega}\mathbf{Y}$, at $O(pmn)$. In the dictionary update stage, computing the gradient $\mathbf{H}$ is the dominating part. Computing the product $\mathbf{X}\mathbf{Y}^T$ requires $O(pmn)$ operations. The time complexity of $\mathbf{\Omega}\mathbf{Y}\mathbf{Y}^T$ is $O(pm^2)$ with precomputed $\mathbf{Y}\mathbf{Y}^T$. As a result, the dictionary update stage requires $O(pmn)$ operations with the usual case $n > m$. The total time complexity of each iteration of the Analysis SimCO algorithm thus scales as $O(pmn)$.

In contrast, the computational complexity of the Analysis K-SVD algorithm is dominated by the analysis pursuit algorithm, i.e. Backward-Greedy (BG). For one training sample, the complexity of BG is $O(pm^2)$ and the computational complexity becomes $O(pm^3)$ if the Optimized BG (OBG) is applied instead [2]. Given n training samples, the complexity of the Analysis K-SVD algorithm is $O(pm^2 n)$ using BG or $O(pm^3 n)$ using OBG.

Our proposed algorithm leads to a significant reduction of the computational cost as compared with the Analysis K-SVD, as also confirmed in the next section.

4. SIMULATION RESULTS

We conducted some experiments for both synthetic and real data to demonstrate the performance of the Analysis SimCO algorithm.

4.1. Experiments With Synthetic Data

We randomly generated a dictionary and a set of data with fixed cosparsity l based on the dictionary. The elements of the ground-truth analysis dictionary $\mathbf{\Omega} \in \mathbb{R}^{p \times m}$ were generated from the Gaussian distribution with zero mean and unit variance and the rows of $\mathbf{\Omega}$ were normalized. Then the training signals were generated based on $\mathbf{\Omega}$ and normalized.

In the tests, we fixed the parameters $p = 50,\ m = 25,\ l = 21,\ n = 50000$. We choose Analysis K-SVD [2], AOL [5] and LOST [7] as baseline algorithms. The cases of training with clean and noisy signals were both tested. In the noisy case, the noise level was $\sigma = 0.04$ ($SNR = 25dB$). For the algorithm AOL, we applied its noiseless version (NL)AOL to the noiseless case and its noise-aware version (NA)AOL to the noisy case. The algorithms were initialized with a data-driven dictionary in which each row was orthogonal to a random set of $m - 1$ examples as in [2]. We applied 300 iterations of the Analysis K-SVD using the OBG algorithm, 300 iterations of (NA)AOL, and 1000 iterations of the Analysis SimCO, (NL)AOL and LOST[1]. All algorithms ran 5 times for the noiseless and noisy case respectively. The target dimension of signals in the Analysis K-SVD was set as 4. The parameters of LOST were set empirically as $\alpha = 10^{-4},\ \eta = \lambda = \mu = 50,\ s = 29,\ p = 20$ (Here p is a parameter in the objective function of LOST, not the number of atoms in $\mathbf{\Omega}$). The step size of (NL)AOL was $\eta = 5 \times 10^{-6}$. For the (NA)AOL algorithm, we use the same setup as originally suggested by the authors. The iteration number of the operator update step was 10000, the initial step size $\alpha = 10^{-7}$ and $\rho = 0.67$. For the cosparse signal update step, the iteration number was 1000, and $\gamma = \lambda = 0.3$.

Following the experiments in [2], we used the percentage of the recovered atoms to measure the performance of the algorithms for recovering the ground-truth dictionary. A row $\boldsymbol{\omega}_j$ in the true dictionary $\mathbf{\Omega}$ is regarded as recovered if [2]

$$\min_i (1 - |\hat{\boldsymbol{\omega}}_i \boldsymbol{\omega}_j^T|) < 0.01 \tag{10}$$

where $\hat{\boldsymbol{\omega}}_i$ are the atoms of the trained dictionary.

