Graph regularized seismic dictionary learning

Lina Liu\textsuperscript{1}, Jianwei Ma\textsuperscript{2}, Gerlind Plonka\textsuperscript{3}

\textbf{ABSTRACT}

We propose a new regularization method for sparse representation and denoising of seismic data. Our approach is particularly based on two ingredients, a sparse data representation in a learned dictionary and a similarity measure for image patches that is evaluated using the Laplacian matrix of a graph.

Dictionary learning (DL) methods aim to find a data-dependent basis or a frame that admits a sparse data representation while capturing the characteristics of the given data. We propose two algorithms for dictionary learning based on clustering and singular value decomposition (SVD), called first and second dictionary construction (FDC and SDC). Besides employing an adapted dictionary we also consider a similarity measure for the local geometric structures of the seismic data using the Laplacian matrix of a graph. The proposed method achieves better denoising performance than existing denoising methods, both in terms of peak signal-to-noise ratio values and visual estimation of weak-even preservation. Comparisons of experimental results on field data using traditional FX deconvolution (FX-Decon), a prediction process in frequency-space domain, and curvelet thresholding methods are also provided.

\textsuperscript{1}Department of Mathematics and Institute of Geophysics, Harbin Institute of Technology, Harbin, China, lina.liu@mathematik.uni-goettingen.de
\textsuperscript{2}Department of Mathematics and Institute of Geophysics, Harbin Institute of Technology, Harbin, China, jma@hit.edu.cn
\textsuperscript{3}Institute for Numerical and Applied Mathematics, University of Göttingen, Göttingen, Germany, plonka@math.uni-goettingen.de
INTRODUCTION

Extracting the fundamental structures of seismic data by sparse representations and denoising are two of the most fundamental, widely studied, and significant problems in seismic data processing. The two problems are closely related since for given noisy measurements, the extraction of the important data structures can be at the same time understood as a denoising process, where one has to separate the original seismic data from noise. Note that this process is an ill-posed inverse problem since noise consists of random values. Often, a first linear denoising (usually based on averaging of measurements) is already applied during the acquisition process in order to achieve data sets with lower noise level. To improve denoising results, regularization methods are required that provide stable and unique solutions. A regularization method is a technique for solving ill-posed inverse problems by formulating a minimization problem, where the objective function includes besides the fidelity term (that measures the distance to the given data) additional "regularization" terms that force special properties of the desired data. Therefore, regularization methods serve to impose desired features on subsurface images. The well-known Tikhonov regularization models force smoothness of the data and tend to produce models where discontinuities are blurred. They have been for instance applied to solve the acoustic inverse problem of crosshole seismology (Reiter and Rodi, 1996). Other regularization methods using total variation minimization can provide high-resolution images of the subsurface, where edges and discontinuities are properly preserved. For example, Anagaw and Sacchi (2012) applied total variation minimization to the problem of estimating acoustic velocity perturbations using a single scattering Born modeling operator. Another regularization method that has attracted a renewed interest and considerable attention in signal processing literature is $L_1$-norm regularization. Here a regularization term is included into the objective function that forces sparsity of the data in the spatial domain or in a transform domain. This approach is popular for seismic data processing methods such as seismic data denoising and interpolation (Herrmann and Hennenfent, 2008). However, the above mentioned methods do not exploit the full geometric structure characteristics of the data as e.g. the spatial
relative position of the events. To overcome this issue, additional regularization terms have to be incorporated in order to satisfy specific requirements in various scenarios.

In this paper, we propose a regularization approach with two additional constraints,

a) forcing sparsity of the data in a learned (data-dependent) dictionary,

b) minimizing a similarity measure that is evaluated by a Laplacian matrix of a graph (graph regularization).

In the following, we explain these two new ingredients in more detail.

a) In this paper, dictionaries are bases or frames of the finite dimensional space $\mathbb{R}^N$ of real vectors of length $N$ or equivalently of $\mathbb{R}^{n \times n}$, the space of digital images with $n$ rows and $n$ columns, where $N = n^2$. A dictionary with $k$ elements (or atoms) is a set of $k$ vectors in $\mathbb{R}^N$ that can be simply represented by a matrix $D \in \mathbb{R}^{N \times k}$, where in this case the $k$ columns of length $N$ are the atoms of $D$. If $k \geq N$ and $D$ has full rank $N$, then all vectors $y \in \mathbb{R}^N$ can be represented as linear combinations of atoms in $D$, i.e., there exists an $x \in \mathbb{R}^k$ with $y = Dx$. If $k > N$ and $D$ has full rank $N$, then the representation of $y$ by dictionary atoms is not longer unique, and the dictionary is called over-complete. The dictionary matrix $D$ can be seen as a transform matrix, and one is usually interested to construct $D$ such that the given seismic data (either vectors or image patches) can be sparsely represented by this transform, i.e., the main structure can be already presented by a linear combination of a small number of atoms of the dictionary. Indeed, sparse transforms offer a nice way to implement sparse representations and denoising of seismic data. Whether denoising is effective or not strongly depends on dictionary selection or on the choice of the sparse transform. Wavelet transforms have been often used for solving seismic data denoising problems (see e.g. Chanerley and Alexander, 2002). Zhang and Ulrych (2003) presented a physical wavelet frame for seismic data denoising that used the special characteristics of seismic data. Nowadays, curvelets are one of most popular tools for seismic data representation and denoising (Hennenfent and Herrmann, 2006; Neelamani et al., 2008; Ma and Plonka, 2010). In addition to such sparse representation methods, Bonar and Sacchi (2012) applied a nonlocal means algorithm to seismic data denoising by
utilizing similar samples or pixels within the image regardless of their spatial proximity.

A new popular and highly effective approach for solving seismic denoising problems is the sparse representation of seismic data using a learned dictionary. The learned dictionary is adapted to the special structure of the given data. Compared to denoising methods by Pan et al., 1999; Starck et al., 2002 based on dictionaries being not data-dependent, a dictionary trained through a dictionary learning method can provide a sparser representation of seismic data. Different dictionary learning methods have already been applied to the seismic data denoising process. Bechouche and Ma (2014) used the method of optimal direction (MOD) to teach the dictionary (see also Engan et al., 1999), while Yu et al. (2015) proposed a dictionary learning seismic data denoising method that used a data driven tight frame (DDTF). However, almost all patch-based dictionary learning denoising methods convert seismic data patches or volumes into one-dimensional (1D) vectors for training, and thereby lose the inherent 2D or 3D geometric structures of the seismic data.

In this paper, we will propose two new methods for dictionary learning using the singular value decomposition (SVD) and a clustering method. Our approach connects ideas of Zeng et al. (2015) with ideas borrowed from generalized wavelet constructions. For the two methods, we will use training patches from the given seismic data to learn the dictionary. Particularly, for the denoising problem, the used patches are also noisy. The methods are simple to implement and may be of relevance not only for denoising but also for other seismic data applications. The first dictionary construction (FDC) employs the SVD of covariance matrices of similar training patches. The second dictionary construction (SDC) uses the similarity of training patches and a clustering method. The main idea of our two dictionary learning algorithms is the following. In the first step, we cluster the training patches into subsets according to a similarity measure using a binary tree structure. Then we build average patches (center matrices) for each subset. The dictionary elements are then constructed as linear combinations of small-rank approximations of the obtained average patches.
b) The second ingredient of our proposed regularization method is a term that measures similarity of patches. This similarity measure is determined by the Laplacian matrix of a graph and is therefore called graph regularization term. In recent years, graph regularization has been used for describing the relationship between image patches (Elmoataz et al., 2008; Bougleux et al., 2009; Kheradmand and Milanfar, 2014), e.g. by building a $K$-nearest neighbor graph to encode the geometric information in the data. Such graph-based methods have been used for image denoising (Tang et al., 2013; Yankelevsky and Elad, 2016), image representation (Zheng et al., 2011), and image super-resolution (Lu et al., 2012). In this work, we will particularly employ the assumption that if two training data patches are close with respect to the obtained similarity measure than also their sparse dictionary representations are close. Similar assumptions have been also used in various manifold learning methods to explore such structures (Belkin and Niyogi, 2003; Liu et al., 2014; Zheng et al., 2011).

Our complete denoising method can be understood as an iterative method. First we derive a dictionary from training patches which are patches of the given noisy seismic data. Then we present the noisy data sparsely in this dictionary and solve the minimization problem, where the graph regularization term is applied to the sparse data in the transformed (dictionary) domain. We repeat this process with the resulting denoised data patches that now serve in a next iteration to obtain an improved dictionary etc.

