A Fast Dictionary Learning Algorithm for Image Denoising
Hai-yang LI*, Chao YUAN and Heng-yuan WANG
School of Science, Xi'an Polytechnic University, Xi'an 710048, P.R. China
*Corresponding author

Keywords: Sparse presentation, Dictionary learning, Graph Laplacian, Clustering.

Abstract. The K-SVD is one of the well-known and effective methods to learn a universal and overcomplete dictionary. However, K-SVD is very expensive because many iteration steps are needed. What’s more, when it converts 2D data patches into 1D vectors for training or learning, K-SVD breaks down the inherent geometric structure of the data. To overcome these limitations, employing a subspace partition technique, we propose an efficient and fast algorithm, the fast top-bottom two-dimensional subspace partition algorithm, for learning overcomplete dictionaries. Experimental simulations demonstrate that our dictionary learning approach is effective for image denoising.

Introduction

Sparse and redundant representation modeling has recently received extensive research interest and found successful applications in compressive sensing[1], image processing tasks (compression, denoising, zooming)[2], linear regression and variable selection[3]. The sparse representation problem describes that a signal can be approximated 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[4]. This dictionary is overcomplete, that is, the number of atoms is more than the dimension of each atom. More precisely, a target signal $y \in \mathbb{R}^N$ is described as $y \approx Dx$, where $D \in \mathbb{R}^{N \times M}$ is an overcomplete dictionary and $x$ is a vector containing the representation coefficient of $y$. We are seeking the sparsest solution $x$, the one with the fewest nonzero entries. This can be formulated as the problem

$$
\min_x \|y-Dx\|_2^2 \quad \text{s.t.} \quad \|x\|_0 \leq k
$$

where $\|x\|_0$ stands for the so called $l_0$ norm that counts the number of nonzero elements of $x$, and $k$ stands for the maximum number of nonzero elements. Exact determination of sparsest representations is known to be an NP-hard problem. Thus a number of algorithms have been proposed to provide the sparsest approximation of a signal, including Orthogonal Matching Pursuit (OMP)[5] and Basis Pursuit (BP)[6]. More concretely, given a training data matrix $Y \in \mathbb{R}^{N \times M}$, containing $M$ signals $\{y_i \in \mathbb{R}^N\}_{i=1}^M$, dictionary learning is a procedure to find a dictionary $D \in \mathbb{R}^{N \times M}$. The solution can be obtained by solving the following problem

$$
\min_{D,X} \sum_{i=1}^M \|y_i - Dx_i\|_2^2 \quad \text{s.t.} \quad \|x_i\|_0 \leq k_0
$$

Most dictionary learning algorithms perform two stages[7]: sparse coding and dictionary update. In the sparse coding stage, keeping dictionary $D$ fixed, $X$ is computed by solving (1). In Dictionary update stage, with a fixed $X$, the dictionary $D$ is updated to reduce the approximation error.

The main difference among most dictionary learning algorithms, such as K-SVD algorithm[8], the Dictionary Pair Learning on the Grassmann-manifold algorithm (DPLG)[9] and first dictionary learning (FDL)[10], is the way of updating the dictionary. However, K-SVD algorithm is expensive. In particular, many iteration steps are needed since the atoms of the dictionary are updated component by component and the K-SVD updates each atom along with the coefficients in $D$ that multiply it using

256
singular value decomposition (SVD). It is worth noting that the K-SVD convert 2D data patches into 1D vectors for training and learning, and this conversion breaks down the inherent structure of the data. To overcome these limitation, other methods for dictionary learning came up to replace the K-SVD. For example, Liu et al.[10] and Zeng et al.[11]. Zeng et al. in[11] proposed a dictionary pair learning model (DPL model) for image denosing and designed a corresponding algorithm, called the DPLG algorithm. This algorithm learned an initial dictionary pair by the Top-bottom Two-dimensional Subspace Partition algorithm (TTSP algorithm). The methods of first dictionary learning (FDL) was presented in Liu et al.[10]. The partitioning procedure in FDL is equivalent to the first part of the TTSP algorithm and construction dictionary is different from the second part of the TTSP algorithm.

