Forty-Ninth Annual Allerton Conference
Allerton House, UIUC, Illinois, USA
September 28 - 30, 2011

Simultaneous Codeword Optimization (SimCO) for Dictionary Learning

Wei Dai[†], Tao Xu[*], and Wenwu Wang[*]
[†]Department of Electrical and Electronic Engineering, Imperial College London
[*]Department of Electronic Engineering, University of Surrey
wei.dai1@imperial.ac.uk, {t.xu, w.wang}@surrey.ac.uk

***Abstract*—We consider the data-driven dictionary learning problem. The goal is to seek an over-complete dictionary from which every training signal can be best approximated by a linear combination of only a few codewords. This task is often achieved by iteratively executing two operations: sparse coding and dictionary update. In the literature, there are two benchmark mechanisms to update a dictionary. The first approach, for example the MOD algorithm, is characterized by searching for the optimal codewords while fixing the sparse coefficients. In the second approach, represented by the K-SVD method, one codeword and the related sparse coefficients are simultaneously updated while all other codewords and coefficients remain unchanged. We propose a novel framework that generalizes the aforementioned two methods. The unique feature of our approach is that one can update an arbitrary set of codewords and the corresponding sparse coefficients simultaneously: when sparse coefficients are fixed, the underlying optimization problem is the same as that in the MOD algorithm; when only one codeword is selected for update, it can be proved that the proposed algorithm is equivalent to the K-SVD method; and more importantly, our method allows to update all codewords and all sparse coefficients simultaneously, hence the term *simultaneously codeword optimization (SimCO)*. Under the proposed framework, we design two algorithms, namely the primitive and regularized SimCO. Simulations demonstrate that our approach excels the benchmark K-SVD in terms of both learning performance and running speed.**

I. Introduction

Sparse signal representations have recently received extensive research interests across several communities including signal processing, information theory, and optimization [1], [2], [3], [4]. The basic assumption underlying this technique is that a natural signal can be approximated by the combination of only a small number of elementary components, called *codewords* or *atoms*, that are chosen from a dictionary (i.e., the whole collection of all the codewords). Sparse representations have found successful applications in data interpretation [5], [6], source separation [7], [8], [9], signal denoising [10], [11], coding [12], [13], [14], classification [15], [16], [17], recognition [18], impainting [19], [20] and many more (see e.g. [21]).

Two related problems have been studied either separately or jointly in sparse representations. The first one is sparse coding, that is, to find the sparse linear decompositions of a signal for a given dictionary. Efforts dedicated to this problem have resulted in the creation of a number of algorithms including basis pursuit (BP) [22], matching pursuit (MP) [23], orthogonal matching pursuit (OMP) [24], [25], subspace pursuit (SP) [26], [27], regression shrinkage and selection (LASSO) [28], focal under-determined system solver (FOCUSS) [29], and gradient pursuit (GP) [30]. Sparse decompositions of a signal, however, highly rely on the degree of fits between the data and the dictionary, which leads to the second problem, i.e. the issue of dictionary design.

An over-complete dictionary, of which the number of codewords is greater than the dimension of the signal, can be obtained by either an analytical or a learning-based approach. The analytical approach generates the dictionary based on a predefined mathematical transform, such as discrete Fourier transform (DFT), discrete cosine transform (DCT), wavelets [31], curvelets [32], contourlets [33], and bandelets [34]. Such dictionaries are relatively easier to obtain and more suitable for generic signals. In learning-based approaches, however, the dictionaries are adapted from a set of training data [5], [35], [36], [37], [38], [10], [39], [40], [41], [42]. Although this may involve higher computational complexity, learned dictionaries have the potential to offer improved performance as compared with predefined dictionaries, since the atoms are derived to capture the salient information directly from the signals.

Dictionary learning algorithms are often established on an optimization process involving the iteration between two stages: sparse approximation and dictionary update. First an initial dictionary is given and a signal is decomposed as a linear combination of only a few atoms from the initial dictionary. Then the atoms of the dictionary are trained with fixed or sometimes unfixed weighting coefficients. After that, the trained dictionary is used to compute the new weighting coefficients. The process is iterated until the most suitable dictionary is obtained eventually.

One of the early algorithms that adopted such a two-step structure was proposed by Olshausen and Field [5], [35], where a maximum likelihood (ML) learning method was used to sparsely code the natural images upon a redundant dictionary. Based on the same ML objective function as in [5], Engan [36] developed a more efficient algorithm, called the method of optimal directions (MOD), in which a closed-form solution for the dictionary update has been proposed. Several variants of this algorithm, such as the iterative least squares (ILS) method, have also been developed which were summarized in [43]. In 2006, Aharon, Elad and Bruckstein developed the K-SVD algorithm in [10] by generalizing the K-means algorithm for dictionary learning. This algorithm uses

978-1-4577-1818-2/11/$26.00 ©2011 IEEE

a similar block-relaxation approach to MOD, but updates the dictionary on atom-by-atom basis, without having to compute matrix inversion as required in the original MOD algorithm. The majorization method was proposed by [44] in which the original objective function is substituted by a surrogate function in each step of the optimization process.

In this paper, similar to MOD and K-SVD methods, we focus on the dictionary update step. Our major contributions include

- We propose a novel framework where the dictionary update problem is formulated as an optimization problem on manifolds. This framework allows to update an *arbitrary* subset of the codewords simultaneously, hence the term *simultaneously codeword optimization (SimCO)*. Our framework can be viewed as a generalization of the MOD and K-SVD methods: when sparse coefficients are fixed, the underlying optimization problem is the same as that in the MOD algorithm; when only one codeword is selected for update, the optimization problems that arise in both SimCO and K-SVD are identical.
- We observe that both K-SVD and the primitive SimCO have the same problem: they may converge to an ill-conditioned dictionary rather than a local minimizer. This phenomenon turns out to be the major reason to prevent a better learning performance. To address this issue, we propose a regularized SimCO, which overcomes the ill-condition problem.
- Our empirical tests show that the regularized SimCO that updates all codewords simultaneously enjoys better learning performance and faster running speed than the long-time benchmark K-SVD method.

In terms of theoretical analysis, for the first time, we prove that a gradient search on the Grassmann manifold solves the rank-one matrix approximation problem with probability one. Therefore, when only one codeword is updated in each step, SimCO and K-SVD share the same learning performance with probability one. Due to the space limitations, the results and the proofs are omitted in this paper. Interested readers may refer to the journal version of this paper [45].

The remainder of the paper is organized as follows. Section II formulates the problems of dictionary learning and update. Section III provides necessary preliminaries on manifolds and shows that dictionary update can be cast as an optimization problem on manifolds. The primitive SimCO algorithm is detailed in Section IV while the regularized SimCO algorithm is described in Section V. Numerical performance evaluations are represented in Section VI.