The rest of the paper is organized as follows: In the second section we present our regularization method. We describe the minimization problem obtained by incorporating the sparse representation in a dictionary and particularly derive the graph regularization term that measures the similarity between image patches. We summarize the complete denoising algorithm in the third section DENOISING METHOD, see Algorithm 1. This algorithm can be indeed applied with an arbitrary dictionary $D$. In the fourth section METHODS FOR DICTIONARY LEARNING we describe the two new graph-based dictionary learning methods FDC and SDC. Finally we deal with the problem, how to solve the complete minimization problem arising from this regularization approach in section METHODS TO
SOLVE THE MINIMIZATION PROBLEM, see Algorithm 2. The mathematical justification of Algorithm 2 can be found in the Appendix A. Experimental results and comparative discussions using two kinds of field data are provided in the section EXPERIMENTS. A conclusion and final thoughts are provided in the CONCLUSION. Appendix B contains an extensively described toy example for the dictionary learning methods.

SPARSE REPRESENTATION BY LEARNED DICTIONARIES AND GRAPH REGULARIZATION

Let us first introduce some notations. Let \( \{ Y_1, \ldots, Y_m \} \) (e.g. \( Y_n \in \mathbb{R}^{n \times n} \) consists of \( n \) traces each with \( n \) samples) be a given training set of 2D patches of seismic data. Further, let \( Y := [y_1, \ldots, y_m] \) be an \( N \times m \) matrix with \( N = n^2 \), where the columns \( y_j = \text{vec}(Y_j) \in \mathbb{R}^N \) are the vectorized patches, i.e., we stack all columns of \( Y_j \) into one vector \( y_j \) starting with the first. With \( D := [d_1 \ldots d_k] \in \mathbb{R}^{N \times k} \) we denote the dictionary (matrix), where each column \( d_i \in \mathbb{R}^N \) is one atom in the dictionary. Here, and elsewhere we use the notation := for (mathematical) definitions.

Once, we have determined a suitable dictionary \( D \), a sparsity promoting optimization problem can be formulated as

\[
\min_{X \in \mathbb{R}^{k \times m}} \left( \frac{1}{2} \| Y - DX \|_F^2 + \lambda \| X \|_0 \right), \tag{1}
\]

where \( X = [x_1, \ldots, x_m] \in \mathbb{R}^{k \times m} \) denotes the matrix of sparse coefficient vectors, such that the set of training patches \( Y = [y_1, \ldots, y_m] \) is sparsely represented by \( DX \). Here, \( \lambda \) is a regularization parameter, \( \| \cdot \|_F \) denotes the Frobenius norm of the matrix, and \( \| X \|_0 \) counts the number of non-zero entries of \( X \). Note that this optimization problem is NP-hard and cannot be exactly solved in polynomial time. Therefore, \( \| \cdot \|_0 \) is usually replaced by the convex norm \( \| \cdot \|_1 \), and the relaxed optimization problem reads

\[
\min_{X \in \mathbb{R}^{k \times m}} \left( \frac{1}{2} \| Y - DX \|_F^2 + \lambda \| X \|_1 \right), \tag{2}
\]

where \( \| X \|_1 := \sum_{i=1}^m \| x_i \|_1 = \sum_{i=1}^m \sum_{j=1}^k |x_{i,j}| \) is the sum of absolute values of all entries.
in $\mathbf{X}$. The model (2) has been widely used for data denoising in the last decade, and many algorithms have been developed to solve these optimization problems efficiently, see e.g. Beck and Teboulle (2009); Chambolle and Pock (2011); Needell and Vershynin (2010).

We extend this model for sparse representation of seismic images by adding two important ingredients, **dictionary learning** and **graph regularization**.

**Dictionary learning.** We want to employ a dictionary $\mathbf{D}$ that is adaptively learned from the data. In machine learning and in other applications, see e.g. Aharon et al. (2006); Elad and Aharon (2006); Dong et al. (2013), the following model has been studied already,

$$
\min_{\mathbf{X} \in \mathbb{R}^{k \times m}, \mathbf{D} \in \mathbb{R}^{N \times k}} \left( \frac{1}{2} \| \mathbf{Y} - \mathbf{D}\mathbf{X} \|_F^2 + \lambda \|\mathbf{X}\|_* \right)
$$

with $\| \cdot \|_*$ being either $\| \cdot \|_0$ or $\| \cdot \|_1$, and where K-SVD for dictionary learning is employed. The notion K-SVD has been coined by Aharon et al. (2006). This well-known dictionary learning method to solve (3) combines SVD and K-means clustering. The approach is based on alternating optimization: For a fixed $\mathbf{D}$, an improved sparse matrix $\mathbf{X}$ is computed e.g. by the orthogonal matching pursuit (OMP) method, a greedy algorithm that provides good solutions for (1). For fixed $\mathbf{X}$, the dictionary $\mathbf{D}$ is updated using singular value decomposition (SVD). However, K-SVD is very expensive (see e.g. Liu et al., 2017). In particular, many iteration steps are needed since the atoms of the dictionary are updated one by one, and each time the SVD of a large $N \times N$ matrix is needed. At the same time, two-dimensional structures of the training patches are not explicitly used but all patches are considered only in vectorized form.

Therefore, other methods for dictionary learning came up aiming at a cheaper technique to replace the K-SVD. For example, in Cai et al. (2014) and Liu et al. (2017) a priori dictionary structure is imposed to reduce the number of parameters, as e.g. block-wise Toeplitz structure or (directional) tensor-product frames. Running times of the three methods have been compared in Liu et al. (2017), and the two new methods are both significantly cheaper than the K-SVD method.
Motivated by ideas in Zeng et al. (2015), we will propose two different methods for adaptive dictionary learning in Section METHODS FOR DICTIONARY LEARNING. These methods try to exploit the two-dimensional geometric structure of the training data in a more direct way and are essentially cheaper than K-SVD.

**Graph regularization.** The second ingredient is an extension of the considered functional in (3) by an additional term that measures the similarity between the image patches. Here we follow ideas in Zheng et al. (2011) and Yankelevsky and Elad (2016) and employ a graph that represents the internal topology of the training patches.

For the given set of training patches $\mathbf{Y}_1, \ldots, \mathbf{Y}_m$, we construct a weighted undirected complete graph $G(V, E, W)$, where the finite set $V = \{\mathbf{Y}_1, \ldots, \mathbf{Y}_m\}$ of $m$ vertices represents the given patches. Further, $E = V \times V$ is a set of weighted edges, i.e., each two patches $\mathbf{Y}_i, \mathbf{Y}_j$ are connected by an edge, and the corresponding weights are collected in the weight matrix $W \in \mathbb{R}^{m \times m}$. We measure the similarity of the training patches $\mathbf{Y}_i$ and $\mathbf{Y}_j$ simply by the norm of its difference $\|\mathbf{Y}_i - \mathbf{Y}_j\|_F$. For a pre-defined $K$, we fix the $K$ nearest neighbors of $\mathbf{Y}_i$ (being different from $\mathbf{Y}_i$ itself) by inspecting $\|\mathbf{Y}_i - \mathbf{Y}_j\|_F^2$ for $j \in \{1, \ldots, i-1, i+1, \ldots, m\}$ and define the symmetric weight matrix $W = (W_{i,j})_{i,j=1}^m$ by $W_{i,j} = 1$ if $\mathbf{Y}_j$ is among the $K$ nearest neighbors of $\mathbf{Y}_i$ or if $\mathbf{Y}_i$ is among the $K$ nearest neighbors of $\mathbf{Y}_j$. Otherwise, we set $W_{i,j} = 0$. In particular, $W_{i,i} = 0$ for $i = 1, \ldots, m$. Thus, each row (or column) of $W$ contains at least $K$ ones. The process of graph building is illustrated in Figure 1. The degree of each vertex $\mathbf{Y}_i$, i.e., the number of all edges with weight 1 to the vertex $\mathbf{Y}_i$ is given by $\Delta_i = \sum_{j=1}^m W_{i,j}$. Introducing the diagonal matrix $\Delta = \text{diag}(\Delta_1, \ldots, \Delta_m) \in \mathbb{R}^{m \times m}$, the Laplacian matrix of the graph $G$ is now defined as the matrix $L = \Delta - W \in \mathbb{R}^{m \times m}$. By construction, $L$ is symmetric and positive semidefinite, its non-diagonal entries are non-positive, and the sum of all entries in each column (or row) is zero.