Motivated by ideas in[10] and[11], we propose a fast dictionary learning algorithm for image denosing. Our method is also a two-stage approach that includes dictionary learning stage and sparse coding stage, in which dictionary learning stage is different from the TTSP algorithm in[11] and sparse coding stage is similar to the method in[12] by adding the intrinsic geometric structure of the data through a graph regularized and using the learned dictionary to provide a sparse representation of data patches.

The paper is organized as follows. In Section 2 we describe the graph Laplacian and then provide a brief description of the graph-based dictionary learning. The corresponding optimization algorithm, a fast top-bottom two-dimensional subspace partition algorithm (FTTSP algorithm), is presented in Section 3. Section 4 presents some experiment results. Finally, the paper is concluded in Section 5.

Sparse Coding by Learned Dictionary and Graph Regularization

Recall that sparse coding, keeping the fixed learned dictionary $D$, tries to find a sparse coefficient matrix $X$ by solving (1). However, most of the existing approaches to sparse coding fail to consider the geometrical structure of the data space. In[12], Zheng et al. propose a graph based algorithm, called graph regularized sparse coding (GraphSC), to learn the sparse representations that explicitly take into account the local manifold structure of the data. Specifically, the graph Laplacian is incorporated into the sparse coding objective function as a regularizer. In this way, the obtained representations vary smoothly along the geodesics of the data manifold. By preserving locality, GraphSC can have more discriminating power compared with traditional sparse coding algorithms. Here we follow ideas in Zheng et al.[12] and introduce GraphSC in the following.

For the given set of training patches $\{Y_1, \ldots, Y_m\}$, we construct a weighted undirected complete graph $G(V, E, W)$, where the finite set $V$ of $m$ vertices represents the given patches, $V = \{Y_1, \ldots, Y_m\}$. Further, $E = V \times V$ is a set of weighted edges, and these weights are collected in the weight matrix $W \in \mathbb{R}^{m \times m}$. We define the symmetric weight matrix $W = (W_{i,j})_{i,j=1}^m$ by

$$W_{ij} = \begin{cases} \frac{1}{2\pi h^2} \exp \left(-\frac{\|Y_i - Y_j\|_F^2}{2h^2}\right) & \text{for } i \neq j \\ 0 & \text{for } i = j \end{cases}$$

using the Gaussian kernel and parameter $h$. The degree of each vertex $Y_i$, the number of all edges with weight to the vertex $Y_i$ is given by $D_i = \sum_{j=1}^m W_{i,j}$. Introducing the diagonal degree matrix $D = \text{diag}(D_1, \ldots, D_m)$, the graph Laplacian of $G$ is now given by $L = D - W$ to achieve a sparce matrix $L$. Hence, $L$ is symmetric and positive semi-definite, with non-diagonal entries being non-positive, and the sum of all entries in each column (or row) is zero.

A direct computation shows that

$$\text{Tr}(YLY^T) = \sum_{i,j} W_{i,j} \|Y_i - Y_j\|_F^2 = \sum_{i,j} W_{i,j} \|Y_i - Y_j\|_2^2 = \sum_{i,j} \|Y_i - Y_j\|_F^2$$

(4)

257
measuring the similarity of neighborhood patches in the graph, where we have used the notation \( Y_i \sim Y_j \). For each \( j \), the vector \( Dx_j \) is assumed to be a good approximation of \( y_j \).

Since the transform matrix \( D \) induces a linear mapping, we can suppose that the vectors \( x_i, i=1,\cdots,m \) possess a similar topological structure as \( y_i, i=1,\cdots,m \), and particularly that, if \( y_i \) and \( y_j \) are \( k \)-neighbors with a small distance \( \|y_i - y_j\|_2 \), we also have that \( \|x_i - x_j\|_2 \) is small. Therefore, we incorporate the term

\[
Tr(XLX^T) = \sum_{i,j=1}^{m} W_{i,j} \|x_i - x_j\|_2^2 = \sum_{i,j=1}^{m} \|x_i - x_j\|_2^2
\]

and obtain the new minimization problem

\[
\min_{x \in \mathbb{R}^{m \times D}, D \in \mathbb{R}^{D \times m}} \frac{1}{2}\|Y - DX\|_F^2 + \frac{\alpha}{2} Tr(XLX^T) + \lambda \|X\|_1
\]

where the Laplacian matrix \( L \) only depends on the training data \( Y \) that generates the graph.