II. Dictionary Learning and Update

The dictionary learning problem can be formulated as follows: let $\boldsymbol{Y} \in \mathbb{R}^{m \times n}$ be the training data, where each column of $\boldsymbol{Y}$ corresponds to one training sample; one is looking for the solution to the following optimization problem

$$\min_{\boldsymbol{D} \in \mathbb{R}^{m \times d},\ \boldsymbol{X} \in \mathbb{R}^{d \times n}} \ \|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2 , \quad \text{subject to } \|\boldsymbol{D}_{:,i}\|_2 = 1,\ \forall 1 \le i \le d. \tag{1}$$

Algorithm 1 A typical dictionary learning algorithm

```
Task: find the best dictionary to represent the data sample matrix $\boldsymbol{Y}$.
Initialization: Set the initial dictionary $\boldsymbol{D}^{(1)}$. Set $J = 1$.
Repeat until convergence (use stop rule):
  • Sparse coding stage: Fix the dictionary $\boldsymbol{D}^{(J)}$ and update $\boldsymbol{X}^{(J)}$ using some sparse coding technique.
  • Dictionary update stage: Update $\boldsymbol{D}^{(J)}$, and $\boldsymbol{X}^{(J)}$ as appropriate.
  • $J = J + 1$.
```

where $\|\cdot\|_F$ and $\|\cdot\|_2$ are the Frobenius and l_2 norms respectively. The matrices $\boldsymbol{D}$ and $\boldsymbol{X}$ are often referred to as a dictionary and the corresponding coefficients respectively, and $\boldsymbol{D}_{:,i}$ denotes the i^{th} column (i.e., the i^{th} codeword) of the dictionary $\boldsymbol{D}$. In practice, it is typical that $m < d < n$, i.e., an over-complete dictionary is considered and the number of training samples is larger than the number of codewords. Generally speaking, the optimization problem (1) is ill-posed unless extra constraints are imposed on $\boldsymbol{X}$. The most common constraint on $\boldsymbol{X}$ is that $\boldsymbol{X}$ is sparse, i.e., the number of nonzero entries in $\boldsymbol{X}$, compared with the total number of entries, is small.

Most dictionary learning algorithms consists of two stages: sparse coding and dictionary update. See Algorithm 1 for the diagram of a typical dictionary learning procedure. In the sparse coding stage, the goal is to find a sparse $\boldsymbol{X}$ to minimize $\|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2$ for a given dictionary $\boldsymbol{D}$. In practice, the sparse coding problem is often approximately solved by using either ℓ_1-minimization [46] or greedy algorithms, for example, OMP [25] and SP [26] algorithms.

The focus of this paper is on the dictionary update stage. There are different formulations for this stage, leading to substantially different algorithms. In the MOD [36] method, one fixes the sparse coding matrix $\boldsymbol{X}$ and searches for the optimal dictionary $\boldsymbol{D}$, and hence essentially solves a least square problem.[1] By contrast, in the approach represented by the K-SVD method, one updates both the dictionary $\boldsymbol{D}$ and the nonzero coefficients in $\boldsymbol{X}$. In particular, in each step of the dictionary update stage of the K-SVD algorithm, one updates *one codeword* of the dictionary $\boldsymbol{D}$ and the nonzero coefficients in the corresponding row of the matrix $\boldsymbol{X}$. After sequentially updating all the codewords and their corresponding coefficients, the only fixed is the sparsity pattern, that is, the locations of the non-zeros in $\boldsymbol{X}$. As has been demonstrated empirically in [10], the K-SVD algorithm often enjoys faster convergence and produces more accurate dictionary when compared with the MOD method.

The key characteristic of our approach is to update all codewords and the corresponding non-zero coefficients *simultaneously*. Similar to K-SVD, the sparsity pattern remains

[1] When there are no constraints on the norm of the columns of $\boldsymbol{D}$, minimizing $\|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2$ for given $\boldsymbol{Y}$ and $\boldsymbol{X}$ is a standard least square problem and admits a closed-form solution. When extra constraints on the column norm are imposed, as we shall show shortly, the optimization problem is a least square problem on a product of manifolds. No closed-form solution has been found.

unchanged. More specifically, let $\Omega \subset [d] \times [n]$ contain the indices of non-zero entries in $\boldsymbol{X}$, i.e., $X_{i,j} \neq 0$ for all $(i,j) \in \Omega$ and $X_{i,j} = 0$ for all $(i,j) \notin \Omega$. We refer to this set as the *sparsity pattern*. In the dictionary learning algorithm, the sparsity pattern is often obtained via the sparse coding stage. Given the training data $\boldsymbol{Y}$ and a sparsity pattern Ω, the optimization under consideration is

$$\min_{\boldsymbol{D},\, \boldsymbol{X}} \; \|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2\,, \text{ s.t. } \|\boldsymbol{D}_{:,i}\|_2 = 1,\ \forall i \in [d]\,, \quad \text{and } X_{i,j} = 0,\ \forall\,(i,j) \notin \Omega. \tag{2}$$

With slight modification in (2), the connection between our approach and K-SVD becomes clear. Let $\mathcal{I} \subset [d]$ be an index set. Suppose that one is only interested in updating the codewords indexed by $\mathcal{I}$, i.e., $\boldsymbol{D}_{:,i}$'s with $i \in \mathcal{I}$. Suppose that the coefficients corresponding to other codewords $\boldsymbol{D}_{:,j}$'s, $j \notin \mathcal{I}$, remain constant. Let $\boldsymbol{D}_{:,\mathcal{I}}$ denote the sub-matrix of $\boldsymbol{D}$ formed by the columns of $\boldsymbol{D}$ indexed by $\mathcal{I}$. Let $\boldsymbol{X}_{\mathcal{I},:}$ denote the sub-matrix of $\boldsymbol{X}$ consisting of the rows of $\boldsymbol{X}$ indexed by $\mathcal{I}$. Then the optimization problem that only updates $\boldsymbol{D}_{:,\mathcal{I}}$ and $\boldsymbol{X}_{\mathcal{I},:}$ is given by

$$\min_{\boldsymbol{D}_{:,\mathcal{I}},\, \boldsymbol{X}_{\mathcal{I},:}} \; \|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2\,, \text{ s.t. } \|\boldsymbol{D}_{:,i}\|_2 = 1,\ \forall i \in \mathcal{I}, \quad \text{and } X_{i,j} = 0,\ \forall\,(i,j) \notin \Omega. \tag{3}$$

When the index set $\mathcal{I}$ contains only one entry, the optimization problem (3) is the one that each step of K-SVD dictionary update is designed to solve. When $\mathcal{I} = [d]$, the optimization problems (2) and (3) are identical.

For compositional convenience, we introduce the following notations. Suppose that we are updating the codewords indexed by $\mathcal{I}$, i.e., $\boldsymbol{D}_{:,\mathcal{I}}$, and the corresponding coefficients, i.e., $\boldsymbol{X}_{\mathcal{I},:}$. Define

$$\boldsymbol{Y}_r = \boldsymbol{Y} - \boldsymbol{D}_{:,\mathcal{I}^c}\boldsymbol{X}_{\mathcal{I}^c,:}.$$

where $\mathcal{I}^c$ is a set complementary to $\mathcal{I}$. Clearly,

$$\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X} = \boldsymbol{Y}_r - \boldsymbol{D}_{:,\mathcal{I}}\boldsymbol{X}_{\mathcal{I},:}.$$

Define the following function

$$f_{\mathcal{I}}(\boldsymbol{D}) = \min_{\boldsymbol{X}_{\mathcal{I},:}:\ X_{i,j}=0,\ \forall (i,j)\notin\Omega} \|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2\,.$$

It is clear that

$$f_{\mathcal{I}}(\boldsymbol{D}) = \min_{\boldsymbol{X}_{\mathcal{I},:}:\ X_{i,j}=0,\ \forall (i,j)\notin\Omega} \|\boldsymbol{Y}_r - \boldsymbol{D}_{:,\mathcal{I}}\boldsymbol{X}_{\mathcal{I},:}\|_F^2\,. \tag{4}$$

Hence, the optimization problem (3) can be written as

$$\min_{\boldsymbol{D}_{:,\mathcal{I}}} f_{\mathcal{I}}(\boldsymbol{D}) \text{ subject to } \|\boldsymbol{D}_{:,i}\|_2 = 1,\ \forall i \in \mathcal{I}. \tag{5}$$

The algorithmic details on how to solve (5) is presented in Section IV.

