Simultaneous Codeword Optimization (SimCO) for Dictionary Update and Learning

Wei Dai, *Member IEEE*, Tao Xu, *Student Member IEEE*, Wenwu Wang, *Senior Member IEEE*

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. The focus of this paper is on the dictionary update step, where the dictionary is optimized with a given sparsity pattern. We propose a novel framework where an arbitrary set of codewords and the corresponding sparse coefficients are simultaneously updated, hence the term *simultaneous codeword optimization (SimCO)*. The SimCO formulation not only generalizes benchmark mechanisms MOD and K-SVD, but also allows the discovery that singular points, rather than local minima, are the major bottleneck of dictionary update. To mitigate the problem caused by the singular points, regularized SimCO is proposed. First and second order optimization procedures are designed to solve regularized SimCO. Simulations show that regularization substantially improves the performance of dictionary learning.

I. Introduction

Sparse signal representations have recently received extensive research interest 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

This work was supported by the MOD University Defence Research Centre (UDRC) in Signal Processing, and in part by the Centre for Vision Speech and Signal Processing (CVSSP), the China Scholarship Council (CSC), and the Engineering and Physical Sciences Research Council (EPSRC) of the UK (grant numbers EP/H050000/1 and EP/H012842/1). Parts of the results of this work were presented on the 49th Annual Allerton Conference on Communication, Control, and Computing (Illinois, USA, Sept 28-30, 2011), and the 37th IEEE International Conference on Acoustics, Speech and Signal Processing (Kyoto, Japan, March 25-30, 2012).

W. Dai is with the Department of Electrical and Electronic Engineering, Imperial College London, London SW7 2AZ, United Kingdom (email: wei.dai1@imperial.ac.uk).

T. Xu and W. Wang are with the Department of Electronic Engineering, University of Surrey, Guildford GU2 7XH, United Kingdom (email: t.xu and w.wang@surrey.ac.uk).

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, rely greatly on the degree of fitting between the data and the dictionary, which leads to the second problem, i.e. the issue of dictionary design.

An over-complete dictionary, one in 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 easy 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 this 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 eventually obtained.

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 natural

images upon a redundant dictionary. The sparse approximation step in the ML algorithm [5] which involves probabilistic inference is computationally expensive. In a similar probabilistic framework, Kreutz-Delgado *et al.* [37] proposed a maximum a posteriori (MAP) dictionary learning algorithm, where the maximization of the likelihood function as used in [5] is replaced by the maximization of the posterior probability that a given signal can be synthesized by a dictionary and the sparse coefficients. Based on the same ML objective function as in [5], Engan *et al.* [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. This method is one of the earliest methods that implements the concept of sparification process [43]. Several variants of this algorithm, such as the iterative least squares (ILS) method, have also been developed which were summarized in [44]. A recursive least squares (RLS) dictionary learning algorithm was recently presented in [45] where the dictionary is continuously updated as each training vector is being processed, which is different from the ILS dictionary learning method. Aharon, Elad and Bruckstein developed the K-SVD algorithm in [10] by generalizing the K-means algorithm for dictionary learning. This algorithm uses a similar block-relaxation approach to MOD, but updates the dictionary on an atom-by-atom basis, without having to compute matrix inversion as required in the original MOD algorithm. The majorization method was proposed by [46] in which the original objective function is substituted by a surrogate function in each step of the optimization process.

In contrast to the generic dictionaries described above, learning structure-oriented parametric dictionaries has also attracted attention. For example, a Gammatone generating function has been used by Yaghoobi *et al.* [47] to learn dictionaries from audio data. In [48], a pyramidal wavelet-like transform was proposed to learn a multiscale structure in the dictionary. Other constraints have also been considered in the learning process to favor the desired structures of the dictionaries, such as the translation-invariant or shift-invariant characteristics of the atoms imposed in [49], [50], [51], [52], [53] and the orthogonality between subspaces enforced in [54], and the de-correlation between the atoms promoted in [55]. An advantage of a parametric dictionary lies in its potential for reducing the number of free parameters, thereby leading to a more efficient implementation and better convergence of dictionary learning algorithms [43]. Other recent efforts in dictionary learning include the search for robust and computationally efficient algorithms, such as [56], [57], and [11], and learning dictionaries from multimodal data [58], [59]. Comprehensive reviews of dictionary learning algorithms can be found in recent survey papers e.g. [43] and [60].

In this paper, similar to MOD and K-SVD methods, we focus on the dictionary update step for

generic dictionary learning. A novel optimization framework is proposed, where an *arbitrary* subset of the codewords are allowed to be updated simultaneously, hence the term *simultaneous codeword optimization (SimCO)*. The proposed framework has the following characteristics.

- SimCO generalizes MOD and K-SVD. We show that the MOD algorithm is in fact an inexact Newton method under the proposed framework while K-SVD can be viewed as a special case of SimCO where only one codeword is selected for update at each iteration. The SimCO framework is general and flexible. This paper presents two possible algorithmic implementations: one is based on gradient descent and the other uses a Newton method.
- The proposed optimization framework allows the discovery of the bottleneck of dictionary update. As opposed to traditional formulations, in the SimCO framework, the objective function involves only the dictionary by treating sparse coefficients as a function of the dictionary. In this way, the gradient can be easily computed and analyzed. Surprisingly, against the traditional belief that local minima are the major problem, we empirically discover that singular points are the bottleneck.
- Regularized SimCO is introduced to mitigate the singularity problem. To avoid the singularity problem, an additive regularization term is introduced. The resulting objective function is differentiable. Significant improvement in empirical performance is observed. This, from another angle, verifies that singularity is the bottleneck.

The remainder of the paper is organized as follows. Section II introduces the SimCO optimization framework, with particular emphasis on the motivations for regularized SimCO. Section III discusses the relation of SimCO to MOD and K-SVD, and the possibility of extending MOD and K-SVD to the regularized versions. Section IV provides necessary preliminaries on manifolds and shows that dictionary update can be cast as an optimization problem on manifolds. The algorithmic details on how to apply the first and second order methods to solve the SimCO optimization problem are presented in Section V. In Section VI, we rigorously prove the deep connection between SimCO and K-SVD. Numerical results of SimCO algorithms are presented in Section VII. Finally, the paper is concluded in Section VIII.

II. The Optimization Framework of SimCO

Dictionary learning is a procedure to find an over-complete dictionary that best represents the training signals. More precisely, let $\boldsymbol{Y} \in \mathbb{R}^{m\times n}$ be the training data, where each column of $\boldsymbol{Y}$ corresponds to one training sample. For a given dictionary size $d \in \mathbb{Z}^+$, the optimal dictionary $\boldsymbol{D}^* \in \mathbb{R}^{m\times d}$ is the one that

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.
- $J = J + 1$.

corresponds to

$$\inf_{\boldsymbol{D}\in\mathbb{R}^{m\times d},\, \boldsymbol{X}\in\mathbb{R}^{d\times n}} \|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2 , \tag{1}$$

where $\|\cdot\|_F$ is the Frobenius norm. Here, the i^{th} column of $\boldsymbol{D}$ is often referred to as the i^{th} *codeword* in the dictionary. 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 is ill-posed unless extra constraints are imposed on the dictionary $\boldsymbol{D}$ and the coefficient matrix $\boldsymbol{X}$. 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 consist 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 [61] or greedy algorithms, for example, OMP [25] and SP [26] algorithms.

The focus of this paper is on the dictionary update stage. Instead of directly solving the joint optimization problem in (1), we view the sparse coefficients as a function of the dictionary so that the optimization is only over the dictionary. Furthermore, our framework allows one to simultaneously update an arbitrary subset of codewords and the corresponding coefficients. This characteristic gives rise to the term *simultaneous codeword optimization* (*SimCO*).