A direct computation shows that the term

$$\text{Tr}(\mathbf{Y} \mathbf{L} \mathbf{Y}^T) = \sum_{i,j=1}^m W_{i,j} \|\mathbf{Y}_i - \mathbf{Y}_j\|_F^2 = \sum_{i,j=1}^m W_{i,j} \|\mathbf{y}_i - \mathbf{y}_j\|_2^2 = \sum_{\mathbf{Y}_i \sim \mathbf{Y}_j} \|\mathbf{Y}_i - \mathbf{Y}_j\|_F^2$$

measures the similarity of neighborhood patches in the graph, where we have used the
notation $Y_i \sim Y_j$ if $W_{i,j} = 1$. Here the trace $\text{Tr}(A)$ of a quadratic matrix $A$ denotes the sum of its diagonal entries. For each $j$, the vector $Dx_j$ is assumed to be a good approximation $y_j$. Since the (fixed) transform matrix $D$ induces a linear mapping, we can suppose that the vectors $x_j$, $j = 1, \ldots, m$ possess a similar topological structure as $y_j$, $j = 1, \ldots, m$, 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

$$\text{Tr}(XLX^T) = \sum_{i,j=1}^{m} W_{i,j} \|x_i - x_j\|_2^2 = \sum_{Y_i \sim Y_j} \|x_i - x_j\|_2^2$$

and obtain the new minimization problem

$$\min_{X \in \mathbb{R}^{k \times m}} \frac{1}{2} \|Y - DX\|_F^2 + \frac{\alpha}{2} \text{Tr}(XLX^T) + \lambda \|X\|_1$$

for $\alpha \geq 0$, where the Laplacian matrix $L$ and the learned dictionary $D$ only depend on the training data $Y$. Note that this model is simpler than the model considered in Yankelevsky and Elad (2016), since we have not incorporated the graph Laplacian into the dictionary learning process.

We remark that the weight matrix $W$ of the graph $G$ can also be defined differently, as e.g. by

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

using the Gaussian kernel and some parameter $h$, or alternatively by employing a thresholded kernel. Note, however, that the weight matrix $W$ and hence the Laplacian matrix $L$ is already determined by the vectorized training patches since the Frobenius norm does not exploit any two-dimensional structures of the training patches $Y_1, \ldots, Y_m$ that cannot be observed after vectorization.

Therefore these two-dimensional features have to be incorporated already into the dictionary by the dictionary learning method.
DENOISING METHOD

The regularization model in (4) can now be employed to obtain a denoising method and at the same time a sparse data representation of the given seismic data $Y$ within the dictionary $D$. The procedure has two stages. First we derive a dictionary $D$ that is learned from the given data, see Section METHODS FOR DICTIONARY LEARNING. Then we fix $D$ and solve the minimization problem with regard to $X$ using Algorithm 2, see Section METHODS TO SOLVE THE MINIMIZATION PROBLEM. For the special denoising application we employ the given noisy seismic data themselves to acquire the set of training patches, i.e., we take patches of the noisy data to obtain the set of training patches $\{Y_1, \ldots, Y_m\}$. The new denoising scheme consists of the following steps that we present in Algorithm 1.

Algorithm 1: Denoising algorithm based on dictionary learning and graph regularization

Input

Noisy training data $Y = [y_1, \ldots, y_m]$

Number of iterations

Parameters $K$, $\alpha$ and $\lambda$

Algorithm

1: Set $Y_D := Y$.

Loop steps 2-5 until the given number of iterations is achieved:

2: Compute the Laplacian matrix $L$ for the given training set $Y_D$.

3: Determine the dictionary $D$ by a dictionary learning algorithm based on $Y_D$.

4: Solve the minimization problem $\min_{X \in \mathbb{R}^{k \times m}} \frac{1}{2} \|Y - DX\|_F^2 + \frac{\alpha}{2} \text{Tr}(XLX^T) + \lambda \|X\|_1$.

5: Reconstruct the data $Y_D := DX$.

Output

Denoised data $Y_D$

In the next sections, we describe the two essential steps 3 and 4 of the algorithm in more
For step 2 to compute the Laplacian matrix $L := (L_{i,j})_{i,j=1}^m$, we can simply proceed as follows. First we compute the symmetric matrix of all distances $\|Y_i - Y_j\|_F^2$. Then, in each row $i = 1, \ldots, m$, we order the nonzero values by size and collect the indices $j$ corresponding to the $K$ smallest distances in the index set $I_i$. For $i, j \in \{1, \ldots, m\}$ and $i \neq j$ we put

$$L_{i,j} := \begin{cases} -1 & \text{for } j \in I_i \text{ or } i \in I_j, \\ 0 & \text{otherwise.} \end{cases}$$

Finally, we set $L_{i,i} = \sum_{j=1}^m |L_{i,j}|$.

**METHODS FOR DICTIONARY LEARNING**

We will propose here two dictionary learning methods to perform step 3 of Algorithm 1. Both are based on a special partition tree structure. We construct the dictionaries in two steps. First we compute a special tree structure to partition the set of our training patches. Then, in a second step, we compute the dictionary based on the obtained subset partitions in the tree. The first step, the tree construction, will be different for our two proposed methods, while the method to determine the dictionary from the tree structure will be the same.

The two methods use the set of training patches $Y_1, \ldots, Y_m \in \mathbb{R}^{n \times n}$ and we construct also the dictionary elements $D_\ell, \ell = 1, \ldots, k$ in the form of patches, such that $d_\ell = \text{vec} D_\ell$ are the columns (atoms) of $D$. The dictionary elements $D_\ell$ will be linear combinations of low-rank approximations of suitable center matrices, which are averages of subsets of $Y_j$ with high similarity. This approach enables us to incorporate also 2D-features into the dictionary that cannot be simply found by employing only the vectorized training patches.

**First dictionary construction (FDC).** Motivated by the ideas in Zeng et al. (2015) we employ for the first dictionary learning method a top-bottom two-dimensional subspace
partition (TTSP) as follows.

**Step 1: Construction of the partition tree.** For the given training set \( Y_1, \ldots, Y_m \in \mathbb{R}^{n \times n} \) 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

\[
C_L := \frac{1}{m} \sum_{i=1}^{m} (Y_i - C)(Y_i - C)^T, \quad C_R := \frac{1}{m} \sum_{i=1}^{m} (Y_i - C)^T(Y_i - C).
\] (5)

Observe that \( C_L \) and \( C_R \) possess the same eigenvalues. Now, the normalized eigenvectors \( u \) and \( v \) corresponding to the maximal eigenvalue of \( C_L \) and \( C_R \) are computed, i.e.,

\[
u := \operatorname{argmax}_{\|x\|_2=1} x^T C_L x, \quad v := \operatorname{argmax}_{\|x\|_2=1} x^T C_R x,
\]

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

\[
s_i := u^T Y_i v, \quad i = 1, \ldots, m.
\]

These numbers give us a measure, how strong each single patch is correlated with the structure found and will be used to obtain a partition of the set of all patches \( \{Y_1, \ldots, Y_m\} \) into two partial sets. For this purpose, we order these numbers by size,

\[s_{\ell_1} \leq s_{\ell_2} \leq \ldots \leq s_{\ell_m},\]

and compute

\[
\kappa := \operatorname{argmin}_{1 \leq \kappa \leq m-1} \left[ \sum_{r=1}^{\kappa} \left( s_{\ell_r} - \frac{1}{\kappa} \sum_{\nu=1}^{\kappa} s_{\ell_{\nu}} \right)^2 + \sum_{r=\kappa+1}^{m} \left( s_{\ell_r} - \frac{1}{m-\kappa} \sum_{\nu=\kappa+1}^{m} s_{\ell_{\nu}} \right)^2 \right].
\] (6)

Using \( \kappa \), the partition \( \{Y_{\ell_1}, \ldots, Y_{\ell_\kappa}\} \cup \{Y_{\ell_{\kappa+1}}, \ldots, Y_{\ell_m}\} \) is derived. This is the clustering \( k \)-means method for the special case \( k = 2 \), that can be simply solved exactly for the set of numbers \( \{s_{\ell_1}, \ldots, s_{\ell_m}\} \) by evaluating the term in (6) for each \( \kappa \).

Having found this first partition, we can proceed to partition the two obtained subsets further, using the same scheme. This procedure yields a binary tree. The root node of the
tree is associated with the full set of training patches \( \{ Y_1, \ldots, Y_m \} \) and the two children nodes are associated with the subsets \( \{ Y_{\ell_1}, \ldots, Y_{\ell_k} \} \) and \( \{ Y_{\ell_{k+1}}, \ldots, Y_{\ell_m} \} \). We introduce the corresponding subsets of indices \( \Lambda_1 := \{ 1, \ldots, m \} \) associated to the root node and \( \Lambda_2 := \{ \ell_1, \ldots, \ell_k \} \subset \Lambda_1 \), \( \Lambda_3 := \{ \ell_{k+1}, \ldots, \ell_m \} \subset \Lambda_1 \) associated to the two children nodes. Applying the partition to the subsets of image patches we proceed numbering the index sets at the nodes of the tree by going through each layer from left to right. We stop the further partition of a subset, if it contains less than a predefined number of elements. In Appendix B we give an extensive example of this construction and present the tree obtained by this procedure.

While this partitioning procedure is equivalent to the first part of the TTSP algorithm proposed in Zeng et al. (2015), we compose now the data-dependent dictionary differently from the idea in Zeng et al. (2015) as follows.