III. Preliminaries on Manifolds

To design an algorithm to solve the optimization problem (5) properly, we introduce some preliminary knowledge about Stiefel and Grassmann manifolds. In particular, the Stiefel manifold $\mathcal{U}_{m,1}$ is defined as

$$\mathcal{U}_{m,1} = \left\{\boldsymbol{u} \in \mathbb{R}^m :\ \boldsymbol{u}^T\boldsymbol{u} = 1\right\}.$$

The Grassmann manifold $\mathcal{G}_{m,1}$ is defined as

$$\mathcal{G}_{m,1} = \left\{\text{span}\left(\boldsymbol{u}\right) :\ \boldsymbol{u} \in \mathcal{U}_{m,1}\right\}.$$

Here, the notations $\mathcal{U}_{m,1}$ and $\mathcal{G}_{m,1}$ follow from the convention in [47], [48]. Note that each element in $\mathcal{U}_{m,1}$ is a unit-norm vector while each element in $\mathcal{G}_{m,1}$ is a one-dimensional subspace in $\mathbb{R}^m$. For any given $\boldsymbol{u} \in \mathcal{U}_{m,1}$, it can generate a one-dimensional subspace $\mathscr{U} \in \mathcal{G}_{m,1}$. Meanwhile, any given $\mathscr{U} \in \mathcal{G}_{m,1}$ can be generated from different $\boldsymbol{u} \in \mathcal{U}_{m,1}$: if $\mathscr{U} = \text{span}\left(\boldsymbol{u}\right)$, then $\mathscr{U} = \text{span}\left(-\boldsymbol{u}\right)$ as well.

With these definitions, the dictionary $\boldsymbol{D}$ can be interpreted as the Cartesian product of d many Stiefel manifolds $\mathcal{U}_{m,1}$. Each codeword (column) in $\boldsymbol{D}$ is one element in $\mathcal{U}_{m,1}$. It looks straightforward that optimization over $\boldsymbol{D}$ is an optimization over the product of Stiefel manifolds.

What is not so obvious is that the optimization is actually over the product of Grassmann manifolds. For any given pair $(\boldsymbol{D}, \boldsymbol{X})$, if the signs of $\boldsymbol{D}_{:,i}$ and $\boldsymbol{X}_{i,:}$ change simultaneously, the value of the objective function $\|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2$ stays the same. Let $\boldsymbol{D} = [\boldsymbol{D}_{:,1}, \cdots, \boldsymbol{D}_{:,i-1}, \boldsymbol{D}_{:,i}, \boldsymbol{D}_{:,i+1}, \cdots, \boldsymbol{D}_{:,d}]$ and $\boldsymbol{D}' = [\boldsymbol{D}_{:,1}, \cdots, \boldsymbol{D}_{:,i-1}, -\boldsymbol{D}_{:,i}, \boldsymbol{D}_{:,i+1}, \cdots, \boldsymbol{D}_{:,d}]$. Then it is straightforward to verify that $f_{[d]}(\boldsymbol{D}) = f_{[d]}(\boldsymbol{D}')$. In other words, it does not matter what $\boldsymbol{D}_{:,i}$ is; what matters is the generated subspace $\text{span}\left(\boldsymbol{D}_{:,i}\right)$. As shall become explicit later, this phenomenon has significant impacts on the algorithm design and performance analysis.

IV. Basic Algorithm for SimCO Dictionary Update

This section is devoted to solve the optimization problem (5). For simplicity and a proof-of-concept, we use a gradient descent method. The gradient computation is detailed in Subsection IV-A. How to search on the manifold product space is specified in Subsection IV-B. The overall diagram of dictionary update is described in Algorithm 2.

A. Gradient computation.

In this subsection, we give the closed form formulas for computing $f_{\mathcal{I}}(\boldsymbol{D})$ and $\nabla f_{\mathcal{I}}(\boldsymbol{D})$. Due to the space limit, we omit the derivation details, which can be found in the journal version of this paper [45].

Recall the definition of $f_{\mathcal{I}}(\boldsymbol{D})$ in (4). For a given $\boldsymbol{D}$, computing $f_{\mathcal{I}}(\boldsymbol{D})$ is essentially solving the corresponding least square problem. For a given $j \in [N]$, let $\Omega(:,j) = \{i :\ (i,j) \in \Omega\}$. Similarly, we define $\Omega(i,:) = \{j :\ (i,j) \in \Omega\}$. It can be verified that

$$f_{\mathcal{I}}(\boldsymbol{D}) = \sum_{j=1}^{n} \underbrace{\min_{\boldsymbol{X}_{\mathcal{I}\cap\Omega(:,j),j}} \left\|\left(\boldsymbol{Y}_r\right)_{:,j} - \boldsymbol{D}_{:,\mathcal{I}\cap\Omega(:,j)}\boldsymbol{X}_{\mathcal{I}\cap\Omega(:,j),j}\right\|_2^2}_{f_{\mathcal{I},j}(\boldsymbol{D})}. \tag{6}$$

Note that every atomic function $f_{\mathcal{I},j}(\boldsymbol{D})$ corresponds to a quadratic optimization problem of the form $\min_{\boldsymbol{x}}\ \|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|_F^2$. The optimal $\boldsymbol{X}^*$ admits the following closed-from

$$\boldsymbol{X}^*_{i,j} = 0,\ \forall\,(i,j) \notin \Omega,\ \boldsymbol{X}^*_{\mathcal{I}^c,:} = \boldsymbol{X}_{\mathcal{I}^c,:} \\ \boldsymbol{X}^*_{\mathcal{I}\cap\Omega(:,j),j} = \boldsymbol{D}^{\dagger}_{:,\mathcal{I}\cap\Omega(:,j)}\left(\boldsymbol{Y}_r\right)_{:,j},\ \forall j \in [n]\,, \tag{7}$$

where the superscript † denotes the pseudo-inverse of a matrix. Note that the complexity of computing pseudo-inverse could be large. There are multiple ways to compute $\boldsymbol{X}^*_{\mathcal{I}\cap\Omega(:,j),j}$ without computing the pseudo-inverse explicitly, for example, the conjugate gradient method [49]. The gradient of $f_{\mathcal{I}}(\boldsymbol{D})$, with respect to $\boldsymbol{D}_{:,i}$, $i \in \mathcal{I}$, can be computed via

$$\begin{aligned}\nabla_{\boldsymbol{D}_{:,i}} f_{\mathcal{I}}(\boldsymbol{D}) &= -2\left(\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X}^*\right)_{:,\Omega(i,:)} \boldsymbol{X}^{*T}_{i,\Omega(i,:)} \\ &= -2\left(\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X}^*\right)\boldsymbol{X}^{*T}_{i,:}. \qquad (8)\end{aligned}$$

Here, $\Omega(i,:)$ gives the columns of $\boldsymbol{Y}$ whose sparse representation involves the codeword $\boldsymbol{D}_{:,i}$. For convenience, we use the symbol $\boldsymbol{g}_i$ to denote $\nabla_{\boldsymbol{D}_{:,i}} f_{\mathcal{I}}(\boldsymbol{D})$.