In our formulation, we assume that the dictionary matrix $\boldsymbol{D}$ contains unit ℓ_2-norm columns and the sparsity pattern of $\boldsymbol{X}$ remains unchanged. Define

$$\mathcal{D} = \left\{\boldsymbol{D} \in \mathbb{R}^{m\times d} :\ \|\boldsymbol{D}_{:,i}\|_2 = 1,\ \forall i \in [d]\right\}, \tag{2}$$

where $\|\cdot\|_2$ is the ℓ_2-norm and the set $[d] = \{1, 2, \cdots, d\}$. This is the set of all feasible dictionaries. Represent the *sparsity pattern* of $\boldsymbol{X}$ by the index set $\Omega \subset [d] \times [n]$ which contains the indices of all the

non-zero entries in $\boldsymbol{X}$: that is, $X_{i,j} \neq 0$ for all $(i,j) \in \Omega$ and $X_{i,j} = 0$ for all $(i,j) \notin \Omega$. Define

$$\mathcal{X}(\Omega) = \left\{ \boldsymbol{X} \in \mathbb{R}^{d \times n} : \ X_{i,j} = 0, \ \forall (i,j) \notin \Omega \right\}. \tag{3}$$

This is the set of all feasible $\boldsymbol{X}$ given sparsity pattern Ω. The dictionary update problem is formulated as

$$\inf_{\boldsymbol{D} \in \mathcal{D}} f(\boldsymbol{D}) = \inf_{\boldsymbol{D} \in \mathcal{D}} \underbrace{\inf_{\boldsymbol{X} \in \mathcal{X}(\Omega)} \|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2}_{f(\boldsymbol{D})}. \tag{4}$$

To evaluate $f(\boldsymbol{D})$ for a given $\boldsymbol{D}$, one needs to solve the least squares problem $\inf_{\boldsymbol{X} \in \mathcal{X}(\Omega)} \|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2$. Denote the optimal $\boldsymbol{X}$ by $\boldsymbol{X}(\boldsymbol{D})$, which can be viewed as a function of $\boldsymbol{D}$. An update in $\boldsymbol{D}$ results in an update of $\boldsymbol{X}(\boldsymbol{D})$. In other words, both $\boldsymbol{D}$ and $\boldsymbol{X}$ are simultaneously updated.

One may notice that the optimal $\boldsymbol{X}$ that solves the least squares problem $\inf_{\boldsymbol{X} \in \mathcal{X}(\Omega)} \|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2$ may not be unique. Non-unique solutions happen only when $\boldsymbol{D}$ is singular, formally defined as follows. For a given sparsity patten Ω, let $\Omega(:,j) = \{i : \ (i,j) \in \Omega\}$. Let $\boldsymbol{D}_{:,\Omega(:,j)}$ be the sub-matrix of $\boldsymbol{D}$ containing the columns indexed by $\Omega(:,j)$. A dictionary $\boldsymbol{D}$ is *singular* under sparsity patten Ω if there exists $j \in [n]$ such that the columns of $\boldsymbol{D}_{:,\Omega(:,j)}$ are linearly dependent, i.e., $\boldsymbol{D}_{:,\Omega(:,j)}$ does not have full column rank. At a singular point, $\boldsymbol{X}(\boldsymbol{D})$ is not uniquely defined. This can be solved by arbitrarily choosing one of the multiple solutions as the choice of $\boldsymbol{X}(\boldsymbol{D})$ does not affect the value of $f(\boldsymbol{D})$.

The singularity problem brings several algorithmic problems.

1) Severe performance deterioration in dictionary update. Our empirical experiments (detailed in Section VII-A) show that, when the dictionary update procedure fails in finding a globally optimal solution, most likely it converges to a singular point, i.e., an ill-conditioned dictionary.
2) Slow convergence in dictionary update. Let $\lambda_{\min}\left(\boldsymbol{D}_{:,\Omega(:,j)}\right)$ be the minimum singular value of the matrix $\boldsymbol{D}_{:,\Omega(:,j)}$. When it is close to zero, the curvature (Hessian) of $f(\boldsymbol{D})$ is large and the gradient changes significantly in the neighborhood of a singular point. Optimization algorithms typically suffer from a very slow convergence rate.
3) Instability in the subsequent sparse coding stage. When $\lambda_{\min}\left(\boldsymbol{D}_{:,\Omega(:,j)}\right)$ is close to zero, the solution to the least squares problem $\inf_{\boldsymbol{X}_{\Omega(:,j),j}} \left\|\boldsymbol{Y}_{:,j} - \boldsymbol{D}_{:,\Omega(:,j)}\boldsymbol{X}_{\Omega(:,j),j}\right\|_F^2$ becomes unstable: small changes in $\boldsymbol{Y}_{:,j}$ often result in very different least squares 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) [61], which requires that the singular values of submatrices of $\boldsymbol{D}$ center around 1. An ill-conditioned $\boldsymbol{D}$ violates RIP and

hence results in sparse coefficients that are sensitive to noise.

To mitigate the singularity problem, we propose to add a regularization term into the objective function:

$$\inf_{\boldsymbol{D}\in\mathcal{D}} f_\mu(\boldsymbol{D}) = \inf_{\boldsymbol{D}\in\mathcal{D}} \underbrace{\inf_{\boldsymbol{X}\in\mathcal{X}(\Omega)} \|\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X}\|_F^2 + \mu\|\boldsymbol{X}\|_F^2}_{f_\mu(\boldsymbol{D})}, \tag{5}$$

where $\mu > 0$ is a properly chosen constant. Hereafter, we refer to (4) and (5) as *primitive SimCO* and *regularized SimCO* respectively. Note that when $\mu = 0$, regularized SimCO reduces to primitive SimCO. In practice, one may consider first using regularized SimCO ($\mu > 0$) to obtain a reasonably good dictionary and then reduce the regularization constant μ to zero to refine the dictionary further. This two-step procedure often results in a well-conditioned dictionary that fits the training data. See the simulation part (Section VII) for examples.

The effect of the regularization term is to remove the singular point. Let $\boldsymbol{Y}_{:,j}$ be the j^{th} column of $\boldsymbol{Y}$. Let $\boldsymbol{X}_{\Omega(:,j),j}$ be the sub-vector of $\boldsymbol{X}_{:,j}$ formed by the entries indexed by $\Omega(:,j)$. Let $m_j = |\Omega(:,j)|$ be the number of non-zeros in the j^{th} column of $\boldsymbol{X}$. Define

$$\tilde{\boldsymbol{y}}_j = \begin{bmatrix} \boldsymbol{Y}_{:,j} \\ \boldsymbol{0}_{m_j} \end{bmatrix},\ \boldsymbol{D}_j = \boldsymbol{D}_{:,\Omega(:,j)},\ \text{and}\ \tilde{\boldsymbol{D}}_j = \begin{bmatrix} \boldsymbol{D}_{:,\Omega(:,j)} \\ \sqrt{\mu}\cdot\boldsymbol{I}_{m_j} \end{bmatrix}, \tag{6}$$

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. Then

$$\begin{aligned} f_\mu(\boldsymbol{D}) &= \inf_{\boldsymbol{X}\in\mathcal{X}(\Omega)} \|\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X}\|_F^2 + \mu\|\boldsymbol{X}\|_F^2 \\ &= \sum_{j=1}^{n} \inf_{\boldsymbol{X}_{\Omega(:,j),j}} \left\|\boldsymbol{Y}_{:,j} - \boldsymbol{D}_{:,\Omega(:,j)}\boldsymbol{X}_{\Omega(:,j),j}\right\|_2^2 + \mu\left\|\boldsymbol{X}_{\Omega(:,j),j}\right\|_2^2 \\ &= \sum_{j=1}^{n} \underbrace{\inf_{\boldsymbol{X}_{\Omega(:,j),j}} \left\|\tilde{\boldsymbol{y}}_j - \tilde{\boldsymbol{D}}_j\boldsymbol{X}_{\Omega(:,j),j}\right\|_2^2}_{f_{\mu,j}(\boldsymbol{D}_j),\ \text{or equivalently}\ f_{\mu,j}\left(\tilde{\boldsymbol{D}}_j\right)}. \end{aligned} \tag{7}$$

When $\mu > 0$, the matrix $\tilde{\boldsymbol{D}}_j$ in the j^{th} atomic function $f_{\mu,j}\left(\tilde{\boldsymbol{D}}_j\right)$ has full column rank. The objective function $f_\mu(\boldsymbol{D})$ is always continuous and contains no singular points. The algorithmic details for solving the regularized SimCO are presented in Section V.