The dictionary learning algorithm by employing GraphSC, called dictionary learning based on graph regularization, is outlined in Algorithm 1.

**Algorithm 1** Dictionary learning based on graph regularization

**Input:** Training data \( Y = [Y_1, \cdots, Y_m] \); Parameters \( \alpha \) and \( \lambda \)

**Procedures:**

1. Compute the graph Laplacian \( L \) for the given training set \( Y \).
2. Determine the dictionary \( D \) by a dictionary learning algorithm based on \( Y \).
3. Solve the minimization problem \( \min_{x \in \mathbb{R}^{m \times D}, D \in \mathbb{R}^{D \times m}} \frac{1}{2}\|Y - DX\|_F^2 + \frac{\alpha}{2} Tr(XLX^T) + \lambda \|X\|_1 \).
4. Reconstruct the data \( Y_D = DX \).

Loop steps until the given number of iterations is achieved.

**Output Data** \( Y_D \).

In the following, we introduce the algorithm for solving the third step in Algorithm 1 briefly. That is to say, we will solve the minimization problem

\[
\min_{x \in \mathbb{R}^{m \times D}, D \in \mathbb{R}^{D \times m}} \frac{1}{2}\|Y - DX\|_F^2 + \frac{\alpha}{2} Tr(XLX^T) + \lambda \|X\|_1
\]

for given (noisy) training data \( Y \) and the dictionary \( D = [d_1, \cdots, d_m] \), where \( d_i = \text{col}D_i \) (the vectorized \( D_i \)) are the dictionary elements constructed in above. We suggest here to solve the problem using the split Bregman iteration see e.g. Goldstein and Osher[13]; Plonka and Ma[14] which is in the considered case equivalent to the Alternating Direction Method of Multipliers (ADMM), see Yankelevsky and Elad[15]. We outline the algorithm in the following.

**Algorithm 2** Graph regularized sparse coding

**Input:** Training data \( Y = [Y_1, \cdots, Y_m] \); Laplacian matrix \( L \); Learned dictionary \( D \); \( X^0 = Z^0 = B^0 = 0 \); Parameters \( \lambda, u, \alpha > 0 \); Number of iterations

**Algorithm**

Iterate until the given number of iterations is achieved:

1. Compute \( X^{t+1} \) as the solution to \( (D^T D + uI)X + \alpha XL = D^T Y + u(Z^t - B^t) \).
2. Compute \( Z^{t+1} \) componentwisely by employing shrinkage \( z_{i,j}^{t+1} = T_{\lambda, \mu}(x_{i,j}^{t+1} + B_{i,j}^t) \).
3. Update \( B^{t+1} = B^t - Z^{t+1} + X^{t+1} \)

**Output** \( X \)
A Fast Dictionary Learning Method

Here we will propose a fast dictionary learning method which is based on a special partition tree structure. Our method is also a two-stage approach that includes dictionary learning stage and sparse coding stage. In dictionary learning stage, we construct the dictionary in two steps. We first obtain a tree structure to partition the set of our training patches and then construct the dictionary based on the computed subset partitions in the tree, using a fast top-bottom two-dimensional subspace partition algorithm (FTTSP algorithm). The first step, the tree construction, is different from the first part of the TTSP algorithm, while the second step, constructing the dictionary, is equivalent to the second part of method SDL. In sparse coding stage, we add the intrinsic geometric structure of the data through a graph regularized and use the learned dictionary to provide a sparse representation of data patches. The FTTSP algorithm is outlined in Algorithm 3.