When $\mathcal{I} = [d]$, i.e., all the codewords and the corresponding non-zero coefficients are simultaneously updated, the closed form formulas for both $\boldsymbol{X}^*$ and $\nabla_{\boldsymbol{D}} f$ have the following simplified form,

$$\begin{aligned}&\boldsymbol{X}^*_{i,j} = 0,\ \forall (i,j) \notin \Omega, \\ &\boldsymbol{X}^*_{\Omega(:,j),j} = \boldsymbol{D}^{\dagger}_{:,\Omega(:,j)} \boldsymbol{Y}_{:,j},\ \forall j \in [N],\end{aligned}$$

and

$$\nabla_{\boldsymbol{D}} f_{[d]}(\boldsymbol{D}) = -2\left(\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X}^*\right)\boldsymbol{X}^{*T}.$$

B. Line search along the gradient descent direction

The line search mechanism used in this paper is significant different from the standard one for the Euclidean space. Specifically, we employ a line search procedure over the product space of Grassmann manifolds. As has been discussed in Section III, the objective function $f_{\mathcal{I}}$ is indeed a function on the product of Grassmann manifolds. The search path should lie on the same space as well. For this purpose, we use the geodesic path on $\mathcal{G}_{m,1}$. A geodesic path on $\mathcal{G}_{m,1}$ is similar to the straight line in the Euclidean space: for any given two distinct points on $\mathcal{G}_{m,1}$, the geodesic path is the shortest length path that connects these two points [47]. In particular, let $\mathscr{U} \in \mathcal{G}_{m,1}$ and $\boldsymbol{u} \in \mathcal{U}_{m,1}$ be the corresponding generator matrix (not unique). Consider a search direction $\boldsymbol{h} \in \mathbb{R}^m$ with $\|\boldsymbol{h}\|_2 = 1$ and $\boldsymbol{h}^T\boldsymbol{u} = 0$. Then the geodesic path starting from $\boldsymbol{u}$ along the direction $\boldsymbol{h}$ is given by [47]

$$\boldsymbol{u}(t) = \boldsymbol{u}\cdot\cos t + \boldsymbol{h}\cdot\sin t,\ t \in \mathbb{R}.$$

For the dictionary update problem at hand, the line search path is defined as follows. Let $\boldsymbol{g}_i = \nabla_{\boldsymbol{D}_{:,i}} f_{\mathcal{I}}(\boldsymbol{D})$ be the gradient vector defined in (8). We define

$$\bar{\boldsymbol{g}}_i = \boldsymbol{g}_i - \boldsymbol{D}_{:,i}\boldsymbol{D}^T_{:,i}\boldsymbol{g}_i,\ \forall i \in \mathcal{I}. \qquad (9)$$

According to [47], $\bar{\boldsymbol{g}}_i$ is in fact the gradient of f with respect to $\boldsymbol{D}_{:,i}$ on the Grassmann manifold. The line search path for dictionary update, say $\boldsymbol{D}(t)$, $t \geq 0$, is defined as

$$\begin{cases}\boldsymbol{D}_{:,i}(t) = \boldsymbol{D}_{:,i} \quad \text{if } i \notin \mathcal{I} \text{ or } \|\bar{\boldsymbol{g}}_i\|_2 = 0, \\ \boldsymbol{D}_{:,i}(t) = \boldsymbol{D}_{:,i}\cos\left(\|\bar{\boldsymbol{g}}_i\|_2 t\right) - \left(\bar{\boldsymbol{g}}_i / \|\bar{\boldsymbol{g}}_i\|_2\right)\sin\left(\|\bar{\boldsymbol{g}}_i\|_2 t\right) \\ \qquad\qquad \text{if } i \in \mathcal{I} \text{ and } \|\bar{\boldsymbol{g}}_i\|_2 \neq 0.\end{cases} \qquad (10)$$

where $\boldsymbol{g}_i = \nabla_{\boldsymbol{D}_{:,i}} f_{\mathcal{I}}(\boldsymbol{D})$ can be computed via (8).

Algorithm 2 summarizes one iteration of the proposed line search algorithm. Note that the proposed algorithm is by no means optimized. There are multiple ways to reduce the number of function evaluations [49, Chapter 3]. We use the current design in Algorithm 2 for implementation convenience.

Algorithm 2 One iteration of the line search algorithm for dictionary update.

```
Task: Use line search mechanism to update the dictionary $\boldsymbol{D}$.
Input: $\boldsymbol{Y}$, $\boldsymbol{D}$, $\boldsymbol{X}$
Output: $\boldsymbol{D}'$ and $\boldsymbol{X}'$.
Parameters: $t_4 > 0$: initial step size. $g_{\min} > 0$: the threshold below which a gradient can be viewed as zero.
Initialization: Let $c = \left(\sqrt{5}-1\right)/2$.
  1) Let $t_1 = 0$. Compute $f(\boldsymbol{D})$ and the corresponding gradient on the Grassmann manifold (9). If $\|\bar{\boldsymbol{g}}_i\|_2 \leq g_{\min}\|\boldsymbol{Y}\|_F^2$ for all $i \in \mathcal{I}$, then $\boldsymbol{D}' = \boldsymbol{D}$, $\boldsymbol{X}' = \boldsymbol{X}$, and quit.
  2) Let $t_3 = ct_4$ and $t_2 = (1-c)\,t_4$.
Part A: the goal is to find $t_4 > 0$ s.t. $f(\boldsymbol{D}(t_1)) > f(\boldsymbol{D}(t_2)) > f(\boldsymbol{D}(t_3)) \leq f(\boldsymbol{D}(t_4))$. Iterate the following steps.
  3) If $f(\boldsymbol{D}(t_1)) \leq f(\boldsymbol{D}(t_2))$, then $t_4 = t_2$, $t_3 = ct_4$ and $t_2 = (1-c)\,t_4$.
  4) Else if $f(\boldsymbol{D}(t_2)) \leq f(\boldsymbol{D}(t_3))$, then $t_4 = t_3$, $t_3 = t_2$ and $t_2 = (1-c)\,t_4$.
  5) Else if $f(\boldsymbol{D}(t_3)) > f(\boldsymbol{D}(t_4))$, then $t_2 = t_3$, $t_3 = t_4$ and $t_4 = t_3/c$.
  6) Otherwise, quit the iteration.
Part B: the goal is to shrink the interval length $t_4 - t_1$ while trying to keep the relation $f(\boldsymbol{D}(t_1)) > f(\boldsymbol{D}(t_2)) > f(\boldsymbol{D}(t_3))$. Iterate the following steps until $t_4 - t_1$ is sufficiently small.
  1) 7)If $f(\boldsymbol{D}(t_1)) > f(\boldsymbol{D}(t_2)) > f(\boldsymbol{D}(t_3))$, then $t_1 = t_2$, $t_2 = t_3$ and $t_3 = t_1 + c\,(t_4 - t_1)$.
  8) Else $t_4 = t_3$, $t_3 = t_2$ and $t_2 = t_1 + (1-c)\,(t_4 - t_1)$.
Output: Let $t^* = \underset{t \in \{t_1,t_2,t_3,t_4\}}{\arg\min}\ f(\boldsymbol{D}(t))$ and $\boldsymbol{D}' = \boldsymbol{D}(t^*)$. Compute $\boldsymbol{X}'$ according to (7).
```

V. Regularized SimCO

As will be detailed in Section VI-A, both K-SVD and the primitive SimCO may result in ill-conditioned dictionaries. This section is devoted to design a mechanism, referred to as regularized SimCO, to address this problem.

Regarding the dictionary update stage of the K-SVD and SimCO methods, the ill-condition of the dictionary is described as follows. Fix the sparsity pattern Ω. The matrix $\boldsymbol{D}_{:,\Omega(:,j)}$ contains the codewords that are involved in representing the training sample $\boldsymbol{Y}_{:,j}$. We say the dictionary $\boldsymbol{D}$ is ill-conditioned with respect to the sparsity pattern Ω if

$$0 \approx \lambda_{\min}\left(\boldsymbol{D}_{:,\Omega(:,j)}\right) \ll \lambda_{\max}\left(\boldsymbol{D}_{:,\Omega(:,j)}\right)$$

for some $j \in [n]$. Here, $\lambda_{\min}(\cdot)$ and $\lambda_{\max}(\cdot)$ give the smallest and largest singular values of a matrix, respectively.