So far, we have considered only the case where all the codewords and the corresponding nonzero coefficients are simultaneously updated. It is worth noting that SimCO accommodates the case of simultaneously updating an arbitrary subset of codewords and the corresponding nonzero coefficients. More precisely, let

$\mathcal{I} \subseteq [d]$ be the index set of the codewords to be updated. That is, only codewords $\boldsymbol{D}_{:,i}$'s, $i \in \mathcal{I}$, are to be updated while all other codewords $\boldsymbol{D}_{:,i}$'s, $i \notin \mathcal{I}$, 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 $\mathcal{I}$. 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 $\mathcal{I}$. Then $\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X} = \boldsymbol{Y}_r - \boldsymbol{D}_{:,\mathcal{I}}\boldsymbol{X}_{\mathcal{I},:}$. Replacing the $\boldsymbol{Y}$, $\boldsymbol{D}$ and $\boldsymbol{X}$ in (4) and (5) by $\boldsymbol{Y}_r$, $\boldsymbol{D}_{:,\mathcal{I}}$ and $\boldsymbol{X}_{\mathcal{I},:}$ respectively, the optimization framework developed for the full set $[d]$, i.e., (4) and (5), can be readily applied to the case $\mathcal{I} \subset [d]$. For this reason, the discussions hereafter will center around the full set $[d]$ case (the subscript $\mathcal{I}$ will be dropped).

Finally, we would like to comment on the column-norm constraint imposed on the dictionary in (2). This constraint appears in K-SVD but not in MOD. Theoretically, the performance of a given dictionary is invariant to the column norms: a scaling in columns of $\boldsymbol{D}$ can be compensated by an inverse scaling in the corresponding rows of $\boldsymbol{X}$. On the other hand, the constraint on the column norms has certain advantages:

1) A normalized dictionary $\boldsymbol{D} \in \mathcal{D}$ is required in regularized SimCO. The regularization term $\mu \|\boldsymbol{X}\|_F^2$ is useful only when the column norms of $\boldsymbol{D}$ are fixed. Otherwise, the regularized objective function (5) can be reduced simply by scaling up the columns of $\boldsymbol{D}$.
2) A normalized dictionary $\boldsymbol{D} \in \mathcal{D}$ plays an important role in identifying singular points. As detailed in Section VII-A, the gradient of the objective function $f(\boldsymbol{D})$ is used to distinguish between singular points and local minimizers. Since scaling the columns of the dictionary results in scaling in the gradient (see (10) for more details), a normalization is necessary.
3) A normalized dictionary $\boldsymbol{D} \in \mathcal{D}$ is preferred in the sparse coding stage. Sparse coding algorithms rely heavily on the magnitudes of the coefficients $X_{i,j}$'s, $(i,j) \in [d] \times [n]$, which are affected by the column norms of $\boldsymbol{D}$. It is a standard practice to normalize the columns of $\boldsymbol{D}$ before applying sparse coding algorithms.

III. Relation to the State of the Art

In this section, we discuss how primitive SimCO is related to two benchmark algorithms MOD and K-SVD. Furthermore, as regularization substantially improves the performance (motivated in Section II and empirically demonstrated in Section VII), we regularize MOD and K-SVD as well. Here, we would

like to emphasize that the regularization technique is designed to handle the singularity problem, which is observed via the SimCO framework. The authors are not aware of regularized versions of MOD and K-SVD in the literature.

In MOD, the dictionary update involves iteratively performing two steps: first fix $\boldsymbol{D}$ and solve $\boldsymbol{X}$ for $\inf_{\boldsymbol{X}\in\mathcal{X}(\Omega)} \|\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X}\|_F^2$; then fix $\boldsymbol{X}$ and solve $\boldsymbol{D}$ for $\inf_{\boldsymbol{D}\in\mathbb{R}^{m\times d}} \|\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X}\|_F^2$. Both steps involve only solving a least squares problem. Denote the dictionaries before and after an iteration by $\boldsymbol{D}$ and $\boldsymbol{D}'$ respectively. Then the updated sparse coefficients are $\boldsymbol{X}(\boldsymbol{D})$ and the updated dictionary is given by $\boldsymbol{D}'=\boldsymbol{Y}\boldsymbol{X}^T(\boldsymbol{D})\left(\boldsymbol{X}(\boldsymbol{D})\boldsymbol{X}^T(\boldsymbol{D})\right)^{-1}$.

MOD can be viewed as an inexact Newton method to solve the primitive SimCO problem without the column norm constraint. To see it, after dropping the column-norm constraint, the optimization problem in SimCO becomes $\inf_{\boldsymbol{D}\in\mathbb{R}^{m\times d}} f(\boldsymbol{D})$ where $f(\boldsymbol{D})=\inf_{\boldsymbol{X}\in\mathcal{X}(\Omega)} \|\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X}\|_F^2$. Consider the Newton iteration for dictionary update, where the gradient and the Hessian are given by $\nabla f(\boldsymbol{D}) = -2\left(\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X}(\boldsymbol{D})\right)\boldsymbol{X}^T(\boldsymbol{D})$ and

$$\begin{aligned}\nabla^2 f(\boldsymbol{D}) = {} & 2\left(\nabla\boldsymbol{D}\right)\boldsymbol{X}(\boldsymbol{D})\boldsymbol{X}^T(\boldsymbol{D}) \\ & + 2\boldsymbol{D}\left(\nabla\boldsymbol{X}(\boldsymbol{D})\right)\boldsymbol{X}^T(\boldsymbol{D}) + 2\boldsymbol{D}\boldsymbol{X}(\boldsymbol{D})\left(\nabla\boldsymbol{X}(\boldsymbol{D})\right)^T\end{aligned}$$

respectively (see Section V-B for more details on how to compute Hessian). Note that the computation of $\nabla\boldsymbol{X}(\boldsymbol{D})$ is complicated. To reduce the computational complexity, one may approximate $\nabla^2 f$ by omitting the terms involving $\nabla\boldsymbol{X}(\boldsymbol{D})$, i.e., approximate $\nabla^2 f$ by $2\left(\nabla\boldsymbol{D}\right)\boldsymbol{X}(\boldsymbol{D})\boldsymbol{X}^T(\boldsymbol{D})$. Following from this approximation, the objective function at the neighborhood of a given dictionary $\boldsymbol{D}_0$ can be approximated by $f(\boldsymbol{D})\approx\|\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X}(\boldsymbol{D}_0)\|_F^2$. The optimal dictionary with respect to the approximated objective function is then given by $\boldsymbol{D}'=\boldsymbol{Y}\boldsymbol{X}^T(\boldsymbol{D}_0)\left(\boldsymbol{X}(\boldsymbol{D}_0)\boldsymbol{X}^T(\boldsymbol{D}_0)\right)^{-1}$, which coincides with the update rule in MOD.