Step1: Construction of the partition tree. For the given training set \( Y_1, \cdots, Y_m \in \mathbb{R}^{n\times n} \) of image patches. We compute the mean

\[
C := \frac{1}{m} \sum_{i=1}^{m} Y_i \in \mathbb{R}^{n\times n}
\]

and the two non-symmetric \((n\times n)\) covariance matrices

\[
C_L := \frac{1}{m} \sum_{i=1}^{m} (Y_i - C)(Y_i - C)^T \quad C_R := \frac{1}{m} \sum_{i=1}^{m} (Y_i - C)^T (Y_i - C)
\]

Now, the normalized eigenvectors \( u_1, u_2 \) and \( v_1, v_2 \) corresponding to the first two eigenvalues of \( C_L \) and \( C_R \) is computed. By “the first two eigenvalues” we refer to the two largest eigenvalues

\[
u := \arg \max_{\|x\|_1 = 1} x^T C_L x, \quad v := \arg \max_{\|x\|_1 = 1} x^T C_R x,
\]

representing the main structures of the training patches being not captured by the mean patch \( C \). We compute the numbers

\[
s_{i1} = u_1^T Y_i v_1, \quad s_{i2} = u_2^T Y_i v_2, \quad i = 1, \cdots, m. \tag{10}
\]

These numbers \( \{ s_{i1} | i = 1, \cdots, m \} \) and \( \{ s_{i2} | i = 1, \cdots, m \} \) give us a measure, how strong each single patch is correlated to the found structure and will be used to obtain a partition of the set of all patches \( \{Y_1, \cdots, Y_m\} \) into four partial sets.

First, in the first level, we partition the one-dimensional real number \( \{ s_{i1} | i = 1, \cdots, m \} \) into two clusters \( \{ s_{i1}^{j1} \} \) and \( \{ s_{i1}^{j2} \} \) by K-means, in which \( j1 \cup j2 = \{1, \cdots, m\} \) and \( j1 \cap j2 = \emptyset \). Then, in the second level, we partition the one-dimensional real number sets \( \{ s_{i2} | i \in j1 \} \) and \( \{ s_{i2} | i \in j2 \} \) into two clusters \( \{ s_{i2}^{j11} \}, \{ s_{i2}^{j12} \} \) and \( \{ s_{i2}^{j21} \}, \{ s_{i2}^{j22} \} \) by K-means respectively, in which \( j1 \cup j2 = j1, j21 \cup j22 = j2 \) and \( j1 \cap j2 = \emptyset \). Therefore, \( \{Y_1, \cdots, Y_m\} = \{Y_{j1}\} \cup \{Y_{j2}\} \) and \( \{Y_{j1} = \{Y_{j11}\} \cup \{Y_{j12}\}, \{Y_{j2} = \{Y_{j21}\} \cup \{Y_{j22}\}. \)

In this way, we can divide two level tree structure with every calculation.

Remarks. 1. Having found this first partition, we can proceed to partition the obtained subsets further, using the same scheme. This procedure yields a binary tree, where we stop the further partition of a subset, if it contains two training data that automatically separate the two classes.

2. Since two level tree structures are obtained in every calculation, the algorithm of our article speeds up TTSP algorithm.

3. If we yield a binary tree with the first three eigenvalues or more eigenvalues in above procedure, then the algorithm has a better convergence rate but a weaker performances such as structural similarity index (SSIM), peak signal to noise ratio (PSNR) and root mean square error (RMSE) in general.
Step 2: Determine the dictionary from the partition tree. Each knot $k$ in the tree is now associated with a subset of training patches $\{Y_j\}_{j=1}^m$, where $\wedge_k \subseteq \{1, \cdots, m\}$ denotes the subset of indices of these patches. We assume that the root of the tree has the knot number $k=1$ (i.e. $\wedge_1 = \{1, \cdots, m\}$) and we proceed numbering by going through each level from left to right. For each knot $k$, we compute the mean (center)

$$C_k = \frac{1}{|\wedge_k|} \sum_{Y_j \in \wedge_k} Y_j$$

and the normalized singular vectors to the maximal singular value of $C_k^T C_k$ and $C_k^T C_k$, i.e.

$$u_k := \arg \max_{\|x\|=1} C_k^T C_k x \quad \quad v_k := \arg \max_{\|x\|=1} C_k^T C_k x .$$

If $\lambda_k$ denotes the maximal singular value of $C_k$, then $\lambda_k u_k v_k^T$ is the best rank-1 approximation of $C_k$, since $u_k$ and $v_k$ are the first vectors in the singular value decomposition of $C_k$.