The ill-condition of $\boldsymbol{D}$ brings two problems:

1) Slow convergence in the dictionary update stage. When $\lambda_{\min}\left(\boldsymbol{D}_{:,\Omega(:,j)}\right)$ is close to zero, the curvature (Hessian matrix) of $f_{\mathcal{I}}(\boldsymbol{D})$ is large. That means, in a neighborhood of $\boldsymbol{D}$, the gradient changes very fast. As a result, a gradient descent algorithm typically suffers from a very slow convergence rate.

2) Instability in the subsequent sparse coding stage. When $\lambda_{\min}\left(\boldsymbol{D}_{:,\Omega(:,j)}\right)$ is close to zero, the solution of the least square problem $\min_{\boldsymbol{X}_{\Omega(:,j),j}} \left\|\boldsymbol{Y}_{:,j} - \boldsymbol{D}_{:,\Omega(:,j)}\boldsymbol{X}_{\Omega(:,j),j}\right\|_F^2$ becomes unstable: small changes in $\boldsymbol{Y}_{:,j}$ often result in very different least square solutions $\boldsymbol{X}^*_{\Omega(:,j),j}$. It is well known that the stability of sparse coding relies on the so called restricted isometry condition (RIP) [46]. RIP says that for a given positive integer $K < n$, the conditional numbers of *all sub-matrices* of $\boldsymbol{D}$ of the form $\boldsymbol{D}_{:,\mathcal{J}}$, where $\mathcal{J} \subset [n]$ and $|\mathcal{J}| \leq K$, are uniformly upper bounded. An ill-conditioned $\boldsymbol{D}$ clearly does not satisfy RIP. Hence, the sparse coefficients produced by sparse coding will be sensitive to noise.

It is also worth mentioning an important difference between a local minimizer and an ill-conditioned dictionary. When the trained dictionary is close to a local minimizer or when the trained dictionary is ill-conditioned, it is typical that the objective function stops decreasing as the number of iterations increases. Hence it is usually difficult to distinguish these two cases by studying the objective function only. However, the difference becomes apparent if one looks at the gradient of the objective function (with respect to the dictionary): the gradient of the objective function is close to zero in the neighborhood of a local minimizer while it remains relatively large in the neighborhood of an ill-conditioned dictionary. The same phenomenon is not isolated as it was also observed in the manifold learning approach for the low-rank matrix completion problem [48].

To mitigate the problem of the ill-conditioned $\boldsymbol{D}$, we propose to optimize a regularized objective function

$$\widetilde{f}_{\mathcal{I}}\left(\boldsymbol{D}\right) = \sum_{j=1}^{n} \min_{\boldsymbol{X}_{\mathcal{I}\cap\Omega(:,j),j}} \underbrace{\left(\left\| \left(\boldsymbol{Y}_r\right)_{:,j} - \boldsymbol{D}_{:,\mathcal{I}\cap\Omega(:,j)}\boldsymbol{X}_{\mathcal{I}\cap\Omega(:,j),j} \right\|_2^2 + \mu \left\|\boldsymbol{X}_{\mathcal{I}\cap\Omega(:,j),j}\right\|_2^2 \right)}_{\widetilde{f}_{\mathcal{I},j}(\boldsymbol{D})}. \tag{11}$$

where $\mu > 0$ is a constant. The motivation is as follows: when $\lambda_{\min}\left(\boldsymbol{D}_{:,\mathcal{I}\cap\Omega(:,j)}\right) \approx 0$ for some j, the corresponding optimal $\widetilde{\boldsymbol{X}}^*_{\mathcal{I}\cap\Omega(:,j),j}$ to solve $f_{\mathcal{I},j}\left(\boldsymbol{D}\right)$ is large; after the regularized term $\mu\left\|\boldsymbol{X}_{\mathcal{I}\cap\Omega(:,j),j}\right\|_2^2$ is introduced, the optimal $\widetilde{\boldsymbol{X}}^*_{\mathcal{I}\cap\Omega(:,j),j}$ to solve $\widetilde{f}_{\mathcal{I},j}\left(\boldsymbol{D}\right)$ is uniformly bounded. As a result, with the regularized term, the optimization of $\widetilde{f}_{\mathcal{I}}\left(\boldsymbol{D}\right)$ over $\boldsymbol{D}$ tends to provide a well-conditioned solution $\boldsymbol{D}$ with small $\left\|\boldsymbol{X}_{\mathcal{I},:}\right\|_F^2$.

With the new objective function (11), the steps described in Algorithm 2 can be directly applied. The only difference lies in the computation of the new function $\widetilde{f}_{\mathcal{I}}\left(\boldsymbol{D}\right)$ and the corresponding gradient $\nabla_{\boldsymbol{D}}\widetilde{f}_{\mathcal{I}}\left(\boldsymbol{D}\right)$. In order to compute $\widetilde{f}_{\mathcal{I}}\left(\boldsymbol{D}\right)$, one needs to solve the least square problem in (11). Let $m_j = \left|\mathcal{I}\cap\Omega\left(:,j\right)\right|$. It is clear that $\boldsymbol{D}_{:,\boldsymbol{X}_{\mathcal{I}\cap\Omega(:,j),j}} \in \mathbb{R}^{m\times m_j}$ and $\boldsymbol{X}_{\mathcal{I}\cap\Omega(:,j),j} \in \mathbb{R}^{m_j}$. Define

$$\widetilde{\boldsymbol{Y}}_{r,j} = \left[\begin{array}{c} \left(\boldsymbol{Y}_r\right)_{:,j} \\ \boldsymbol{0}_{m_j} \end{array}\right], \text{ and } \widetilde{\boldsymbol{D}}_j = \left[\begin{array}{c} \boldsymbol{D}_{:,\mathcal{I}\cap\Omega(:,j)} \\ \sqrt{\mu}\cdot\boldsymbol{I}_{m_j} \end{array}\right],$$

where $\boldsymbol{0}_{m_j}$ is the zero vector of length m_j, and $\boldsymbol{I}_{m_j}$ is the $m_j \times m_j$ identity matrix. The optimal $\widetilde{\boldsymbol{X}}^*_{\mathcal{I}\cap\Omega(:,j),j}$ to solve the least square problem in (11) is given by

$$\widetilde{\boldsymbol{X}}^*_{\mathcal{I}\cap\Omega(:,j),j} = \widetilde{\boldsymbol{D}}_j^{\dagger}\widetilde{\boldsymbol{Y}}_{r,j}. \tag{12}$$

Hence, the new objective function $\widetilde{f}_{\mathcal{I}}\left(\boldsymbol{D}\right)$ is computed via

$$\widetilde{f}_{\mathcal{I}}\left(\boldsymbol{D}\right) = \left\|\boldsymbol{Y}_r - \boldsymbol{D}\widetilde{\boldsymbol{X}}^*\right\|_F^2 + \mu\cdot\left\|\widetilde{\boldsymbol{X}}^*_{\mathcal{I},:}\right\|_F^2, \tag{13}$$

and the gradient of $\widetilde{f}_{\mathcal{I}}$ regarding to $\boldsymbol{D}_{:,\mathcal{I}}$ is given by

$$\nabla_{\boldsymbol{D}_{:,\mathcal{I}}}\widetilde{f}_{\mathcal{I}}\left(\boldsymbol{D}\right) = -2\left(\boldsymbol{Y} - \boldsymbol{D}\widetilde{\boldsymbol{X}}^*\right)\widetilde{\boldsymbol{X}}^{*T}_{\mathcal{I},:}. \tag{14}$$

(Computation details are omitted due to the space limitations. See the journal version of this paper [45] for details.) Replace the $\boldsymbol{X}^*_{\mathcal{I},:}$, $f_{\mathcal{I}}\left(\boldsymbol{D}\right)$ and $\nabla_{\boldsymbol{D}_{:,\mathcal{I}}}f_{\mathcal{I}}\left(\boldsymbol{D}\right)$ in Algorithm 2 with the $\widetilde{\boldsymbol{X}}^*_{\mathcal{I},:}$, $\widetilde{f}_{\mathcal{I}}\left(\boldsymbol{D}\right)$, and $\nabla_{\boldsymbol{D}_{:,\mathcal{I}}}\widetilde{f}_{\mathcal{I}}\left(\boldsymbol{D}\right)$ computed above. We obtain the regularized SimCO for dictionary update.