**Step 2: Determine the dictionary from the partition tree.** Each node in the tree is now associated with a subset of training patches \( \{ Y_j \}_{j \in \Lambda_k} \), where \( \Lambda_k \subset \{ 1, \ldots, m \} \) denotes the subset of indices of these patches. Now, for each node of the tree, i.e., for each index set \( \Lambda_k \), we compute the mean (center) matrix

\[
C_k := \frac{1}{|\Lambda_k|} \sum_{i \in \Lambda_k} Y_i
\]

and the normalized eigenvectors to the maximal eigenvalue of \( C_k C_k^T \) and \( C_k^T C_k \), i.e.,

\[
u_k := \arg \max_{\|x\|_2 = 1} C_k C_k^T x, \quad v_k := \arg \max_{\|x\|_2 = 1} C_k^T C_k x.
\]

If \( \lambda_k \) denotes the maximal singular value of \( C_k \) then we have \( C_k C_k^T u_k = \lambda_k^2 u_k \), \( C_k^T C_k v_k = \lambda_k^2 v_k \). Thus, \( \lambda_k u_k v_k^T \) 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 nodes with index sets \( \Lambda_{2k} \) and \( \Lambda_{2k+1} \) to the same parent node with center matrices \( C_{2k} \) and \( C_{2k+1} \)
we set
\[
\tilde{D}_k := \lambda_{2k} u_{2k} v_{2k}^T - \lambda_{2k+1} u_{2k+1} v_{2k+1}^T,
\]
\[
D_k := \frac{\tilde{D}_k}{\|\tilde{D}_k\|_F},
\]
thereby capturing the difference of main structures of \(C_{2k}\) and \(C_{2k+1}\). Note that this procedure is essentially different from the construction in Zeng et al. (2015).

Remarks. 1. We remark that instead of approximating the centers \(C_k\) by a rank-1 matrix, we can also use a more exact approximation with matrices of higher rank using the singular value decomposition. For sparse representation purposes and if the patches do not contain noise, one may even use the centers \(C_k\) directly instead of their approximations of smaller rank.

2. The procedure avoids the problem of having dictionary elements being very similar. Strong similarity of dictionary elements especially occurs if only the center matrices of the subsets in the nodes of the tree, or a small rank approximation of these center matrices are employed, see Zeng et al. (2015).

3. Our dictionary construction can be understood as a generalized wavelet approach, where the dictionary elements for \(k > 1\) are "wavelet elements" while \(D_1\) is the low-pass element.

Second dictionary construction (SDC). We want to propose a second method for dictionary learning.

Step 1: Construction of the partition tree. This time, we construct the partition tree in a different way, using the similarity of training patches as we did already for the graph regularization. Here, we use the weights
\[
w_{i,j} := \|Y_i - Y_j\|_F^2
\]
(7)
to measure the similarity between \(Y_i\) and \(Y_j\).

First we order the patches such that \(Y_1\) has the smallest Frobenius norm. Now, we compute the weights \(w_{1,j}\) for \(j = 1, \ldots, m\) according to (7) and order them by size,
\[
w_{1,\ell_1} \leq w_{1,\ell_2} \leq \ldots w_{1,\ell_m},
\]
where \( \ell_1 = 1 \) since \( w_{1,1} = 0 \) is the smallest weight. Now, similarly as before in the first construction, we divide the set of training patches into two subsets \( \{ Y_{\ell_1}, \ldots, Y_{\ell_k} \} \) and \( \{ Y_{\ell_{k+1}}, \ldots, Y_{\ell_m} \} \) using

\[
\hat{\kappa} := \arg\min_{1 \leq \kappa \leq m-1} \left[ \sum_{r=1}^{\kappa} \left( w_{1,\ell_r} - \frac{1}{\kappa} \sum_{\nu=1}^{\kappa} w_{1,\ell_\nu} \right)^2 + \sum_{r=\kappa+1}^{m} \left( w_{1,\ell_r} - \frac{1}{m-\kappa} \sum_{\nu=\kappa+1}^{m} w_{1,\ell_\nu} \right)^2 \right].
\]

We proceed to partition the obtained subsets using the same procedure and obtain a binary tree, where each node is associated with a subset of training patches.

**Step 2: Determine the dictionary from the partition tree.** We proceed analogously as for the first construction and apply the rank-1 approximation of the mean \( \mathbf{C} = \mathbf{C}_1 \) and the first dictionary element as well as the normalized differences of the rank-1 approximations of the centers of each pair of two children in the partition tree as further dictionary elements.

We consider a toy example to illustrate the two dictionary learning methods in the appendix B (see also Figure 2).

Let us shortly summarize the computational cost for the two dictionary learning methods FDC and SDC. For FDC we have to compute for the first partition the matrices \( \mathbf{C}_L \) and \( \mathbf{C}_R \) and need \( \mathcal{O}(mn^3) \) operations. The computation of \( \hat{\kappa} \) is most expensive with \( \mathcal{O}(m^2) \) operations. All further partition steps are cheaper since the number of patches decays. We obtain overall computational cost of at most \( \mathcal{O}(km(m+n^3)) \) for the first step of FDC, where \( \mathcal{O}(k) \) bounds the number of partitions in the tree, and \( k \) is the number of wanted dictionary elements. For the SDC method, we need to compute the weights \( w_{i,j} \) in (7) with an effort of \( \mathcal{O}(m^2n^2) \) first. The remaining effort is governed by computing \( \hat{\kappa} \) with \( \mathcal{O}(m^2) \) operations in the first partition and with less effort in the further partitions. Since the weights \( w_{i,j} \) have to be computed only once, we get an overall computational cost of at most \( \mathcal{O}(m^2(n^2 + k)) \).

The second step, namely the determination of the dictionary from the tree is the same for both methods and needs at most \( \mathcal{O}(n^3k) \) operations. For an over-complete dictionary we can assume that \( k > n^2 \) and the number \( m \) of training patches is much larger than \( k \). Therefore, both algorithms have an effort governed by \( \mathcal{O}(m^2k) \). The second method gets
a bit cheaper than the first, if we assume that the weights $w_{i,j}$ in (7) are already known, since they are also needed for computing the graph regularization term.

**METHODS TO SOLVE THE MINIMIZATION PROBLEM**

In the fourth step of the proposed denoising Algorithm 1 we have to solve the minimization problem

$$\min_{X \in \mathbb{R}^{k \times m}} \left( \frac{1}{2}\|Y - DX\|^2_F + \frac{\alpha}{2} \text{Tr}(XLX^T) + \lambda\|X\|_1 \right),$$

for given (noisy) training data $Y$ and the given dictionary $D = [d_1 \ldots d_k]$, where $d_j = \text{vec}D_j \in \mathbb{R}^N$ are the dictionary elements constructed in the last section.

We suggest here to solve the problem using the split Bregman iteration see e.g. Goldstein and Osher (2009); Plonka and Ma (2011) which is in the considered case equivalent to the Alternating Direction Method of Multipliers (ADMM), see Yankelevsky and Elad (2016). For other approaches we also refer e.g. to Chambolle and Pock (2011) or to Lee et al. (2007). Since we have not found a suitable presentation of a computational method for the particular problem (8) in the literature, we present the mathematical details to solve this optimization problem in Appendix A. These considerations lead to the following Algorithm 2 to perform step 4 of Algorithm 1.
Algorithm 2 : Solving the minimization problem

Input

Noisy training data \( \mathbf{Y} = [\mathbf{y}_1, \ldots, \mathbf{y}_m] \in \mathbb{R}^{N \times m} \)

Laplacian matrix \( \mathbf{L} \in \mathbb{R}^{m \times m} \)

Learned dictionary \( \mathbf{D} \in \mathbb{R}^{N \times k} \)

\( \mathbf{X}^0 = \mathbf{Z}^0 = \mathbf{B}^0 = \mathbf{0}_{k \times m} \)

Parameters \( \lambda, \mu, \alpha > 0 \)

Number of iterations \( \ell \)

Algorithm

Iterate until the given number of iterations is achieved:

1: Compute \( \mathbf{X}^{\ell+1} \) as the solution of
\[
(D^T \mathbf{D} + \mu \mathbf{I}) \mathbf{X} + \alpha \mathbf{X} \mathbf{L} = D^T \mathbf{Y} + \mu (\mathbf{Z}^\ell - \mathbf{B}^\ell).
\]