Using a similar approximation for the corresponding Hessian matrix, MOD can be adapted to solve the regularized SimCO problem. We refer to it as *regularized MOD*. It again iteratively performs two steps: first fix $\boldsymbol{D}$ and solve $\boldsymbol{X}$ for $\inf_{\boldsymbol{X}\in\mathcal{X}(\Omega)} \|\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X}\|_F^2+\mu\|\boldsymbol{X}\|_F^2$; then fix $\boldsymbol{X}$ and solve $\boldsymbol{D}$ for $\inf_{\boldsymbol{D}\in\mathbb{R}^{m\times d}} \|\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X}\|_F^2$. Substantial improvement in empirical performance can be observed in Section VII.

The relation between SimCO and K-SVD is straightforward. Consider the SimCO where only one codeword and the corresponding sparse coefficients are updated. The resulting objective function is the

same as that of K-SVD. More specifically, suppose that only the i^{th} codeword and the corresponding sparse coefficients are updated. Let $\boldsymbol{Y}_r = \boldsymbol{Y} - \boldsymbol{D}_{:,\{i\}^c}\boldsymbol{X}_{\{i\}^c,:}$. Then both SimCO and K-SVD aim at solving

$$\inf_{\boldsymbol{D}_{:,i}:\, \|\boldsymbol{D}_{:,i}\|_2=1} \inf_{\boldsymbol{X}_{i,\Omega(i,:)}} \left\|(\boldsymbol{Y}_r)_{:,\Omega(i,:)} - \boldsymbol{D}_{:,i}\boldsymbol{X}_{i,\Omega(i,:)}\right\|_2^2.$$

In K-SVD, singular value decomposition (SVD) is used to solve the above optimization problem.

Nevertheless, it is not clear how to extend K-SVD to the regularized case. Let $m_i = |\Omega(i,:)|$. With the regularization term, the optimization problem becomes

$$\begin{aligned}
&\inf_{\boldsymbol{D}_{:,i}:\, \|\boldsymbol{D}_{:,i}\|_2=1} \inf_{\boldsymbol{X}_{i,\Omega(i,:)}\in\mathbb{R}^{m_i}} \left\|(\boldsymbol{Y}_r)_{:,\Omega(i,:)} - \boldsymbol{D}_{:,i}\boldsymbol{X}_{i,\Omega(i,:)}\right\|_2^2 + \mu\left\|\boldsymbol{X}_{i,\Omega(i,:)}\right\|_2^2 \\
&= \inf_{\boldsymbol{d}:\, \|\boldsymbol{d}\|_2=1} \inf_{\boldsymbol{x}\in\mathbb{R}^{m_i}} \left\| \underbrace{\begin{bmatrix} (\boldsymbol{Y}_r)_{:,\Omega(i,:)} \\ \boldsymbol{O}_{m_i} \end{bmatrix}}_{\tilde{\boldsymbol{Y}}_{r,i}} - \underbrace{\begin{bmatrix} \boldsymbol{d} \\ \sqrt{\mu}\mathbf{1}_{m_i} \end{bmatrix}}_{\tilde{\boldsymbol{d}}} \boldsymbol{x}^T \right\|_2^2,
\end{aligned}$$

where $\boldsymbol{O}_{m_i}$ is the $m_i \times m_i$ zero matrix, $\mathbf{1}_{m_i} \in \mathbb{R}^{m_i}$ is an all-one vector, and we have simplified the notations $\boldsymbol{D}_{:,i}$ and $\boldsymbol{X}_{i,\Omega(i,:)}$ to $\boldsymbol{d}$ and $\boldsymbol{x}$ respectively. If we apply SVD to $(\boldsymbol{Y}_r)_{:,\Omega(i,:)}$, the solution is exactly identical to that in the original K-SVD. If we apply SVD to $\tilde{\boldsymbol{Y}}_{r,i}$, the left singular vectors are not of the form $\tilde{\boldsymbol{d}}$: the last m_i entries of the left singular vectors are always zero. In either case, in contrast to the original K-SVD, SVD cannot solve the joint optimization problem. In the numerical comparison part of this paper, we use regularized SimCO with $\mathcal{I} = \{i\} \subset [d]$ to solve this optimization problem, and refer to the resulting algorithm as regularized "K-SVD" although it involves no SVD.

IV. Preliminaries on Manifolds

Our approach for solving the optimization problem (4) and (5) relies on the notion of 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 $\mathcal{G}_{m,1} = \left\{\text{span}\,(\boldsymbol{u}) :\ \boldsymbol{u} \in \mathcal{U}_{m,1}\right\}$. Here, the notations $\mathcal{U}_{m,1}$ and $\mathcal{G}_{m,1}$ follow from the convention in [62], [63]. 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 $\mathscr{U} = \text{span}\,(\boldsymbol{u})$, then $\mathscr{U} = \text{span}\,(-\boldsymbol{u})$ 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 span $(\boldsymbol{D}_{:,i})$. As shall become explicit later, this phenomenon has a significant impact on algorithm design and analysis.

It is worth noting that the performance of a given dictionary is invariant to the permutations of the codewords. However, how to effectively address this permutation invariance analytically and algorithmically remains an open problem.

V. Implementation Details for SimCO

This section presents the algorithmic details on how to solve the optimization problems (4) and (5). As the primitive SimCO is a special case of regularized SimCO where $\mu = 0$, the descriptions below center around regularized SimCO. One of the key properties of SimCO is that the objective function $f_\mu(\boldsymbol{D})$ only involves the dictionary. To minimize this objective function, derivatives of this function need to be evaluated. First and second order optimization procedures can be implemented. Note that first order methods are often conceptually easier to understand but slower in convergence rate, while second order methods are typically faster in convergence rate but more complicated in the computation of the search direction. In this section, we first outline the proposed algorithms, then give details on the computations of the first and second order derivatives, and finally discuss the line search path that satisfies the column norm constraint.

A. Outline of Algorithms

A natural choice of first order optimization procedures is the gradient descent line search method. Algorithm 2 summarizes one iteration of the proposed procedure. The computational details of the gradient ∇f_μ and line search path $\boldsymbol{D}(t)$ are presented in Sections V-B and V-C respectively, where t is a step size. For proof-of-concept, we use the method of golden section search[1] (see [65] for a detailed description).

[1]Algorithm 2 looks more complicated than popular gradient descent methods in standard textbooks, e.g., [64]. We choose this implementation because it mimics the ideal gradient descent with infinitesimal steps more authentically than other optimization methods of which the step size may be so large that local minimizers or singular point may not be seen. In the simulation part, we use Algorithm 2 to catch the singular points.

Algorithm 2 One iteration in a gradient descent line search algorithm.

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 $f_\mu(\boldsymbol{D})$ and $\nabla f_\mu(\boldsymbol{D})$. If $\|\nabla f_\mu\|_F \leq g_{\min}\|\boldsymbol{Y}\|_F^2$, then $\boldsymbol{D}' = \boldsymbol{D}$, $\boldsymbol{X}' = \boldsymbol{X}$, and quit.
2) Set line search direction $\boldsymbol{H} = -\nabla f_\mu$. Let $t_3 = ct_4$ and $t_2 = (1-c)\,t_4$.

Part A: the goal is to find $t_4 > 0$ s.t. $f(\boldsymbol{D}(t_1)) > f(\boldsymbol{D}(t_2)) > f(\boldsymbol{D}(t_3)) \leq f(\boldsymbol{D}(t_4))$, where $\boldsymbol{D}(t)$ is defined via (15). 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 keeping $f(\boldsymbol{D}(t_1)) > f(\boldsymbol{D}(t_2)) > f(\boldsymbol{D}(t_3))$. Iterate the following steps until $t_4 - t_1$ is sufficiently small.