The dictionary is now determined as follows. We fix the first dictionary element $D_1 = u_1 v_1^T$ capturing the main structure of the mean $C = C_1$. Further, for each pair of children knots $2k$ and $2k+1$ to the same parent with means $C_{2k}$ and $C_{2k+1}$, we set

$$D_k := D_k^* = \lambda_{2k} u_{2k} v_{2k}^T - \lambda_{2k+1} u_{2k+1} v_{2k+1}^T, \quad D_k := \frac{D_k^*}{\|D_k^*\|_F}$$

thereby capturing the difference of main structures of $C_{2k}$ and $C_{2k+1}$. Let $d_i = \text{col} D_i$ (the vectorized $D_i$, $i = 1, \cdots, m$), we construct the dictionary $D = [d_1, \cdots, d_m]$.

Algorithm 3 (FTTSP algorithm) Fast top-bottom two-dimensional subspace partition algorithm

Input: Training image patches, the maximum depth of the binary tree.

Procedures:

1: The first node is the root node including all image patches.

2: For all image patches in the current leaf node, run the following 1)-4) steps:

1) Compute eigenvectors $u_1, v_1$ and $u_2, v_2$ corresponding to the first two eigenvalues of the two covariance matrices.

2) Compute the two-dimensional projection representations of all image patches from this node, that is, $s_{ij} = u_i^T Y_{ij}$ and $s_{ij} = u_i^T Y_{ij}, i = 1, \cdots, m$.

3) Partition the one-dimensional real number set $\{s_{1j}\}$ into two clusters $\{s_{1j}^{(1)}\}$ and $\{s_{1j}^{(2)}\}$ by K-means. Then partition the image patches corresponding to these two clusters into the left child $\{Y_{ij}^{(1)}\}$ and the right child $\{Y_{ij}^{(2)}\}$. Simultaneously the depth of the node is added one.

4) Partition the one-dimensional real number sets $\{s_{1j}^{(1)}\}$ into two clusters $\{s_{1j}^{(11)}\}$ and $\{s_{1j}^{(12)}\}$ by K-means. Then partition the image patches $\{Y_{ij}^{(1)}\}$ corresponding to these two clusters into the left child $\{Y_{ij1}\}$ and the right child $\{Y_{ij2}\}$. Partition the image patches $\{Y_{ij2}\}$ in an analogous manner. Simultaneously the depth of the node is added one.

3: IF the depth of the node is larger than the maximum depth or the number of image patches in this leaf node is smaller than the row number or column number of the image patches, THEN stop the partition. ELSE repeat Step 2 recursively for the left child node and the right child node.
4: Compute the dictionary for each node by the following 1)-3) steps:
  1) Compute the center \( C \) and the normalized singular vectors \( u_1 \) and \( v_1 \) to the maximal singular value of \( C^T C \) and \( C^T C \) in the root node. Construct the first atom of dictionary \( D_1 = u_1 v_1^T \).
  2) For each pair of children knots \( 2k \) and \( 2k+1 \) to the same parent with means \( C_{2k} \) and \( C_{2k+1} \), compute the normalized singular vectors \( u_{2k}, v_{2k} \) and \( u_{2k+1}, v_{2k+1} \) to the maximal singular value \( \lambda_{2k} \) and \( \lambda_{2k+1} \).
  3) Compute the atom \( D_k := \frac{D_k^*}{\|D_k\|} \), where \( D_k^* := \lambda_{2k} u_{2k} v_{2k}^T - \lambda_{2k+1} u_{2k+1} v_{2k+1}^T \).