2: Compute \( \mathbf{Z}^{\ell+1} \) componentwisely by employing soft shrinkage
\[
z_{i,j}^{\ell+1} = \mathcal{T}_{\lambda/\mu}(x_{i,j}^{\ell+1} + B_{i,j}^\ell) := \begin{cases} 
  x_{i,j}^{\ell+1} + B_{i,j}^\ell - \frac{\lambda}{\mu} & \text{for } (x_{i,j}^{\ell+1} + B_{i,j}^\ell) \geq \frac{\lambda}{\mu}, \\
  x_{i,j}^{\ell+1} + B_{i,j}^\ell + \frac{\lambda}{\mu} & \text{for } (x_{i,j}^{\ell+1} + B_{i,j}^\ell) \leq -\frac{\lambda}{\mu}, \\
  0 & \text{otherwise},
\end{cases}
\]

for \( i = 1, \ldots, k, \ j = 1, \ldots, m. \)

3: Update \( \mathbf{B}^{\ell+1} = \mathbf{B}^\ell - \mathbf{Z}^{\ell+1} + \mathbf{X}^{\ell+1}. \)

Output \( \mathbf{X} \)

Note that the matrix equation
\[
(D^T \mathbf{D} + \mu \mathbf{I}) \mathbf{X} + \alpha \mathbf{X} \mathbf{L} = D^T \mathbf{Y} + \mu (\mathbf{Z}^\ell - \mathbf{B}^\ell)
\] (9)

that has to be solved with regard to the unknown \((k \times k)\) matrix \( \mathbf{X} \) in the first step of Algorithm 2, has a very special structure. We shortly show how it can be inverted into a usual linear equation system. Let the Kronecker product of two matrices \( \mathbf{A} = (a_{ij})_{i,j=1}^k \in \mathbb{R}^{k \times k} \) and \( \mathbf{B} \in \mathbb{R}^{k \times k} \) be defined as

\[
\mathbf{A} \otimes \mathbf{B} := \begin{bmatrix} 
  a_{11} \mathbf{B} & \ldots & a_{1k} \mathbf{B} \\
  \vdots & \ddots & \vdots \\
  a_{k1} \mathbf{B} & \ldots & a_{kk} \mathbf{B}
\end{bmatrix} \in \mathbb{R}^{k^2 \times k^2},
\]
and let \( \text{vec} X = x \in \mathbb{R}^{k^2} \) denote the vectorization of the matrix \( X \) formed by stacking the columns of \( X \) into a single vector. We can use now the property
\[
\text{vec}(AXB) = (B^T \otimes A) \text{vec} X = (B^T \otimes A) x,
\]
see e.g. Horn and Johnson (1991), Lemma 4.3.1, and obtain
\[
\begin{align*}
\text{vec} \left( (D^T D + \mu I)X \right) &= (I_k \otimes (D^T D + \mu I)) x, \\
\text{vec} (\alpha XL) &= \alpha (L \otimes I_k) x,
\end{align*}
\]
where \( I_k \) denotes the identity matrix of size \( k \times k \). Thus (9) is equivalent with the linear equation system
\[
\left[ (I_k \otimes (D^T D + \mu I)) + \alpha (L \otimes I_k) \right] x = \text{vec} \left( D^T Y + \mu (Z^\ell - B^\ell) \right).
\]
For efficient solutions of equations of the form (9) we refer to Bartels and Stewart (1972); Bhatia and Rosenthal (1997), see also Appendix A.

**EXPERIMENTS**

In this section, we verify the seismic data denoising performance of the proposed FDC-graph and SDC-graph method in Algorithm 1 and compare their performance to results of the regularization method without the graph regularization term, i.e., with \( \alpha = 0 \). We highlight the advantages and disadvantages of using the graph regularization term on a field data set with known amounts of random noise added. Comparisons to a traditional FX (frequency domain in the time direction and spatial domain in trace direction) deconvolution (FX-Decon) algorithm (Canales., 1984; Gulunay, 1986) and to state-of-the-art curvelet denoising by thresholding (Hennenfent and Herrmann, 2006) are also provided. A Matlab implementation for the FX-Decon method is available from the author M.D. Sacchi, see [http://www-geo.phys.ualberta.ca/saig/SeismicLab](http://www-geo.phys.ualberta.ca/saig/SeismicLab). The Matlab code for curvelet denoising can be found in the curvelet toolbox, see [http://www.curvelet.org](http://www.curvelet.org).

Determining an objective data quality metric plays an important role in denoising applications. The peak-signal-to-noise ratio (PSNR) (see e.g. Zoran and Weiss, 2011) in dB
is given by

\[ PSNR = 20 \log_{10} \left( \frac{\max(Y_C)}{\text{std2}(Y_C - Y_D)} \right), \]

where \( Y_D \) presents the denoised data and \( Y_C \) the clear data. The function \( \text{std2} \) computes the standard deviation of \( Y_C - Y_D \). Computation time (in seconds), and the recovered error (i.e., error = \( \frac{||Y_C - Y_D||_2^2}{||Y_C||_F^2} \)) are used as the performance measurements in our quantitative evaluation. The parameter \( K \) for building the graph Laplacian, where we need to fix the \( K \) nearest neighbors, is always taken to be \( K = 6 \).

To qualitatively evaluate our algorithm’s effectiveness on seismic data, we compare the following six algorithms for denoising of seismic data in Figure 4(b), where the noise standard deviation \( \sigma \) is 25: FX-Decon, Curvelet algorithm, FDC-OMP, SDC-OMP, FDC-Graph, and SDC-Graph. For all methods we tried to choose the parameters to achieve optimal denoising results. Here FDC-OMP and SDC-OMP denote the denoising methods that we obtain by solving the optimization problem (1) by orthogonal matching pursuit with a dictionary \( D \) learned by FDC or SDC. Note that the minimization problem (1) even uses the better semi-norm \( ||X||_0 \) instead of \( ||X||_1 \), but does not include the graph-regularization term \( \frac{\alpha}{2} \text{Tr}(XX^T) \). Similarly, FDC-Graph and SDC-Graph denote the methods obtained from Algorithm 1, using the first (FDC) or the second (SDC) dictionary learning approach. To see the impact of the graph regularization term, we have taken the FDC-OMP and the SDC-OMP here instead of applying Algorithm 1 with \( \alpha = 0 \), since these methods work even slightly better. The FX-Decon algorithm has been applied to a window of size 64 × 64 samples overlapping on 8 samples in both the time and spatial dimensions with filters of length 6. The curvelet denoising algorithm decomposes the noisy data into 5 different scales and different directions, and 91% percent of the small curvelet coefficients are removed by thresholding. For the seismic data in Figure 4(b), the FDC-Graph and SDC-Graph algorithms use a set of 961 patches that constitutes the training set, where the size of patches is 16 × 16. The regularization parameters taken in Algorithm 2 to solve in (11) (see Appendix A) have been empirically chosen to be \( \alpha = 1.6; \lambda = 0.4; \mu = 0.05 \) for FDC-Graph and \( \alpha = 1.6; \lambda = 0.5; \mu = 0.04 \) for SDC-Graph. For the dictionary learning part, the number
of patches contained in each node determines the number of layers of the tree. We have set the minimal number of patches in a subset corresponding to one node to 6 for the FDC-Graph algorithm and to 16 for the SDC-Graph algorithm. The dictionary learned by the FDC and SDC approach are shown in Figure 5, where the training patches are taken from the noisy data in Figure 4(b). The main difference between FDC-OMP/SDC-OMP and FDC-Graph/SDC-Graph is that by graph regularization the geometric similarity of data patches is employed. We observe from the close-up window in Figure 6 that the results using the graph regularization methods maintain the even continuous and weak features of the original data. Figure 7 shows the separated noise, i.e., the difference between denoising results and field data one in Figure 4(a). From the single trace comparison in Figure 8 we can see that all methods achieve strong amplitudes close to the clear data trace. But for weak amplitudes, the denoising results using the Graph regularization methods are much better compared to the methods without using Graph regularization. The statistical results with various noise standard deviation are shown in Figure 9. In general, we observe that the graph regularization term is helpful for capturing the features of seismic data.

Finally, we have applied FX-Decon, curvelet denoising, and the proposed FDC-Graph, and SDC-Graph algorithms to denoising on field data two (the size of data is $256 \times 256$) in Figure 4(c). FX-Decon was applied to windows of size $50 \times 50$ samples overlapping on 16 samples in both dimensions and with filters of length 4. The Curvelet denoising algorithm decomposes the noisy data into 5 scales and 98% percent of the small curvelet coefficients are removed by thresholding. For the FDC-Graph and SDC-Graph algorithms we employ Algorithm 1 and Algorithm 2, where the parameters are empirically chosen to be $\alpha = 1.2$; $\lambda = 0.5$; $\mu = 0.08$ for FDC-Graph and $\alpha = 1.2$; $\lambda = 0.5$; $\mu = 0.06$ for SDC-Graph. Note that the field data two is an actual field data where noise was not artificially added; the noise seen in the data is the original noise captured along with the data acquisition. The denoising results are presented in Figure 10 and separated noise of field data two is given in Figure 11. We can see that the seismic event in curvelet denoising result is smoother than for the other methods in the Figure 10. In Figure 11, the separated noise obtained by curvelet denoising
contains still information of signal while the FDC-Graph, and SDC-Graph algorithms work significantly better for noise separation. From the magnified windows in Figures 10 and 11 we can see that the FDC-Graph and SDC-Graph methods better recover the weak features found in the original data compared to the other methods.