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 $t^* = \underset{t\in\{t_1,t_2,t_3,t_4\}}{\arg\min} f(\boldsymbol{D}(t))$. Set $\boldsymbol{D}' = \boldsymbol{D}(t^*)$ and compute $\boldsymbol{X}'$ according to (9).

The idea is to use the golden ratio to successively narrow the search range of t inside which a local minimum exists. To implement this idea, we design a two-step procedure in Algorithm 2: in the first step (Part A), we increase/decrease the range of t, i.e., $(0, t_4)$, so that it contains a local minimum and the objective function looks unimodal in this range; in the second step (Part B), we use the golden ratio to narrow the range so that we can accurately locate the minimizer. Note that the proposed algorithm is by no means optimal. Other ways to do a gradient descent efficiently can be found in [64, Chapter 3].

For second order optimization methods, we choose line search Newton-CG (LSNCG) method [64, Ch. 7]. (It turns out that the trust region method [64], another popular second order optimization method, is not quite numerically stable under certain conditions.) It is worth noting that LSNCG, which uses the exact Hessian, typically exhibits faster convergence rate than MOD, where an approximate of the Hessian is employed.

Before discussing the details, let us first understand the ideas behind LSNCG. Let $\boldsymbol{D}$ and $\boldsymbol{D}'$ be the dictionaries before and after a line search step. In Newton methods, $\boldsymbol{D}' - \boldsymbol{D} = -\left(\nabla^2 f_\mu\right)^{-1}\nabla f_\mu$. However, note that the Hessian $\nabla^2 f_\mu$ is an $(m\times d)\times(m\times d)$ matrix with entries $\partial^2 f_\mu / \left(\partial \boldsymbol{D}_{i,j}\partial \boldsymbol{D}_{k,\ell}\right)$. Computing it explicitly and taking the inverse are computationally expensive. The main idea of LSNCG method is to use the conjugate gradient method [64, Ch. 5] to avoid explicit computation of the Hessian and its

Algorithm 3 One iteration in the LSNCG algorithm

Input: $\boldsymbol{Y}$, $\boldsymbol{D}$, $\boldsymbol{X}$

Output: $\boldsymbol{D}'$ and $\boldsymbol{X}'$.

Initialization: Set $\boldsymbol{Z}^{(0)} = \boldsymbol{0} \in \mathbb{R}^{m\times d}$, $\boldsymbol{R}^{(0)} = \nabla f_\mu$, $\boldsymbol{H}^{(0)} = -\boldsymbol{R}^{(0)}$, and $J = 0$. Define tolerance $\epsilon = \min\left(0.5, \sqrt{\|\nabla f_\mu\|_F}\right)\|\nabla f_\mu\|_F$. Define shrink constant $\rho \in (0,1)$.

Part A: the goal is to find the Newton direction $\boldsymbol{H}$ using conjugate gradient method. Perform the following iterations.

1) If $\langle \nabla f_\mu, \nabla_{\boldsymbol{H}^{(J)}} \nabla f_\mu \rangle \geq 0$, set $\boldsymbol{H} = \boldsymbol{H}^{(J)}$ and quite the iterations.
2) Set $\alpha^{(J)} = \left\langle \boldsymbol{R}^{(J)}, \boldsymbol{R}^{(J)} \right\rangle / \left\langle \boldsymbol{H}^{(J)}, \nabla_{\boldsymbol{H}^{(J)}} \nabla f_\mu \right\rangle$, $\boldsymbol{Z}^{(J+1)} = \boldsymbol{Z}^{(J)} + \alpha^{(J)} \boldsymbol{H}^{(J)}$, and $\boldsymbol{R}^{(J+1)} = \boldsymbol{R}^{(J)} + \alpha^{(J)} \nabla_{\boldsymbol{H}^{(J)}} \nabla f_\mu$.
3) If $\left\|\boldsymbol{R}^{(J+1)}\right\|_F < \epsilon$, set $\boldsymbol{H} = \boldsymbol{Z}^{(J+1)}$ and quit the iterations.
4) Set $\beta^{(J+1)} = \left\langle \boldsymbol{R}^{(J+1)}, \boldsymbol{R}^{(J+1)} \right\rangle / \left\langle \boldsymbol{R}^{(J)}, \boldsymbol{R}^{(J)} \right\rangle$, $\boldsymbol{H}^{(J+1)} = -\boldsymbol{R}^{(J+1)} + \beta^{(J+1)} \boldsymbol{H}^{(J)}$, and then $J = J+1$.

Part B: Line search along $\boldsymbol{H}$.

5) Start with $t = 1$ and repeat setting $t = \rho t$ until $f_\mu(\boldsymbol{D}(t)) < f_\mu(\boldsymbol{D})$. Set $t^* = t$, $\boldsymbol{D}' = \boldsymbol{D}(t^*)$ and compute $\boldsymbol{X}'$ according to (9).

inverse. The steps of LSNCG are based on the concept of directional derivative. Let $\boldsymbol{H} \in \mathbb{R}^{m\times r}$ be a matrix of the same dimension of $\boldsymbol{D}$. The directional derivative of f_μ along direction $\boldsymbol{H}$ is defined as

$$\nabla_{\boldsymbol{H}} f_\mu(\boldsymbol{D}) = \lim_{\tau\to 0} \frac{f_\mu(\boldsymbol{D} + \boldsymbol{H}\tau) - f_\mu(\boldsymbol{D})}{\tau}.$$

Instead of computing the Hessian, LSNCG only involves directional derivative of the gradient, i.e.,

$$\nabla_{\boldsymbol{H}} \nabla f_\mu(\boldsymbol{D}) = \lim_{\tau\to 0} \frac{\nabla f_\mu(\boldsymbol{D} + \boldsymbol{H}\tau) - \nabla f_\mu(\boldsymbol{D})}{\tau}.$$

Note that both ∇f_μ and $\nabla_{\boldsymbol{H}} \nabla f_\mu$ are $m \times d$ matrices[2] and admit closed forms (computation details are given in Section V-B). The computational complexity is greatly reduced. Algorithm 3 summarizes one iteration of the LSNCG procedure for dictionary update, where $\langle \boldsymbol{A}, \boldsymbol{B} \rangle$ represents the inner product of matrices $\boldsymbol{A}$ and $\boldsymbol{B}$.

B. Computation of the First and Second Order Derivatives

We now compute ∇f_μ and $\nabla_{\boldsymbol{H}} \nabla f_\mu$. From the decomposition $f_\mu = \sum_j f_{\mu,j}$ derived in (7), it is clear that

$$\nabla f_\mu = \sum_j \nabla f_{\mu,j}. \tag{8}$$

[2]The entries of $\nabla f_\mu \in \mathbb{R}^{m\times d}$ are $\partial f_\mu / \partial \boldsymbol{D}_{i,j}$ and those of $\nabla_{\boldsymbol{H}} \nabla f_\mu$ are $\nabla_{\boldsymbol{H}} (\partial f_\mu / \partial \boldsymbol{D}_{i,j})$.

For any given $\boldsymbol{H} \in \mathbb{R}^{m\times d}$, define $\boldsymbol{H}_j = \boldsymbol{H}_{:,\Omega(:,j)}$. It can be verified that

$$\begin{aligned}\nabla_{\boldsymbol{H}}\nabla f_\mu &= \nabla_{\boldsymbol{H}}\left(\sum_j \nabla f_{\mu,j}\right)\\ &= \sum_j \nabla_{\boldsymbol{H}}\nabla f_{\mu,j} = \sum_j \nabla_{\boldsymbol{H}_j}\nabla f_{\mu,j}.\end{aligned}$$

As a result, it suffices to compute $\nabla f_{\mu,j}$ and $\nabla_{\boldsymbol{H}_j}\nabla f_{\mu,j}$ for each atomic function.