In addition, we introduce an operator $\|\mathbf{x}\|_0^\epsilon$ to count the number of the elements in $\mathbf{x} \in \mathbb{R}^p$ that are below the threshold ϵ, i.e.

$$\|\mathbf{x}\|_0^\epsilon = \#\{i : |x_i| < \epsilon, i = 1, 2, ..., p\} \tag{11}$$

where x_i denotes the ith element of $\mathbf{x}$. We can get the cosparsities of the training signals in the trained dictionaries by applying this operator to their representation vectors. Here we use the threshold $\epsilon = 0.001$. The average cosparsity of all the training signals is used to measure the quality of the trained dictionaries, as the final goal of ADL is to find a dictionary that can sparsely represent the training signals.

The percentage of the recovered atoms and the average cosparsity over iterations are plotted in Fig. 1. For the recovery rate, the Analysis SimCO outperforms (NL)AOL/

[1] Due to high computational complexity, we performed 300 iterations of the Analysis K-SVD and (NA)AOL which took nearly 8 hours and 32 hours respectively.

(NA)AOL and it is slightly lower than LOST and Analysis K-SVD, in both noiseless and noisy case. The average cosparsity obtained by the Analysis SimCO is closer to the ground-truth than LOST and (NL)AOL/(NA)AOL, but slightly worse than Analysis K-SVD. These results demonstrate that our proposed algorithm offers competitive performance as compared with these baseline algorithms.

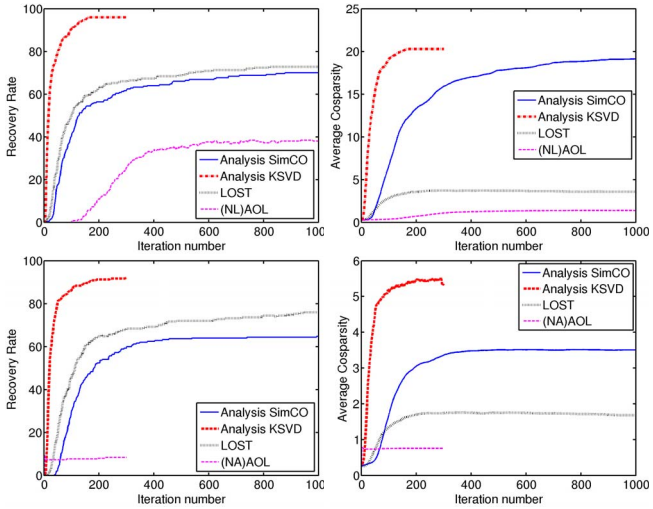

Fig. 1. Recovery Rate and Average Cosparsity over iterations in noiseless case (top) and noisy case (bottom).

Although the results of Analysis K-SVD for the two metrics are the best, its computational load is much higher than the other algorithms except (NA)AOL. The average running time per iteration is 93.77s for Analysis K-SVD (using parallel computation with two work-stations when performing OBG), 0.21s for (NL)AOL, 385.66s for (NA)AOL, and 1.78s for LOST respectively. The Analysis SimCO needs on average only 0.56s per iteration.[2] Even though more iterations of the Analysis SimCO were applied, the running time for the Analysis SimCO is still much shorter than that of the Analysis K-SVD and (NA)AOL. The higher efficiency of the Analysis SimCO results from the optimisation framework that can update multiple rows simultaneously in each iteration.

4.2. Experiments With Real Data

In the experiments with real data, we applied the analysis dictionary learning algorithms to image denoising. Four images, "Boats", "House", "Lena' and "Peppers" were manually corrupted by additive white Gaussian noise of zero mean and standard deviation σ. We tested the cases of three different noise levels $\sigma = 5,\ 25,\ 45$. For each noisy image, 7×7 patches were extracted and used as samples to learn an overcomplete dictionary of size 63×49. The proposed algorithm, Analysis K-SVD [2], LOST [7] and (NA)AOL [5] were applied respectively to learn analysis dictionaries, and then the analysis pursuit algorithm OBG [2] was applied to reconstruct the images with the trained dictionaries. The error-threshold of OBG for image recovery was $\epsilon = 1.15\sqrt{m}\sigma$. The algorithms were still initialized with a data-driven dictionary, same as the experiments with synthetic data.

Table 1. The Denoising PSNR Results in decibels.