The computational costs of the new FDC-Graph and SDC-Graph algorithms are similar as for the same test since we have the same size of field data and have used the same size of training patches $16 \times 16$ as well as the same minimal number of patches in subsets in the dictionary construction as for the first test, see Figure 9(e).

**CONCLUSION**

In this paper we proposed a regularization approach for seismic data processing using graph regularization and dictionary learning methods. Experimental results achieved on field seismic data show that the proposed method provides improved denoising results compared to methods that only employ a learned dictionary. However, the new method requires the solution of a minimization problem at each iteration step and is therefore more time consuming than similar methods employing OMP but no graph regularization. Also, the recovery for stronger noise levels is not satisfactory up to now. Future work will focus on improving the computational speed of the algorithm and on efficient choice of optimal parameters. Furthermore, we will study this approach for other geophysical inversion problems such as migration, imaging, and full waveform inversion.

**ACKNOWLEDGEMENTS**

This work is supported by National Nature Science Foundation of China (grant number: NSFC 91330108, 41374121, 61327013), and the Fundamental Research Funds for the Central Universities (grant number: HIT.PIRS.A201501).
APPENDIX A

We introduce the mathematical details of the split Bregman algorithm to derive Algorithm 2, which is used to solve the fourth step of proposed denoising Algorithm 1. Let’s recall the problem in the fourth step of Algorithm 1,

\[
\min_{X \in \mathbb{R}^{k \times m}} \left( \frac{1}{2} \|Y - DX\|_F^2 + \frac{\alpha}{2} \text{Tr}(XLX^T) + \lambda \|X\|_1 \right). \tag{10}
\]

First, we introduce a further variable \(Z \in \mathbb{R}^{k \times m}\) and rewrite the problem as follows,

\[
\min_{X, Z \in \mathbb{R}^{k \times m}} \frac{1}{2} \|Y - DX\|_F^2 + \frac{\alpha}{2} \text{Tr}(XLX^T) + \frac{\mu}{2} \|X - Z\|_F^2 + \lambda \|Z\|_1 \tag{11}
\]

with some fixed parameter \(\mu > 0\). Now, let

\[
E(X, Z) := \lambda \|Z\|_1 + \frac{1}{2} \|Y - DX\|_F^2 + \frac{\alpha}{2} \text{Tr}(XLX^T),
\]

such that (11) is of the form

\[
\min_{X, Z \in \mathbb{R}^{k \times m}} E(X, Z) + \frac{\mu}{2} \|X - Z\|_F^2.
\]

We introduce the so-called Bregman distance

\[
D_E((X, Z), (X^\ell, Z^\ell)) := E(X, Z) - E(X^\ell, Z^\ell) - \langle P^\ell, X - X^\ell \rangle - \langle Q^\ell, Z - Z^\ell \rangle,
\]

where \((P^\ell, Q^\ell)\) is a subgradient of \(E\) at \((X^\ell, Z^\ell)\), i.e., \(P^\ell \in \partial_X E(X^\ell, Z^\ell)\) and \(Q^\ell \in \partial_Z E(X^\ell, Z^\ell)\), where \(\partial_X E(X^\ell, Z^\ell)\) and \(\partial_Z E(X^\ell, Z^\ell)\) denote the subdifferentials of \(E\) at \((X^\ell, Z^\ell)\) with respect to \(X\) and \(Z\). Now, to solve (11), we replace \(E(X, Z)\) by the Bregman distance and consider the iteration

\[
(X^{\ell+1}, Z^{\ell+1}) = \text{argmin}_{X, Z \in \mathbb{R}^{k \times m}} \left\{ D_E((X, Z), (X^\ell, Z^\ell)) + \frac{\mu}{2} \|X - Z\|_F^2 \right\}
\]

\[
= \text{argmin}_{X, Z \in \mathbb{R}^{k \times m}} \left\{ E(X, Z) - \langle P^\ell, X \rangle - \langle Q^\ell, Z \rangle + \frac{\mu}{2} \|X - Z\|_F^2 \right\}. \tag{12}
\]

This iteration is sensible, since the two additional terms \(\langle P^\ell, X \rangle\) and \(\langle Q^\ell, Z \rangle\) vanish for the desired solution \((X, Z)\) of (11). From (12) we derive the necessary conditions

\[
0 \in \partial_X E(X^{\ell+1}, Z^{\ell+1}) - P^\ell + \mu(X^{\ell+1} - Z^{\ell+1}),
\]

\[
0 \in \partial_Z E(X^{\ell+1}, Z^{\ell+1}) - Q^\ell - \mu(X^{\ell+1} - Z^{\ell+1}).
\]
Since \((P^{\ell+1}, Q^{\ell+1}) \in \partial E(X^{\ell+1}, Z^{\ell+1})\) we obtain the following recursions from these conditions,

\[
P^{\ell+1} = P^\ell - \mu(X^{\ell+1} - Z^{\ell+1}), \quad Q^{\ell+1} = Q^\ell + \mu(X^{\ell+1} - Z^{\ell+1}),
\]
and particularly, \(P^{\ell+1} + Q^{\ell+1} = P^\ell + Q^\ell\) for all \(\ell\). Now, introducing \(B^\ell := \frac{1}{\mu} Q^\ell\), the second equation in (13) implies the recursion

\[
B^{\ell+1} = B^\ell - Z^{\ell+1} + X^{\ell+1}.
\]

Moreover, by

\[
\frac{\mu}{2} \| X - Z + B^\ell \|_F^2 = \frac{\mu}{2} \| X - Z \|_F^2 + \mu \langle X - Z, B^\ell \rangle + \frac{\mu}{2} \| B^\ell \|_F^2
\]
we can rewrite (12) as

\[
(X^{\ell+1}, Z^{\ell+1}) = \arg\min_{X,Z \in \mathbb{R}^{k \times m}} \left\{ E(X,Z) + \frac{\mu}{2} \| X - Z + B^\ell \|_F^2 - \mu \langle X - Z, B^\ell \rangle - \langle P^\ell, X \rangle - \langle Q^\ell, Z \rangle \right\}
\]

where we have used that \(-\mu \langle X - Z, B^\ell \rangle = -\langle Q^\ell, X \rangle + \langle Q^\ell, Z \rangle\). Taking the initial matrices \(B^0 = P^0 = Q^0 = 0\), it follows that \(P^\ell + Q^\ell = 0\) for all \(\ell\), and we arrive at the iteration

\[
(X^{\ell+1}, Z^{\ell+1}) = \arg\min_{X,Z \in \mathbb{R}^{k \times m}} \left\{ E(X,Z) + \frac{\mu}{2} \| X - Z + B^\ell \|_F^2 \right\},
\]

\[
B^{\ell+1} = B^\ell - Z^{\ell+1} + X^{\ell+1}.
\]

Applying alternating minimization, this leads to the following iteration scheme,

\[
X^{\ell+1} = \arg\min_{X \in \mathbb{R}^{k \times m}} \left\{ \frac{1}{2} \|DX - Y\|_F^2 + \frac{\alpha}{2} \text{Tr}(XLX^T) + \frac{\mu}{2} \| X - Z^{\ell} + B^\ell \|_F^2 \right\},
\]

\[
Z^{\ell+1} = \arg\min_{Z \in \mathbb{R}^{k \times m}} \left\{ \lambda \|Z\|_1 + \frac{\mu}{2} \| X^{\ell+1} - Z + B^\ell \|_F^2 \right\},
\]

\[
B^{\ell+1} = B^\ell - Z^{\ell+1} + X^{\ell+1}.
\]

The functional in the first equation of (14) is differentiable, and we obtain the necessary condition

\[
D^T(DX - Y) + \alpha XL + \mu(X - Z^\ell + B^\ell) = 0,
\]
\[(D^T D + \mu I)X + \alpha XL = D^T Y + \mu (Z^\ell - B^\ell).\]

This equation is uniquely solvable if the eigenvalues \(\alpha_1, \ldots, \alpha_k\) of \((D^T D + \mu I)\) and the eigenvalues \(\beta_1, \ldots, \beta_k\) of \(\alpha L\) satisfy

\[\alpha_i + \beta_j \neq 0 \quad \forall i = 1, \ldots, k, j = 1, \ldots, m,\]

see Bartels and Stewart (1972); Bhatia and Rosenthal (1997). This is true in our case since \((D^T D + \mu I)\) is positive definite for \(\mu > 0\) and \(L\) is positive semidefinite.