The j^{th} atomic function for regularized SimCO, defined in (7), is of the form $f_{\mu,j} = \left\|\tilde{\boldsymbol{y}}_j - \tilde{\boldsymbol{D}}_j\boldsymbol{x}_j\left(\tilde{\boldsymbol{D}}_j\right)\right\|_2^2$ where

$$\boldsymbol{x}_j\left(\tilde{\boldsymbol{D}}_j\right) = \tilde{\boldsymbol{D}}_j^{\dagger}\tilde{\boldsymbol{y}}_j = \left(\tilde{\boldsymbol{D}}_j^T\tilde{\boldsymbol{D}}_j\right)^{-1}\tilde{\boldsymbol{D}}_j^T\tilde{\boldsymbol{y}}_j. \tag{9}$$

Again, let $m_j = |\Omega(:,j)|$. Then $\tilde{\boldsymbol{D}}_j \in \mathbb{R}^{(m+m_j)\times m_j}$. Note that, $f_{\mu,j}$ can be regarded as a function of either $\tilde{\boldsymbol{D}}_j$ or $\boldsymbol{D}_j$. We first compute $\nabla f_{\mu,j}\left(\tilde{\boldsymbol{D}}_j\right) \in \mathbb{R}^{(m+m_j)\times m_j}$, i.e., the gradient of $f_{\mu,j}$ with respect to $\tilde{\boldsymbol{D}}_j$, and then obtain $\nabla f_{\mu,j}(\boldsymbol{D}_j) \in \mathbb{R}^{m\times m_j}$, i.e., the gradient of $f_{\mu,j}$ with respect to $\boldsymbol{D}_j$ from $\nabla f_{\mu,j}\left(\tilde{\boldsymbol{D}}_j\right)$. The gradient with respect to $\tilde{\boldsymbol{D}}_j$ is given by[3]

$$\begin{aligned}\nabla f_{\mu,j}(\tilde{\boldsymbol{D}}_j) &= \left.\frac{\partial f_{\mu,j}}{\partial\tilde{\boldsymbol{D}}_j}\right|_{\boldsymbol{x}_j\left(\tilde{\boldsymbol{D}}_j\right)} + \left.\frac{\partial f_{\mu,j}}{\partial\boldsymbol{x}_j}\right|_{\tilde{\boldsymbol{D}}_j}\cdot\frac{\partial\boldsymbol{x}_j}{\partial\tilde{\boldsymbol{D}}_j}\\ &= -2\left(\tilde{\boldsymbol{y}}_j - \tilde{\boldsymbol{D}}_j\boldsymbol{x}_j\right)\boldsymbol{x}_j^T - 2\tilde{\boldsymbol{D}}_j^T\left(\tilde{\boldsymbol{y}}_j - \tilde{\boldsymbol{D}}_j\boldsymbol{x}_j\right)\cdot\frac{\partial\boldsymbol{x}_j}{\partial\tilde{\boldsymbol{D}}_j}.\end{aligned}$$

Note that $\tilde{\boldsymbol{y}}_j - \tilde{\boldsymbol{D}}_j\boldsymbol{x}_j = \tilde{\boldsymbol{y}}_j - \tilde{\boldsymbol{D}}_j\tilde{\boldsymbol{D}}_j^{\dagger}\tilde{\boldsymbol{y}}_j$ is orthogonal to the columns of $\tilde{\boldsymbol{D}}_j$. One has $\tilde{\boldsymbol{D}}_j^T\left(\tilde{\boldsymbol{y}}_j - \tilde{\boldsymbol{D}}_j\boldsymbol{x}_j\right) = \boldsymbol{0}$. As a result,

$$\nabla f_{\mu,j}(\tilde{\boldsymbol{D}}_j) = -2\left(\tilde{\boldsymbol{y}}_j - \tilde{\boldsymbol{D}}_j\boldsymbol{x}_j\right)\boldsymbol{x}_j^T. \tag{10}$$

From the definition of $\tilde{\boldsymbol{D}}_j$ in (6), $\boldsymbol{D}_j$ is a sub-matrix of $\tilde{\boldsymbol{D}}_j$, and therefore $\nabla f_{\mu,j}(\boldsymbol{D}_j)$ is also a sub-matrix of $\nabla f_{\mu,j}(\tilde{\boldsymbol{D}}_j)$, i.e., $\nabla f_{\mu,j}(\boldsymbol{D}_j) = \left(\nabla f_{\mu,j}(\tilde{\boldsymbol{D}}_j)\right)_{1:m,1:m_j}$.

A similar procedure is used to compute the second order derivative $\nabla_{\boldsymbol{H}_j}\nabla f_\mu(\boldsymbol{D}_j)$. For a given $\boldsymbol{H}_j \in \mathbb{R}^{m\times m_j}$, define $\tilde{\boldsymbol{H}}_j = \left[\boldsymbol{H}_j^T, \boldsymbol{O}_{m_j}\right]^T \in \mathbb{R}^{(m+m_j)\times m_j}$, where $\boldsymbol{O}_{m_j}$ is the $m_j\times m_j$ zero matrix. By the definition of $\tilde{\boldsymbol{D}}_j$ and directional derivative, it can be verified that $\nabla_{\boldsymbol{H}_j}\nabla f_\mu(\boldsymbol{D}_j) = \left(\nabla_{\tilde{\boldsymbol{H}}_j}\nabla f_\mu\left(\tilde{\boldsymbol{D}}_j\right)\right)_{1:m,1:m_j}$. We

[3]Note that the term $\left.\frac{\partial f_{\mu,j}}{\partial\boldsymbol{x}_j}\right|_{\tilde{\boldsymbol{D}}_j}\cdot\frac{\partial\boldsymbol{x}_j}{\partial\tilde{\boldsymbol{D}}_j}$ is a product of a vector and a tensor defined as $\sum_k\left(\left.\frac{\partial f_{\mu,j}}{\partial(\boldsymbol{x}_j)_k}\right|_{\tilde{\boldsymbol{D}}_j}\right)\cdot\frac{\partial(\boldsymbol{x}_j)_k}{\partial\tilde{\boldsymbol{D}}_j}$, where $(\boldsymbol{x}_j)_k$ is the k^{th} element of the vector $\boldsymbol{x}_j$.

compute $\nabla_{\tilde{\boldsymbol{H}}_j}\nabla f_\mu\left(\tilde{\boldsymbol{D}}_j\right)$ as follows. For any $\tilde{\boldsymbol{H}}_j$,

$$\begin{aligned}&\frac{1}{2}\nabla_{\tilde{\boldsymbol{H}}_j}\nabla f_\mu\left(\tilde{\boldsymbol{D}}_j\right)\\&=\left(\nabla_{\tilde{\boldsymbol{H}}_j}\tilde{\boldsymbol{D}}_j\right)\boldsymbol{x}_j\boldsymbol{x}_j^T+\tilde{\boldsymbol{D}}_j\left(\nabla_{\tilde{\boldsymbol{H}}_j}\boldsymbol{x}_j\right)\boldsymbol{x}_j^T-\left(\tilde{\boldsymbol{y}}_j-\tilde{\boldsymbol{D}}_j\boldsymbol{x}_j\right)\left(\nabla_{\tilde{\boldsymbol{H}}_j}\boldsymbol{x}_j\right)^T.\end{aligned}\tag{11}$$