5: Collect the atom of all leaf nodes into the dictionary \( D = \{d_1, \cdots, d_m\} \), where \( d_i = \text{col}D_i \) (the vectorized \( D_i \)).

Output the dictionary \( D \).

Experiments

In this section, we present experiments to evaluate the dictionary performance of our proposed algorithm compared with other algorithms introduced in the paper. In the part, we present experimental result, with the aim of training a dictionary for sparsely representing natural image patches. We then turn to test the image denoising performance of the dictionary learned by our approach.

Our simulations were performed in MATLAB R2010b environment on a system with 3.8 GHz CPU and 4 GB RAM, under Microsoft Windows 7 operating system. As a rough measure of complexity, we will mention the running times of the algorithms.

We show the improvement achieved by applying the above methods to the image denosing problem. In this set of experiments, we employ the methodology[8] given by Elad. We choose three well-known images of size 256×256 as test images, including “Barbara”, “Boat”, “House”. Each image is contaminated by artificially adding zeros-mean white Gaussian noise at five different variances.

An objective image quality metric plays an important role in image denoising applications. Currently, three classical image quality assessment metrics are typically used: the Peak Signal-to-Noise Ratio (PSNR) and the measure of Structural Similarity (SSIM). We cut the noised image into small patches of size 8×8. The regularization parameters have been empirically chosen to be \( \alpha = 1.6; \lambda = \frac{30}{\sigma}; u = 0.05 \). It is also noted that the denoised image is obtained with an average constant.
### Table 1. PSNR values of the denoised results.

| Algorithm | Image | $\sigma$ | $\sigma$=5 | $\sigma$=10 | $\sigma$=15 | $\sigma$=20 | $\sigma$=25 |
|-----------|-------|---------|-----------|-----------|-----------|-----------|-----------|
|           |       | PSNR    | TIME      | PSNR      | TIME      | PSNR      | TIME      |
| K-SVD     |   | 38.063  | 114.43    | 34.439    | 75.41    | 32.339    | 46.34    |
|           |   | 30.807  | 34.57    | 29.554    | 29.38    |
| FDL       |   | 38.287  | 16.51    | 34.899    | 11.39    | 32.765    | 9.53    |
|           |   | 31.521  | 8.36    | 30.575    | 7.43    |
| Ours      |   | 38.386  | 11.38    | 34.537    | 9.26    | 32.456    | 7.38    |
|           |   | 31.582  | 6.71    | 30.521    | 6.02    |

### Table 2. SSIM values of the denoised result.

| Image | $\sigma$ | K-SVD | TTPS | Ours |
|-------|---------|-------|------|------|
|       | 5       | 0.962 | 0.965 | 0.966 |
| Barbara | 10      | 0.935 | 0.940 | 0.938 |
|         | 15      | 0.911 | 0.919 | 0.912 |
|         | 20      | 0.883 | 0.904 | 0.905 |
|         | 25      | 0.849 | 0.886 | 0.885 |
| Boat   | 5       | 0.941 | 0.935 | 0.938 |
|         | 10      | 0.885 | 0.886 | 0.888 |
|         | 15      | 0.851 | 0.854 | 0.850 |
|         | 20      | 0.818 | 0.826 | 0.823 |
|         | 25      | 0.772 | 0.790 | 0.781 |
| House  | 5       | 0.956 | 0.957 | 0.958 |
|         | 10      | 0.906 | 0.922 | 0.922 |
|         | 15      | 0.877 | 0.891 | 0.886 |
|         | 20      | 0.860 | 0.874 | 0.873 |
|         | 25      | 0.846 | 0.860 | 0.863 |
Table 1 presents the final denoising PSNR results obtained from K-SVD, FDL algorithms with additionally the fixed ours methods. The SSIM results of the three test methods are reported in Table 2. Figure displays the original, noisy and denoised “Barbara”, “Boat”, “House”, images for noise level $\sigma=20$. Based on these results, we can observe that our proposed algorithm, FTTSP algorithm, and FDL algorithm in general not only cost less time but also provide higher PSNR result and SSIM values in image denoising compared with the K-SVD algorithm. Although FTTSP algorithm and FDL algorithm have the similar results in general, it is noticeable that FTTSP algorithm needs less time compared with FDL.