The second subproblem in (14) can be simply solved by component-wise shrinkage. For each single component \(z_{i,j}^{\ell+1}\) of \(Z^{\ell+1} = (z_{i,j}^{\ell+1})_{i=1,j=1}^{k,m}\) we have

\[z_{i,j}^{\ell+1} = \arg\min_{z \in \mathbb{R}} \{\lambda |z| + \frac{\mu}{2} |z - x_{i,j}^{\ell+1} - B_{i,j}^\ell|^2\},\]

i.e.,

\[0 \in \lambda \frac{z_{i,j}^{\ell+1}}{|z_{i,j}^{\ell+1}|} + \mu \left(z_{i,j}^{\ell+1} - x_{i,j}^{\ell+1} - B_{i,j}^\ell\right),\]

where \(\frac{z}{|z|}\) denotes the set \([-1, 1]\) if \(z = 0\). Hence, we find the solution by soft shrinkage,

\[z_{i,j}^{\ell+1} = T_{\lambda/\mu}(x_{i,j}^{\ell+1} + B_{i,j}^\ell) := \begin{cases} x_{i,j}^{\ell+1} + B_{i,j}^\ell - \frac{\lambda}{\mu} & \text{for } (x_{i,j}^{\ell+1} + B_{i,j}^\ell) \geq -\frac{\lambda}{\mu}, \\ x_{i,j}^{\ell+1} + B_{i,j}^\ell + \frac{\lambda}{\mu} & \text{for } (x_{i,j}^{\ell+1} + B_{i,j}^\ell) \leq -\frac{\lambda}{\mu}, \\ 0 & \text{otherwise.} \end{cases}\]

**APPENDIX B**

In our paper, we have proposed two dictionary learning algorithms, called FDC and SDC. In order to illustrate the dictionary learning step more clearly, we will give a small toy example, where the training set contains the following 10 training patches of size 3 \(\times\) 3,

\[
\begin{align*}
Y_1 &= \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}, & Y_2 &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}, & Y_3 &= \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}, & Y_4 &= \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}.
\end{align*}
\]
We construct the partition trees using the two algorithms in section METHODS FOR DICTIONARY LEARNING and show, how the dictionary construction can be directly incorporated into the partition tree construction. The example also shows that the trees obtained and thus the dictionary elements found slightly differ for the two approaches. For the explicit computations we round to 2 digits in matrix entries and to 4 digits for entries of singular values and singular vectors/eigenvectors.

**FDC algorithm**

**First layer:** We start with the training set \( Y_1, Y_2, \ldots, Y_{10} \). We compute the mean (center) matrix

\[
C_1 = \frac{1}{10} \sum_{i=1}^{10} Y_i = \begin{pmatrix} 0.6 & 0.5 & 0.4 \\ 0.2 & 0.7 & 0.8 \\ 0 & 0.1 & 0.6 \end{pmatrix}
\]

and the normalized eigenvectors to the maximal eigenvalue of

\[
C_1 C_1^T = \begin{pmatrix} 0.77 & 0.79 & 0.29 \\ 0.79 & 1.17 & 0.55 \\ 0.29 & 0.55 & 0.37 \end{pmatrix} \quad \text{and} \quad C_1^T C_1 = \begin{pmatrix} 0.40 & 0.44 & 0.40 \\ 0.44 & 0.75 & 0.82 \\ 0.40 & 0.82 & 1.16 \end{pmatrix}.
\]

We obtain

\[
u_1 = [-0.5590, -0.7516, -0.3501]^T, \quad v_1 = [-0.3423, -0.5924, -0.7293]^T,
\]

which are the singular vectors of \( C_1 \) to the singular value \( \lambda_1 = 1.4191 \). Thus we get the
first dictionary element

\[
D_1 = u_1 v_1^T = \begin{pmatrix} 0.19 & 0.33 & 0.41 \\ 0.26 & 0.45 & 0.55 \\ 0.12 & 0.21 & 0.26 \end{pmatrix},
\]

and \(\lambda_1 D_1\) represents the optimal rank-1 approximation of \(C_1\).

**Second layer:** The training set is now divided into two partial sets according to the description for FDC tree partitioning. This can be done as follows:

1. We calculate the two non-symmetric covariance matrices \(C_L\) and \(C_R\) in (5),

\[
C_L = \begin{pmatrix} 0.73 & -0.19 & -0.90 \\ -0.19 & 0.53 & 0.05 \\ -0.09 & 0.05 & 0.33 \end{pmatrix}, \quad C_R = \begin{pmatrix} 0.40 & 0.16 & -0.10 \\ 0.16 & 0.55 & 0.08 \\ -0.10 & 0.08 & 0.64 \end{pmatrix}.
\]

2. We compute the normalized eigenvectors corresponding to the maximal eigenvalue of \(C_L\) and \(C_R\), \(u = \begin{bmatrix} -0.8415, 0.5062, 0.1890 \end{bmatrix}^T\), \(v = \begin{bmatrix} -0.590, 0.4508, 0.8907 \end{bmatrix}^T\).

3: We compute the one-dimensional projection representations \(s_i := u^T Y_i v\) of all patches, obtaining

\[
s = [s_1, \ldots, s_{10}]^T = \begin{bmatrix} 0.12, 0.28, 0.37, 0.90, -0.40, 0.15, 0.52, -0.68, -0.46, 0.85 \end{bmatrix}^T.
\]

We order these numbers in \(s\) by size, \(s_{\ell_1} \leq s_{\ell_2} \leq \ldots \leq s_{\ell_{10}}\), and find

\[
s_1 = [s_{\ell_1}, \ldots, s_{\ell_{10}}]^T = [s_8, s_9, s_5, s_1, s_6, s_2, s_3, s_7, s_{10}, s_4]^T
\]

\[
= [-0.68, -0.46, -0.40, 0.12, 0.15, 0.28, 0.37, 0.52, 0.85, 0.90]^T.
\]

4: We partition the patches by computing \(\hat{\kappa}\) in (6). We obtain \(\hat{\kappa} = 3\), and the partition of index sets \(\Lambda_1 = \Lambda_2 \cup \Lambda_3\) with \(\Lambda_2 = \{5, 8, 9\}\), \(\Lambda_3 = \{1, 2, 3, 4, 6, 7, 10\}\) is derived. The center matrices of the obtained two node sets are

\[
C_2 = \frac{1}{3}(Y_5 + Y_8 + Y_9), \quad C_3 = \frac{1}{7}(Y_1 + Y_2 + Y_3 + Y_4 + Y_6 + Y_7 + Y_{10}).
\]
We compute the maximal singular values $\lambda_2 = 1.8259$, $\lambda_3 = 1.3648$ of the center matrices $C_2, C_3$ and the first left and right singular vectors $u_2, v_2$ and $u_3, v_3$ of $C_2, C_3$, respectively.

Then, we obtain the second dictionary element $D_2 = \frac{\tilde{D}_2}{\|\tilde{D}_2\|_F}$ with

$$\tilde{D}_2 = \lambda_2 u_2 v_2^T - \lambda_3 u_3 v_3^T = \begin{pmatrix} 0.55 & 1.48 & 1.83 \\
0.27 & 0.31 & 0.51 \\
0.04 & -0.14 & -0.09 \end{pmatrix}.$$ 

Now, $\tilde{D}_2$ is a rank-2 approximation of $C_2 - C_3$ and therefore represents the difference between main structures of $C_2$ and $C_3$ while the common significant structure is already captured by $D_1$. Indeed, since $\lambda_1 D_1$ approximates $C_1 = \frac{3}{10} C_2 + \frac{7}{10} C_3$, the main structures of $C_2$ and $C_3$ can be well presented by $D_1$ and $D_2$.