It is clear that $\nabla_{\tilde{\boldsymbol{H}}_j}\tilde{\boldsymbol{D}}_j=\tilde{\boldsymbol{H}}_j$. To compute the other term $\nabla_{\tilde{\boldsymbol{H}}_j}\boldsymbol{x}_j$, note that $\boldsymbol{x}_j$ is a function of $\tilde{\boldsymbol{D}}_j$ involving matrix inversion. To proceed, we use the fact that for any invertible matrix $\boldsymbol{A}$, it holds $\nabla\boldsymbol{A}^{-1}=-\boldsymbol{A}^{-1}\left(\nabla\boldsymbol{A}\right)\boldsymbol{A}^{-1}$ (derived by differentiating both sides of $\boldsymbol{A}\cdot\boldsymbol{A}^{-1}=\boldsymbol{I}$). As a result, one has $\nabla_{\tilde{\boldsymbol{H}}_j}\boldsymbol{x}_j=\nabla_{\tilde{\boldsymbol{H}}_j}\tilde{\boldsymbol{D}}_j^\dagger\tilde{\boldsymbol{y}}_j$ where $\nabla_{\tilde{\boldsymbol{H}}_j}\tilde{\boldsymbol{D}}_j^\dagger$ is given by

$$\begin{aligned}\nabla_{\tilde{\boldsymbol{H}}_j}\tilde{\boldsymbol{D}}_j^\dagger&=\nabla_{\tilde{\boldsymbol{H}}_j}\left(\left(\tilde{\boldsymbol{D}}_j^T\tilde{\boldsymbol{D}}_j\right)^{-1}\tilde{\boldsymbol{D}}_j^T\right)\\&=-\left(\tilde{\boldsymbol{D}}_j^T\tilde{\boldsymbol{D}}_j\right)^{-1}\nabla_{\tilde{\boldsymbol{H}}_j}\left(\tilde{\boldsymbol{D}}_j^T\tilde{\boldsymbol{D}}_j\right)\left(\tilde{\boldsymbol{D}}_j^T\tilde{\boldsymbol{D}}_j\right)^{-1}\tilde{\boldsymbol{D}}_j^T\\&\quad+\left(\tilde{\boldsymbol{D}}_j^T\tilde{\boldsymbol{D}}_j\right)^{-1}\nabla_{\tilde{\boldsymbol{H}}_j}\tilde{\boldsymbol{D}}_j^T\\&=-\left(\tilde{\boldsymbol{D}}_j^T\tilde{\boldsymbol{D}}_j\right)^{-1}\left(\nabla_{\tilde{\boldsymbol{H}}_j}\tilde{\boldsymbol{D}}_j^T\right)\tilde{\boldsymbol{D}}_j\tilde{\boldsymbol{D}}_j^\dagger-\tilde{\boldsymbol{D}}_j^\dagger\left(\nabla_{\tilde{\boldsymbol{H}}_j}\tilde{\boldsymbol{D}}_j\right)\tilde{\boldsymbol{D}}_j^\dagger\\&\quad+\left(\tilde{\boldsymbol{D}}_j^T\tilde{\boldsymbol{D}}_j\right)^{-1}\nabla_{\tilde{\boldsymbol{H}}_j}\tilde{\boldsymbol{D}}_j^T\\&=\left(\tilde{\boldsymbol{D}}_j^T\tilde{\boldsymbol{D}}_j\right)^{-1}\tilde{\boldsymbol{H}}_j^T\left(\boldsymbol{I}-\tilde{\boldsymbol{D}}_j\tilde{\boldsymbol{D}}_j^\dagger\right)-\tilde{\boldsymbol{D}}_j^\dagger\tilde{\boldsymbol{H}}_j\tilde{\boldsymbol{D}}_j^\dagger.\end{aligned}$$

Define $\tilde{\boldsymbol{y}}_{e,j}=\tilde{\boldsymbol{y}}_j-\tilde{\boldsymbol{D}}_j\boldsymbol{x}_j$. Then,

$$\begin{aligned}\nabla_{\tilde{\boldsymbol{H}}_j}\boldsymbol{x}_j&=\left(\tilde{\boldsymbol{D}}_j^T\tilde{\boldsymbol{D}}_j\right)^{-1}\tilde{\boldsymbol{H}}_j^T\left(\tilde{\boldsymbol{y}}_j-\tilde{\boldsymbol{D}}_j\boldsymbol{x}_j\right)-\tilde{\boldsymbol{D}}_j^\dagger\tilde{\boldsymbol{H}}_j\boldsymbol{x}_j\\&=\left(\tilde{\boldsymbol{D}}_j^T\tilde{\boldsymbol{D}}_j\right)^{-1}\tilde{\boldsymbol{H}}_j^T\tilde{\boldsymbol{y}}_{e,j}-\left(\tilde{\boldsymbol{D}}_j^T\tilde{\boldsymbol{D}}_j\right)^{-1}\tilde{\boldsymbol{D}}_j^T\tilde{\boldsymbol{H}}_j\boldsymbol{x}_j\\&=\left(\tilde{\boldsymbol{D}}_j^T\tilde{\boldsymbol{D}}_j\right)^{-1}\left(\tilde{\boldsymbol{H}}_j^T\tilde{\boldsymbol{y}}_{e,j}-\tilde{\boldsymbol{D}}_j^T\tilde{\boldsymbol{H}}_j\boldsymbol{x}_j\right).\end{aligned}\tag{12}$$

Substitute (12) into (11). One is able to compute $\nabla_{\tilde{\boldsymbol{H}}_j}\nabla f_\mu\left(\tilde{\boldsymbol{D}}_j\right)$, and hence $\nabla_{\boldsymbol{H}_j}\nabla f_\mu\left(\boldsymbol{D}_j\right)$, and $\nabla_{\boldsymbol{H}}\nabla f_\mu$.

C. Line Search Path

The line search mechanism used in this paper is significantly different from the standard one due to the column norm constraint in (2). In a standard line search algorithm, the k^{th} iteration outputs an updated dictionary $\boldsymbol{D}^{(k)}$ via

$$\boldsymbol{D}^{(k)}=\boldsymbol{D}^{(k-1)}+t\cdot\boldsymbol{H},\tag{13}$$

where $t \in \mathbb{R}^+$ is a properly chosen step size and $\boldsymbol{H} \in \mathbb{R}^{m\times d}$ is the line search direction. Common choices for the search direction include $\boldsymbol{H} = -\nabla f_\mu$ for the gradient descent method and $\boldsymbol{H} = -\left(\nabla^2 f_\mu\right)^{-1}\nabla f_\mu$ for the Newton method. However, a direct application of (13) generally results in a dictionary $\boldsymbol{D} \notin \mathcal{D}$.

The line search path in this paper is restricted to the product of Grassmann manifolds. This is because, as has been discussed in Section IV, the objective function f_μ is indeed a function defined on the product of Grassmann manifolds. On the Grassmann manifold $\mathcal{G}_{m,1}$, the geodesic path plays the same role as the straight line in the Euclidean space: given any two distinct points on $\mathcal{G}_{m,1}$, the shortest path that connects these two points is geodesic [62]. Specifically, let $\mathscr{U} \in \mathcal{G}_{m,1}$ be a one-dimensional subspace and $\boldsymbol{u} \in \mathcal{U}_{m,1}$ be the corresponding generator matrix (not unique).[4] 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 [62]

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

Note that $\boldsymbol{u}(t) = -\boldsymbol{u}(t+\pi)$ and hence $\operatorname{span}(\boldsymbol{u}(t)) = \operatorname{span}(\boldsymbol{u}(t+\pi))$. In practice, one can restrict the search path within the interval $t \in [0,\pi)$.