Summary
In the paper, we present a fast top-bottom two-dimensional subspace partition algorithm (FTTSP algorithm) for learning overcomplete dictionary, which is based on a special partition tree structure. In construction of the partition tree step, our algorithm can obtain two level tree structures in every calculation, and hence it costs less time than the first part of the TTSP algorithm and FDL. It is equivalent to the second part of method SDL in construction dictionary step. Experimental results on synthetic data and image patches illustrate that FTTSP and FDL not only have higher quality but also cost less time than K-SVD, and that at the same time, FTTSP needs less time than FDL. In the future, we will consider improving FTTSP algorithm and applying it to other applications.

Acknowledgement
The authors would like to thank editorial and referees for their comments which help us to enrich the content and improve the presentation of the results in this paper. The work was supported by the National Natural Science Foundations of China (11271297) and the Science Foundations of Shaanxi Province of China (2015JM1012).
References

[1] D.L. Donoho, Compressed sensing, IEEE Trans on Information Theory. 52(4) (2006) 1289-1306.
[2] M. Elad, M.A.T. Figueiredo and Y. Ma, On the role of sparse and redundant representations in image processing, Proceedings of the IEEE, 98 (6) (2010), 972–982.
[3] R. Tibshirani, Regression shrinkage and selection via the lasso, J. Royal. Statist. Soc B. 58(1) (2006) 267-288.
[4] S. Mallat and Z. Zhang, Matching pursuits with time-frequency dictionaries, IEEE Trans on Signal Proc. 41(12) (1993), 3397-3415.
[5] Y. Pati, R. Rezaiifar, P. Krishnaprasad, Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition, in Proceedings of the 27th Asilomar Conference on Signals, Systems & Computers. (1993) 40-44.
[6] S. Chen, D. Donoho, M. Saunders, Atomic decomposition by basis pursuit, SIAM Review. 43(1) (2001), 129-159.
[7] W. Dai, T. Xu, W. Wang, Simultaneous codeword optimization (SimCO) for dictionary update and learning, IEEE Trans on Signal Process. 60(12) (2012) 6341-6353.
[8] M. Elad, M. Aharon, Image denoising via sparse and redundant representations over learned dictionaries, IEEE Trans on Image Process. 15 (2006) 3736-3745.
[9] X. Zeng, W. Bian, W. Liu, J. Shen and D. Tao, Dictionary pair learning on Grassman manifolds for image denoising, IEEE Trans on Image Processing. 24 (11) (2015) 4556-4569.
[10] L. Liu, J. Ma, G. Plonka, Graph regularized seismic dictionary learning, http://num.math.uni-goettingen.de/plonka/pdfs/GraphDLrevised.pdf.
[11] X. Zeng, W. Bian, W. Liu, J. Shen and D. Tao, Dictionary pair learning on Grassman manifolds for image denoising, IEEE Trans on Image Processing. 24 (2015) 4556-4569.
[12] M. Zheng, J.J. Bu, C. Chen and C. Wang, Graph regularized sparse coding for image representation, IEEE Trans on Image Processing. 20(5) (2011), 920-930.
[13] T. Goldstein and S. Osher, The split Bregman method for L1-regularized problems, SIAM J. Imaging Sciences. 2(2) (2009) 323-343.
[14] G. Plonka and J. Ma, Curvelet-wavelet regularized split Bregman iteration for compressed sensing, Int. J. Wavelets Multiresolut Inf. Process. 9(1) (2011) 79-110.
[15] Y. Yankelevsky and M. Elad, Dual graph regularized dictionary learning, IEEE Transactions on Signal and Information Processing over Networks. 2(4) (2016) 611-624.