**Third layer:** The partition procedure yields now the subsets of indices $\Lambda_4 = \{8\}$, $\Lambda_5 = \{5,9\}$ of $\Lambda_2$, and the subsets $\Lambda_6 = \{3,6\}$, $\Lambda_7 = \{1,2,4,7,10\}$ of $\Lambda_3$, see also Figure 3. We compute the center matrices of the corresponding partial sets of patches,

$$C_4 = Y_8, \quad C_5 = \frac{1}{2}(Y_5 + Y_9), \quad C_6 = \frac{1}{2}(Y_3 + Y_6), \quad C_7 = \frac{1}{5}(Y_1 + Y_2 + Y_4 + Y_7 + Y_{10}).$$

We get the new dictionary element $D_3 = \frac{D_3}{\|D_3\|_F}$ with

$$\tilde{D}_3 = \lambda_4 u_4 v_4^T - \lambda_5 u_5 v_5^T = \begin{pmatrix} -0.72 & -0.21 & -0.05 \\
-0.42 & -0.10 & 0.01 \\
-0.15 & -0.20 & -0.26 \end{pmatrix},$$

where $\lambda_4 = 1.6180$ and $\lambda_5 = 1.9966$ are maximal singular values of the center matrices $C_4$ and $C_5$, and $u_4, v_4$ and $u_5, v_5$ are first singular vectors of $C_4, C_5$. Similarly, $D_4 = \frac{D_4}{\|D_4\|_F}$ with

$$\tilde{D}_4 = \lambda_6 u_6 v_6^T - \lambda_7 u_7 v_7^T = \begin{pmatrix} 0.05 & -0.10 & -0.13 \\
0.44 & 0.17 & 0.07 \\
0.35 & 0.30 & 0.25 \end{pmatrix},$$

where $\lambda_6 = 1.7184$ and $\lambda_7 = 1.2630$ are maximal singular values of center matrices $C_6$ and $C_7$ with corresponding singular vectors $u_6, v_6, u_7, v_7$. Since $\tilde{D}_3$ is now a good approxima-
tion of $C_4 - C_5 = Y_8 - C_5$, we can obtain a sparse approximation of $Y_8$ already using the dictionary elements $D_1$, $D_2$ and $D_3$.

**Fourth layer:** The index subset $\Lambda_5$ is split into $\Lambda_8 = \{5\}$, $\Lambda_9 = \{9\}$, $\Lambda_6$ is split into $\Lambda_{10} = \{3\}$, $\Lambda_{11} = \{6\}$; and $\Lambda_7$ is split into $\Lambda_{12} = \{1, 7\}$ and $\Lambda_{13} = \{2, 4, 10\}$. Then $D_5$, $D_6$ and $D_7$ can be learned from $\{\Lambda_8, \Lambda_9\}$, $\{\Lambda_{10}, \Lambda_{11}\}$, and $\{\Lambda_{12}, \Lambda_{13}\}$.

**Fifth layer:** The index subset $\Lambda_{12}$ is split into $\Lambda_{14} = \{1\}$ and $\Lambda_{15} = \{7\}$, $\Lambda_{13}$ is split into $\Lambda_{16} = \{2\}$ and $\Lambda_{17} = \{4, 10\}$, and we learn the dictionary elements $D_8$ and $D_9$.

**Sixth layer:** Finally, index subset $\Lambda_{17}$ is split into $\Lambda_{18} = \{4\}$ and $\Lambda_{19} = \{10\}$, and we learn the dictionary element $D_{10}$. The complete partition tree is given in Figure 3(a).

**Remark.** Please keep in mind that we have used here always rank-1 approximations of the center matrices since we assume that the training data are noisy in applications. If the training data are exact (as it is the case in this toy example), then we may use the center matrices themselves instead of their rank-1 approximations, getting the dictionary

\[ D_1 = \frac{C_1}{\|C_1\|_F}, \quad D_2 = \frac{C_2 - C_3}{\|C_2 - C_3\|_F}, \quad D_3 = \frac{C_4 - C_5}{\|C_4 - C_5\|_F}, \quad D_4 = \frac{C_6 - C_7}{\|C_6 - C_7\|_F}, \]

etc., and have for example already an exact sparse representation of $Y_8 = C_4$ as a linear combination of $D_1$, $D_2$ and $D_3$. Indeed, we find

\[ Y_8 = C_4 = C_1 + \frac{7}{10} (C_2 - C_3) + \frac{2}{3} (C_4 - C_5). \]

Applying the SDC algorithm, we obtain a different partition tree, see Figure 3(b), where

\[
\begin{align*}
\Lambda_1 & = \{1, \ldots, 10\} \\
\Lambda_2 & = \{1, 2, 3, 4, 5, 7, 10\}, \quad \Lambda_3 = \{6, 8, 9\}, \\
\Lambda_4 & = \{2\}, \quad \Lambda_5 = \{1, 3, 4, 5, 7, 10\}, \quad \Lambda_6 = \{8, 9\}, \quad \Lambda_7 = \{6\} \\
\Lambda_8 & = \{1, 4, 5, 7\}, \quad \Lambda_9 = \{3, 10\}, \quad \Lambda_{10} = \{8\}, \quad \Lambda_{11} = \{9\} \\
\Lambda_{12} & = \{1\}, \quad \Lambda_{13} = \{4, 5, 7\}, \quad \Lambda_{14} = \{10\}, \quad \Lambda_{15} = \{3\}, \\
\Lambda_{16} & = \{4, 5\}, \quad \Lambda_{17} = \{7\}, \\
\Lambda_{18} & = \{4\}, \quad \Lambda_{19} = \{5\}.
\end{align*}
\]
Observe that this approach provides a different partition than the first, also the number of layers and well as the number of dictionary elements is different.

REFERENCES


Bartels, R.H., and G.W. Stewart, 1972, Solution of the matrix equation $AX + XB = C$: Commun. ACM 15, no. 9, 820-826.


Hein, M., and M. Maier, 2006, Manifold denoising: Advances in Neural Information Pro-
cessing Systems, 19, 561-568.


Figure Captions:

Figure 1: Example for a weighted and undirected graph, where with weight 1 for \( K \)-nearest neighbors.

Figure 2: The process for dictionary learning by FDC and SDC algorithm, where \( \lambda_i \) denotes the maximal singular value of center matrix \( C_i \) at each node, \( u_i \) and \( v_i \) are the first vectors in the singular value decomposition of \( C_i \).

Figure 3: Trees for (a) FDC and (b) SDC dictionary learning for the toy example in Appendix B.

Figure 4: Seismic Data: (a) field data one; (b) field data one with added random noise (noise standard deviation \( \sigma = 25 \)); and (c) field data two.

Figure 5: Comparison of dictionaries learned through two methods: (a) FDC; (b) SDC.

Figure 6: Denoising results for field data with noise standard deviation \( \sigma = 25 \): (a) FX-Decon; (b) Curvelet; (c) FDC-OMP; (d) SDC-OMP; (e) FDC-Graph; (f) SDC-Graph.

Figure 7: Difference between denoising results and field data one in Figure 4(a): (a) FX-Decon; (b) Curvelet; (c) FDC-OMP; (d) SDC-OMP; (e) FDC-Graph; (f) SDC-Graph.

Figure 8: Single trace comparison of the reconstructions of the field data one in Figure 4(a): (a) FX-Decon; (b) Curvelet; (c) FDC-OMP; (d) SDC-OMP; (e) FDC-Graph; (f) SDC-Graph.

Figure 9: Denoising performance of FX-Decon, Curvelet, FDC-OMP, SDC-OMP, FDC-Graph and SDC-Graph algorithms for different noise standard deviation and comparison of computation costs.

Figure 10: Denoising of field data two by FX-Decon (a), Curvelet (b), FDC-Graph (c) and SDC-Graph (d).

Figure 11: Difference between denoising results and field data two by FX-Decon (a), Curvelet (b), FDC-Graph (c) and SDC-Graph (d).
Figure 1: Example for a weighted and undirected graph, where with weight 1 for $K$-nearest neighbors.
Figure 2: The process for dictionary learning by FDC and SDC algorithm, where $\lambda_i$ denotes the maximal singular value of center matrix $C_i$ at each node, $u_i$ and $v_i$ are the first vectors in the singular value decomposition of $C_i$. 
Figure 3: Trees for (a) FDC and (b) SDC dictionary learning for the toy example in Appendix B.
<table>
<thead>
<tr>
<th>Trace number</th>
<th>Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 4: Seismic Data: (a) field data one; (b) field data one with added random noise (noise standard deviation $\sigma = 25$); and (c) field data two.
Figure 5: Comparison of dictionaries learned through two methods: (a) FDC; (b) SDC.
Figure 6: Denoising results for field data with noise standard deviation $\sigma = 25$: (a) FX-Decon; (b) Curvelet; (c) FDC-OMP; (d) SDC-OMP; (e) FDC-Graph; (f) SDC-Graph.
Figure 7: Difference between denoising results and field data one in Figure 4(a): (a) FX-Decon; (b) Curvelet; (c) FDC-OMP; (d) SDC-OMP; (e) FDC-Graph; (f) SDC-Graph.
Figure 8: Single trace comparison of the reconstructions of the field data one in Figure 4(a): (a) FX-Decon; (b) Curvelet; (c) FDC-OMP; (d) SDC-OMP; (e) FDC-Graph; (f) SDC-Graph.
Figure 9: Denoising performance of FX-Decon, Curvelet, FDC-OMP, SDC-OMP, FDC-Graph and SDC-Graph algorithms for different noise standard deviation.
Figure 10: Denoising of field data two by FX-Decon (a), Curvelet (b), FDC-Graph (c) and SDC-Graph (d).
Figure 11: Difference between denoising results and field data two by FX-Decon (a), Curvelet (b), FDC-Graph (c) and SDC-Graph (d).