In practice, one may consider to first use the regularized SimCO to obtain a reasonably good dictionary and then employ the primitive SimCO to refine the dictionary further. This two step procedure often results in a well-conditioned dictionary that fits the training data. Please see the simulation part (Section VI) for an example.

VI. Empirical Tests

In this section, we numerically test the proposed algorithms, i.e., the primitive and the regularized SimCO, using synthetic and real data, and compare them with the benchmark K-SVD method. To simplify the comparison, for both the primitive and the regularized SimCO, the set of updating codewords $\mathcal{I}$ is always set to $\mathcal{I} = [d]$. In Section VI-A, we show that both the K-SVD and the primitive SimCO may result in an ill-conditioned dictionary while adding a regularized term can avoid this problem. Empirical experiments on synthetic and real data are detailed in Sections VI-B and VI-C respectively. The results demonstrate the excellent learning performance of the regularized SimCO. Finally, we compare the running time of the regularized SimCO and K-SVD in Section VI-D. It is interesting to observe that though regularized SimCO is built on a simple gradient descent mechanism, it runs much faster than the benchmark K-SVD dictionary update.

A. Ill-conditioned Dictionaries

In this subsection, we handpick a particular example to show that both the K-SVD and the primitive SimCO may converge to an ill-conditioned dictionary. In the example, the training samples $\boldsymbol{Y} \in \mathbb{R}^{16\times 78}$ are computed via $\boldsymbol{Y} = \boldsymbol{D}_{\text{true}}\boldsymbol{X}_{\text{true}}$, where $\boldsymbol{D}_{\text{true}} \in \mathbb{R}^{16\times 32}$ is a dictionary, $\boldsymbol{X}_{\text{true}} \in \mathbb{R}^{32\times 78}$ is the corresponding sparse coefficient matrix, and each column of $\boldsymbol{X}$ contains exactly 4 nonzero components. To test the training performance of different algorithms, we randomly generate the initial dictionary $\boldsymbol{D}_0$ and the initial coefficient matrix $\boldsymbol{X}_0$: every column of $\boldsymbol{D}_0$ is uniformly generated from the uniform distribution on the Stiefel manifold $\mathcal{U}_{16,1}$; and $\boldsymbol{X}_0$ and $\boldsymbol{X}$ have the exactly same sparsity pattern but the nonzero entries of $\boldsymbol{X}_0$ is randomly drawn from the standard Gaussian distribution. All tested algorithms start with the same input $\boldsymbol{Y}$, $\boldsymbol{D}_0$ and $\boldsymbol{X}_0$.

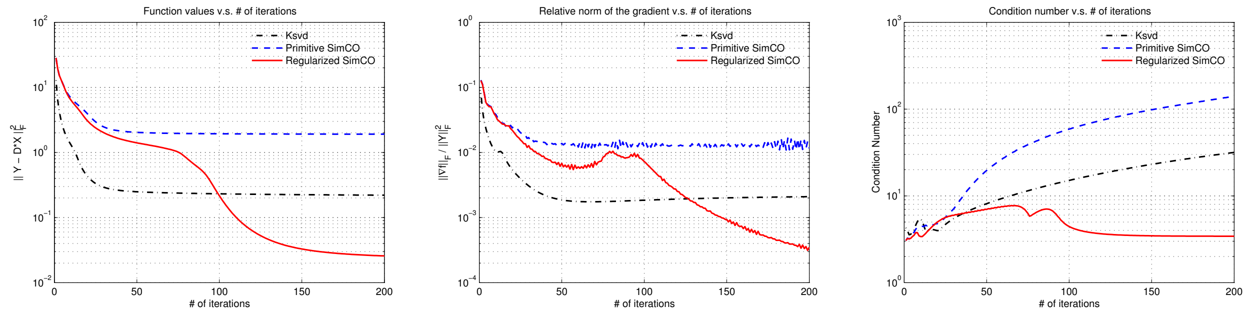


Figure 1: Starting with the same point, the behaviors of K-SVD, primitive SimCO and regularized SimCO are different.

For regularized SimCO, the regularization constant μ is set to $\mu = 0.01$.

The numerical results are presented in Figure 1. In the left sub-figure, though both the K-SVD and the primitive SimCO minimize $f(\boldsymbol{D}) = \min_{\boldsymbol{X}} \|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2$ while the regularized SimCO minimizes $\widetilde{f}(\boldsymbol{D}) = \min_{\boldsymbol{X}} \|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2 + \mu \|\boldsymbol{X}\|_F^2$, we compare only the quantities $\|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2$. In the middle sub-figure, we depict $\nabla_{\boldsymbol{D}} f(\boldsymbol{D})$ for K-SVD and the primitive SimCO, and $\nabla_{\boldsymbol{D}} \widetilde{f}(\boldsymbol{D})$ for the regularized SimCO as the search direction depends on the gradient. In the right sub-figure, we show the condition number of the dictionary defined as

$$\kappa(\boldsymbol{D}) = \max_{1 \leq j \leq d} \lambda_{\max}\left(\boldsymbol{D}_{:,\Omega(:,j)}\right) / \lambda_{\min}\left(\boldsymbol{D}_{:,\Omega(:,j)}\right).$$

Here, note that $\kappa(\boldsymbol{D}_{\text{true}}) = 3.39$.

Figure 1 shows that

1) When the number of iterations exceeds 50, both the K-SVD and the primitive SimCO stops improving the training performance: the value of f decreases very slowly with further iterations. However, the gradients in these two methods surprisingly increase slightly (in the Frobenius norm) with further iterations. This implies that these two methods *do not converge to local minimizers*.
2) The above phenomenon can be well explained by checking the ill-condition of the dictionary. After 100 iterations, $\kappa(\boldsymbol{D}) > 10$ for K-SVD , $\kappa(\boldsymbol{D}) > 50$ for primitive SimCO, and they keep increasing with further iterations. This is very similar to what was observed in [48]: the training variable seems to converge to a singular point.
3) By adding a regularized term, the regularized SimCO avoids the convergence to an ill-conditioned dictionary. Though $\tilde{f} \neq f$, regularized SimCO can finds a good dictionary in terms of f when the regularization constant $\mu > 0$ is chosen small enough.

The necessity of regularized SimCO is therefore clear.