σ (PSNR)	Algorithm	Boats	House	Lena	Peppers
5 (34.15)	Analysis SimCO	36.65	38.25	38.16	37.43
	Analysis K-SVD	**37.12**	**39.12**	**38.45**	**37.82**
	LOST	36.76	38.46	38.16	37.47
	(NA)AOL	36.91	38.56	37.94	37.41
25 (20.20)	Analysis SimCO	**28.61**	**30.53**	**30.75**	**28.75**
	Analysis K-SVD	28.43	30.32	30.47	28.67
	LOST	28.58	30.41	30.65	28.67
	(NA)AOL	27.76	29.57	29.25	27.87
45 (15.07)	Analysis SimCO	**25.98**	**27.45**	**27.88**	**25.68**
	Analysis K-SVD	25.83	27.28	27.74	25.51
	LOST	25.97	27.13	27.74	25.56
	(NA)AOL	24.76	25.87	26.21	24.52

The iteration number of the Analysis K-SVD was 100 and other three algorithms were applied 200 iterations[3]. The cosparsity of the Analysis SimCO was set as $l = 42$ while the corresponding parameters of the Analysis K-SVD algorithm and LOST were $r = 7$ and $s = 21$ receptively, in order to guarantee a fair comparison. Other parameters of LOST were $\alpha = 10^{-11}$, $\eta = \lambda = \mu = 10^5$, $p = 20$. The same parameters were used for (NA)AOL as in the experiment for the synthetic data. Table 1 summarizes the denoising results (image PSNR in dB) averaged over 10 independent tests.

As can be seen from Table 1, the results of all algorithms are very close to each other in general. In the lower noise level cases when $\sigma = 5$, the Analysis K-SVD algorithm gets the best results. When the noise level increases, the proposed Analysis SimCO algorithm outperforms the other three baseline algorithms, showing that the proposed algorithm is potentially more robust to noise corruption.

5. CONCLUSION

We have proposed an analysis dictionary learning algorithm: Analysis SimCO. The dictionary learning process is formulated as an optimisation problem with the sparsity and unit ℓ_2-norm constraints of the atoms in the dictionary. The algorithm iteratively solves this problem by thresholding and the gradient descent method on manifolds. Experimental results on both synthetic and real data demonstrated the competitive performance in recovery rate, average cosparsity, computational efficiency and image denoising, as compared with three baseline algorithms.

[2] All algorithms were implemented in Matlab R2012a and performed with an Intel Core i5 CPU at 3.30GHz and 8GB memory.

[3] The iteration numbers were set empiracally based on the computational time and convergence of the algorithms.

6. REFERENCES

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

[2] R. Rubinstein, T. Peleg, and M. Elad, "Analysis K-SVD: A dictionary-learning algorithm for the analysis sparse model," *IEEE Trans. Signal Process.*, vol. 61, no. 3, pp. 661–677, 2013.

[3] K. Engan, S.O. Aase, and J.H. Hakon, "Method of optimal directions for frame design," in *Proc. IEEE Int. Conf. on Acoust., Speech, and Signal Processing*, 1999, vol. 5, pp. 2443–2446.

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

[5] M. Yaghoobi, S. Nam, R. Gribonval, and M.E. Davies, "Constrained overcomplete analysis operator learning for cosparse signal modelling," *IEEE Trans. Signal Process.*, vol. 61, no. 9, pp. 2341–2355, 2013.

[6] S. Hawe, M. Kleinsteuber, and K. Diepold, "Analysis operator learning and its application to image reconstruction," *IEEE Trans. Image Process.*, vol. 22, no. 6, pp. 2138–2150, 2013.

[7] S. Ravishankar and Y. Bresler, "Learning overcomplete sparsifying transforms for signal processing," in *Proc. IEEE Int. Conf. on Acoust., Speech, and Signal Processing*, 2013, pp. 3088–3092.

[8] S. Ravishankar and Y. Bresler, "Learning sparsifying transforms," *IEEE Trans. Signal Process.*, vol. 61, no. 5, pp. 1072–1086, 2013.

[9] J.A. Tropp and A.C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. Inf. Theory*, vol. 53, no. 12, pp. 4655–4666, 2007.