B. Experiments on Synthetic Data

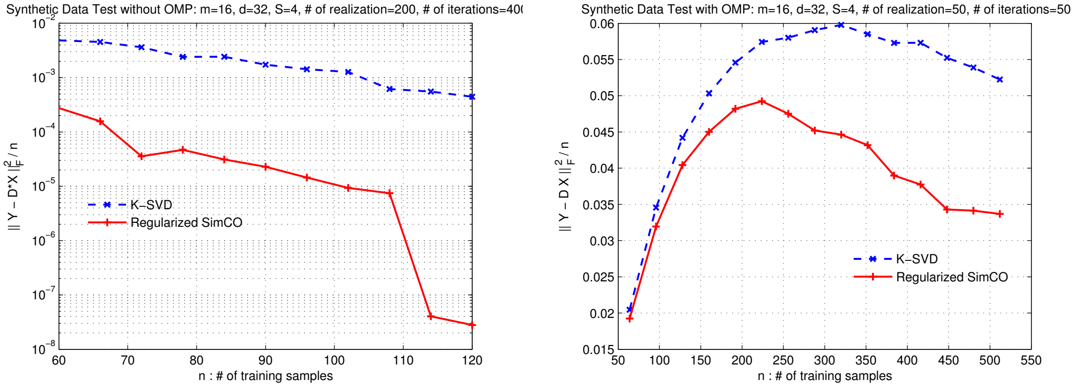
In the synthetic data tests, we assume that $\boldsymbol{Y} = \boldsymbol{D}_{\text{true}}\boldsymbol{X}_{\text{true}}$ where the columns of $\boldsymbol{D}_{\text{true}}$ are randomly generated from the Stiefel manifold $\mathcal{U}_{m,1}$, each column of $\boldsymbol{X}_{\text{true}}$ contains exactly S many non-zeros: the position of the non-zeros are uniformly distributed on the set $\binom{[d]}{S} = \{\{i_1, \cdots, i_S\} : 1 \leq i_k \neq i_\ell \leq d\}$; and the values of the non-zeros are standard Gaussian distributed. In our simulations, we fix $m = 16$, $d = 32$, and $S = 4$. We change the number of training samples n. For each n value, we run 100 random tests. In each random test, we also randomly generate an initial dictionary $\boldsymbol{D}_0$ and an initial coefficient matrix $\boldsymbol{X}_0$.

We first test the performance of dictionary update without taking the effect of sparse coding into consideration. In particular, we assume the true sparsity is known by setting the sparsity pattern of $\boldsymbol{X}_0$ the same as that of $\boldsymbol{X}_{\text{true}}$. Note that the regularized SimCO is equivalent to the primitive SimCO only when the regularization constant $\mu = 0$. The ideal way to test regularized SimCO is to sequentially decrease μ to zero and let the regularized SimCO converge for each value of μ. In practice, we choose the following simple strategy: the total number of iterations is set to 400; we set μ to $1e - 1$, $1e - 2$, $1e - 3$, and $1e - 4$, for iterations 1-100, 101-200, 201-300, and 301-400, respectively. For fair comparison, we also set the number of iterations in K-SVD dictionary update to 400. The numerical results of $\|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2 / n$ versus n are presented in Figure 2. The average performance of the regularized SimCO is consistently better than that of the K-SVD.

Then we evaluate the overall dictionary learning performance by combining the dictionary update and sparse coding stages. For sparse coding, we adopt the OMP algorithm [25] as it has been intensively used for testing the K-SVD method in [10], [50]. The whole dictionary learning process follows Algorithm 1. We refer to the iterations between sparse coding and dictionary learning stages as outer-iterations, and the iterations within the dictionary update stage as inner-iterations. In our test, the numbers of outer-iterations are set to 50 for both the K-SVD and the regularized SimCO, and in each outer iteration, the numbers of inner-iterations of both algorithms are set to 1. Furthermore, in the regularized SimCO, the regularized constant is set to $\mu = 1e - 1$ during the first 30 outer-iterations, and $\mu = 0$ during the rest 20 outer-iterations. The simulation results of $\|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2 / n$ versus n are depicted in Figure 2. Again, the average performance of regularized SimCO is consistently better than that of K-SVD.

C. Numerical Results for Image Denoising

In this subsection, we apply the regularized SimCO to the application of image denoising. A corrupted image with



(a) Dictionary update only

(b) With sparse coding

Figure 2: Performance comparison of K-SVD and regularized SimCO dictionary update.

noise (the middle image of Fig. 3) was used to train the dictionary. For dictionary learning, we iterate the sparse coding and dictionary update stages for 10 times. The sparse coding stage is based on the OMP algorithm implemented in [50]. The regularized SimCO method is used in dictionary update stage: In the first 5 iterations, the regularization constant μ is set to 0.1; In the rest 5 iterations, it is set to zero. During each dictionary update stage, the line search procedure is only performed once. After the dictionary is trained, it is used for reconstructing the image. The right image in Fig. 3 shows the reconstructed one. The SNR is significantly improved from 20.16 dB to 31.08dB.

D. Comments on the Running Time

It is empirically observed that the regularized SimCO runs much faster than the K-SVD algorithm. In our tests, both algorithms are implemented in Matlab only codes. For Figure 2(a), it takes 4.10 hours by the regularized SimCO and 20.93 hours by K-SVD. For Figure 2(b), it takes 5.98 hours by the regularized SimCO and 6.45 hours by K-SVD.[2] For the image denoising in Figure 3, it takes 221 seconds by the regularized SimCO and 2515 seconds by K-SVD. The faster running speed of regularized SimCO is mainly due to the complexity reduction from singular value decomposition (required in K-SVD) for solving the least square problem.

VII. Acknowledgment

This work was partially supported by the MOD University Defence Research Centre (UDRC) in Signal Processing.

References

[1] P. Foldiak, "Forming sparse representations by local anti-Hebbian learning," *Biolog. Cybern.*, vol. 64, pp. 165–170, 1990.

[2] M. S. Lewicki and T. J. Sejnowski, "Learning overcomplete representations," *Neural Comput.*, vol. 12, no. 2, pp. 337–365, 2000.

[3] J. Tropp, "Topics in sparse approximations," Ph.D. dissertation, University of Texas at Austin, 2004.

[4] B. A. Olshausen, C. F. Cadieu, and D. K. Warland, "Learning real and complex overcomplete representations from the statistics of natural images," *Proc. SPIE*, vol. 7446, 2009.

[5] B. A. Olshausen and D. J. Field, "Emergence of simple-cell receptive field properties by learning a sparse code for natural images," *Nature*, vol. 381, pp. 607–609, 1996.

[6] I. Tošić and P. Frossard, "Dictionary learning for stereo image representation," *IEEE Trans. Image Process.*, vol. 20, no. 4, pp. 921–934, 2011.

[7] M. Zibulevsky and B. A. Pearlmutter, "Blind source separation by sparse decomposition of a signal dictionary," *Neural Comput.*, vol. 13, no. 4, pp. 863–882, 2001.

[8] R. Gribonval, "Sparse decomposition of stereo signals with matching pursuit and application to blind separation of more than two sources from a stereo mixture," in *IEEE Int. Conf. Acoust., Speech Signal Process.*, vol. 3, 2002, pp. 3057–3060.

[9] T. Xu and W. Wang, "Methods for learning adaptive dictionary in underdetermined speech separation," in *IEEE Int. Conf. Machine Learning for Signal Processing.*, Beijing, China, 2011.

[10] 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.

[11] M. G. Jafari and M. D. Plumbley, "Fast dictionary learning for sparse representations of speech signals," *IEEE J. Selected Topics in Signal Process.*, vol. 5, no. 5, pp. 1025–1031, 2011.

[12] P. Schmid-Saugeon and A. Zakhor, "Dictionary design for matching pursuit and application to motion-compensated video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 6, pp. 880–886, 2004.

[13] E. Kokiopoulou and P. Frossard, "Semantic coding by supervised dimensionality reduction," *IEEE Trans. Multimedia*, vol. 10, no. 5, pp. 806–818, 2008.

[14] M. D. Plumbley, T. Blumensath, L. Daudet, R. Gribonval, and M. E. Davies, "Sparse representations in audio and music: From coding to source separation," *Proceedings of IEEE*, vol. 98, no. 6, pp. 995–1005, 2010.

[15] K. Huang and S. Aviyente, "Sparse representation for signal classification," in *Conf. Neural Information Processing Systems*, 2007.

[16] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Discriminative learned dictionaries for local image analysis," in *IEEE Int. Conf. Computer Vision and Pattern Recognition*, 2008, pp. 1–8.

[17] K. Schnass and P. Vandergheynst, "A union of incoherent spaces model for classification," in *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, 2010, pp. 5490–5493.

[18] J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 210–227, 2009.

[19] V. Cevher and A. Krause, "Greedy dictionary selection for sparse representation," in *Proc. NIPS Workshop on Discrete Optimization in Machine Learning*, Vancouver, Canada, December 2009.

[20] A. Adler, V. Emiya, M. G. Jafari, M. Elad, R. Gribonval, and M. D. Plumbley, "Audio inpainting," *IEEE Trans. on Audio, Speech*

[2] The difference in the running time is much less significant compared to the other cases because the running time of OMP dominates in this case.

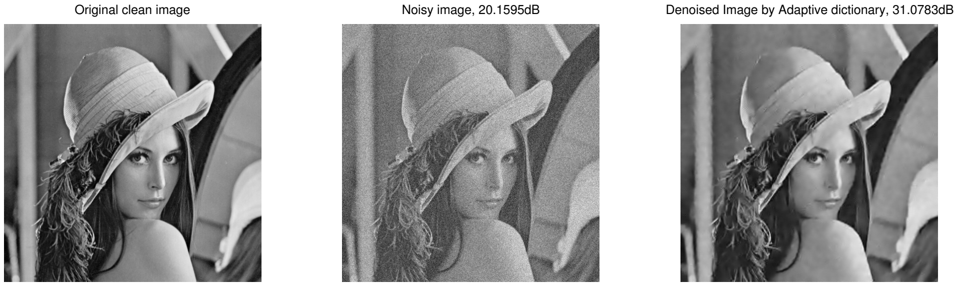


Figure 3: Example of the image denoising using the regularized SimCO. The three images above, from left to right, are the original, corrupted and reconstructed images respectively.

and Language Processing, submitted, 2011. [Online]. Available: http://www.cs.technion.ac.il/ elad/publications/journals

[21] R. G. Baraniuk, E. J. Candès, M. Elad, and Y. Ma, "Applications of sparse representation and compressive sensing," *Proceedings of the IEEE*, vol. 98, no. 6, pp. 906–909, 2010.

[22] S. Chen, D. Donoho, and M. Saunders, "Atomic decomposition by basis pursuit," *SIAM J. Sci. Comput.*, vol. 20, no. 1, pp. 33–61, 1999.

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

[24] Y. C. Pati, R. Rezaiifar, and P. S. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," in *IEEE Asilomar Conference on Signals, Systems and Computers*, 1993, pp. 40–44.

[25] J. 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.

[26] W. Dai and O. Milenkovic, "Subspace pursuit for compressive sensing signal reconstruction," *IEEE Trans. Inform. Theory*, vol. 55, pp. 2230–2249, 2009.

[27] D. Needell and J. A. Tropp, "CoSaMP: Iterative signal recovery from incomplete and inaccurate samples," *Appl. Comp. Harmonic Anal.*, vol. 26, no. 3, pp. 301–321, May 2009.

[28] R. Tibshirani, "Regression shrinkage and selection via the lasso," *J. R. Stat. Soc. Ser. B (Method.)*, vol. 58, no. 1, pp. 267–288, 1996.

[29] I. Gorodnitsky and B. Rao, "Sparse signal reconstruction from limited data using FOCUSS: a re-weighted minimum norm algorithm," *IEEE Trans. Signal Process.*, vol. 45, no. 3, pp. 600–616, 1997.

[30] T. Blumensath and M. E. Davies, "Gradient pursuits," *IEEE Trans. Signal Process.*, vol. 56, no. 6, pp. 2370–2382, 2008.

[31] I. W. Selesnick, R. G. Baraniuk, and N. C. Kingsbury, "The dual-tree complex wavelet transform," *IEEE Signal Process Mag.*, vol. 22, no. 6, pp. 123–151, 2005.

[32] E. J. Candès and D. L. Donoho, "Curvelets - a surprisingly effective nonadaptive representation for objects with edges," *Curves and Surfaces*, 2999.

[33] M. N. Do and M. Vetterli, "The contourlet transform: An efficient directional multiresolution image representation," *IEEE Trans. Image Process.*, vol. 14, no. 12, pp. 2091–2106, 2005.

[34] E. LePennec and S. Mallat, "Sparse geometric image representations with bandelets," *IEEE Trans. Image Process.*, vol. 14, no. 4, pp. 423–438, 2005.

[35] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by V1," *Vis. Res.*, vol. 37, no. 23, pp. 3311–3325, 1997.

[36] K. Engan, S. Aase, and J. H. Husøy, "Method of optimal directions for frame design," in *IEEE Int. Conf. Acoustics, Speech, and Signal Processing*, vol. 5, 1999, pp. 2443–2446.

[37] K. Kreutz-Delgado, J. Murray, B. Rao, K. Engan, T.-W. Lee, and T. J. Sejnowski, "Dictionary learning algorithms for sparse representation," *Neural Comput.*, vol. 15, no. 2, pp. 349–396, 2003.

[38] D. P. Wipf and B. D. Rao, "Sparse bayesian learning for basis selection," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2153–2164, 2004.

[39] M. D. Plumbley, "Dictionary learning for L1-exact sparse coding," *Lect. Notes Comput. Sci.*, vol. 4666, pp. 406–413, 2007.

[40] M. G. Jafari and M. D. Plumbley, "Speech denoising based on a greedy adaptive dictionary algorithm," in *European Signal Processing Conf.*, 2009, pp. 1423–1426.

[41] R. Gribonval and K. Schnass, "Dictionary identification - sparse matrix factorisation via l1 minimisation," *IEEE Trans. Inf. Theory*, vol. 56, no. 7, pp. 3523–3539, 2010.

[42] Q. Geng, H. Wang, and J. Wright, "On the local correctness of L-1 minimization for dictionary learning," *arXiv:1101.5672*, 2011.

[43] K. Engan, K. Skretting, and J. H. Husøy, "Family of iterative LS-based dictionary learning algorithms, ILS-DLA, for sparse signal representation," *Dig. Signal Process.*, vol. 17, no. 1, pp. 32–49, 2007.

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

[45] W. Dai, T. Xu, and W. Wang, "Simultaneous codeword optimization (SimCO) for dictionary update and learning," *http://arxiv.org/abs/1109.5302*, 2011.

[46] E. Candes and T. Tao, "Decoding by linear programming," vol. 51, no. 12, pp. 4203–4215, Dec. 2005.

[47] A. Edelman, T. A. Arias, and S. T. Smith, "The geometry of algorithms with orthogonality constraints," *SIAM J. MATRIX ANAL. APPL.*, vol. 20, no. 2, pp. 303–353, 1999.

[48] W. Dai, E. Kerman, and O. Milenkovic, "A geometric approach to low-rank matrix completion," *IEEE Trans. Inform. Theory*, 2011, accepted.

[49] J. Nocedal and S. J. Wright, *Numerical Optimization*. Springer, 2006.

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