For the dictionary update problem at hand, the line search path is defined as follows. Let $\boldsymbol{H} \in \mathbb{R}^{m\times d}$ be the search direction. ($\boldsymbol{H} = -\nabla f_\mu$ for the gradient descent method and $\boldsymbol{H} = -\left(\nabla^2 f_\mu\right)^{-1}\nabla f_\mu$ for the Newton method.) Let $\boldsymbol{h}_i$ be the i^{th} column of $\boldsymbol{H}$. Define

$$\bar{\boldsymbol{h}}_i = \boldsymbol{h}_i - \boldsymbol{D}_{:,i}\boldsymbol{D}_{:,i}^T\boldsymbol{h}_i,\ \forall i\in\mathcal{I}, \tag{14}$$

so that $\bar{\boldsymbol{h}}_i$ and $\boldsymbol{D}_{:,i}$ are orthogonal. The line search path for dictionary update, say $\boldsymbol{D}(t)$, $t\geq 0$, is given by [62]

$$\begin{cases} \boldsymbol{D}_{:,i}(t) = \boldsymbol{D}_{:,i} \quad \text{if } \left\|\bar{\boldsymbol{h}}_i\right\|_2 = 0, \\ \boldsymbol{D}_{:,i}(t) = \boldsymbol{D}_{:,i}\cos\left(\left\|\bar{\boldsymbol{h}}_i\right\|_2 t\right) + \left(\bar{\boldsymbol{h}}_i/\left\|\bar{\boldsymbol{h}}_i\right\|_2\right)\sin\left(\left\|\bar{\boldsymbol{h}}_i\right\|_2 t\right) \\ \qquad\qquad\qquad \text{if } \left\|\bar{\boldsymbol{h}}_i\right\|_2 \neq 0. \end{cases} \tag{15}$$

VI. Convergence of Primitive SimCO

The focus of this section is on the convergence performance of primitive SimCO when the index set $\mathcal{I}$ contains only one index. The analysis shows deep connections between primitive SimCO and K-SVD. It is clear that the optimization formulations of primitive SimCO and K-SVD are exactly the same when $|\mathcal{I}| = 1$. However, the methods used to solve the optimization problem are quite different: primitive

[4]The generator matrix $\boldsymbol{u}$ is a vector in this case.

SimCO uses standard optimization methods while K-SVD employs SVD. The question is whether these two different approaches will give the same solution eventually. Theorem 1 of this section shows that a gradient descent finds a global optimum with probability one. Hence, when $|\mathcal{I}| = 1$, primitive SimCO and K-SVD are the same in terms of ultimate learning performance. Note that, even though the general case where $|\mathcal{I}| > 1$ is more interesting, it remains open which point SimCO will converge to in this case.

When $|\mathcal{I}| = 1$, the rank-one matrix approximation problem arises in both primitive SimCO and K-SVD. Formally, let $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ be a matrix, where $m \geq 1$ and $n \geq 1$ are arbitrary positive integers. Without loss of generality, assume that $m \leq n$. Suppose that the sorted singular values satisfy $\lambda_1 > \lambda_2 \geq \lambda_3 \geq \cdots \geq \lambda_m$. Define

$$f(\boldsymbol{u}) = \min_{\boldsymbol{w}\in\mathbb{R}^n} \left\|\boldsymbol{A} - \boldsymbol{u}\boldsymbol{w}^T\right\|_F^2, \ \forall \boldsymbol{u} \in \mathcal{U}_{m,1}. \tag{16}$$

The rank-one matrix approximation problem can be written as the following optimization problem

$$\min_{\boldsymbol{u}\in\mathcal{U}_{m,1}} f(\boldsymbol{u}). \tag{17}$$

The performance of gradient descent is analyzed in Theorem 1 for the rank-one matrix approximation problem. To avoid numerical problems that may arise in practical implementations, we consider an ideal gradient descent procedure with infinitesimal step sizes. (Note that true gradient descent requires infinitesimal steps.)

Theorem 1. *Consider a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ and its singular value decomposition. Employ the gradient descent procedure with infinitesimal steps to solve (16). Suppose the starting point, denoted by $\boldsymbol{u}_0$, is randomly generated from the uniform distribution on $\mathcal{U}_{m,1}$. Then the gradient descent procedure finds a global minimizer with probability one.*

The proof is detailed in Appendix A.

Remark 2. The notion of Grassmann manifold is essential in the proof. The reason is that the global minimizer is unique up to the subspace spanned by $\boldsymbol{u}$: if $\boldsymbol{u} \in \mathbb{R}^m$ is a global minimizer, then so is $\boldsymbol{u}'$ for all $\boldsymbol{u}'$ such that $\text{span}(\boldsymbol{u}') = \text{span}(\boldsymbol{u})$.

Remark 3. According to the authors' knowledge, this is the first result showing that a gradient search on Grassmann manifold solves the rank-one matrix approximation problem. In the literature, it has been shown that there are multiple stationary points for the rank-one matrix approximation problem [66, Proposition

4.6.2]. Our results show that a gradient descent method will not converge to any other stationary points than global minimizers. More recently, the rank-one decomposition problem where $\lambda_2 = \lambda_3 = \cdots = \lambda_m = 0$ was studied in [63]. Our proof technique is significantly different as the effects of the eigen-spaces corresponding to $\lambda_2, \cdots, \lambda_m$ need to be considered for the rank-one approximation problem.

VII. Empirical Tests

In this section, we numerically test the proposed primitive and regularized SimCO. In the test of SimCO, all codewords are updated simultaneously, i.e., $\mathcal{I} = [d]$. In Section VII-A, we show that MOD[5], K-SVD, and primitive SimCO may result in an ill-conditioned dictionary while regularization can mitigate this problem. Learning performance of synthetic and real data is presented in Sections VII-B and VII-C respectively. A running time comparison of different algorithms is conducted in Section VII-D.

It is worth noting that Algorithm 2 (gradient descent) is used for the analysis of singular points in Section VII-A because of the reasons explained in Footnote 1. Algorithm 3 (LSNCG) is employed for synthetic and real data tests in Sections VII-B and VII-C due to its fast convergence rate. Both regularized "K-SVD" and regularized MOD are based on second order optimization methods to ensure a fair comparison.

A. Ill-conditioned Dictionaries

In this subsection, we handpick a particular example to show that MOD, K-SVD and 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 $\boldsymbol{D}_{\text{true}} \in \mathbb{R}^{16\times 32}$, $\boldsymbol{X}_{\text{true}} \in \mathbb{R}^{32\times 78}$, and each column of $\boldsymbol{X}$ contains exactly 4 nonzero components. We assume that the sparse coding stage is perfect, i.e., the true sparsity pattern Ω_{true} is available. We start with a particular choice of the initial dictionary $\boldsymbol{D}_0 \in \mathcal{D}$. The regularization constant μ is set to 0 and 0.01 for primitive and regularized SimCOs respectively. Define the *condition number of a dictionary* $\boldsymbol{D}$ as

$$\kappa(\boldsymbol{D}) = \max_{1\le j\le n} \lambda_{\max}(\boldsymbol{D}_j)/\lambda_{\min}(\boldsymbol{D}_j),$$

where $\boldsymbol{D}_j = \boldsymbol{D}_{:,\Omega(:,j)}$. The numerical results are presented in Figure 1, where $f_{\mu=0}$, $\|\nabla f_{\mu=0}\|_F / \|\boldsymbol{Y}\|_F$, and $\kappa(\boldsymbol{D})$ are compared from the left to the right. Note that in this example, $\kappa(\boldsymbol{D}_{\text{true}}) = 3.39$.

The results in Figure 1 show that

[5]In the tested MOD, the columns in $\boldsymbol{D}$ are normalized after each dictionary update. This extra step is performed because many sparse coding algorithms require a normalized dictionary. Furthermore, our preliminary simulations (not shown in this paper) show that the performance of dictionary update could seriously deteriorate if the columns are not normalized.

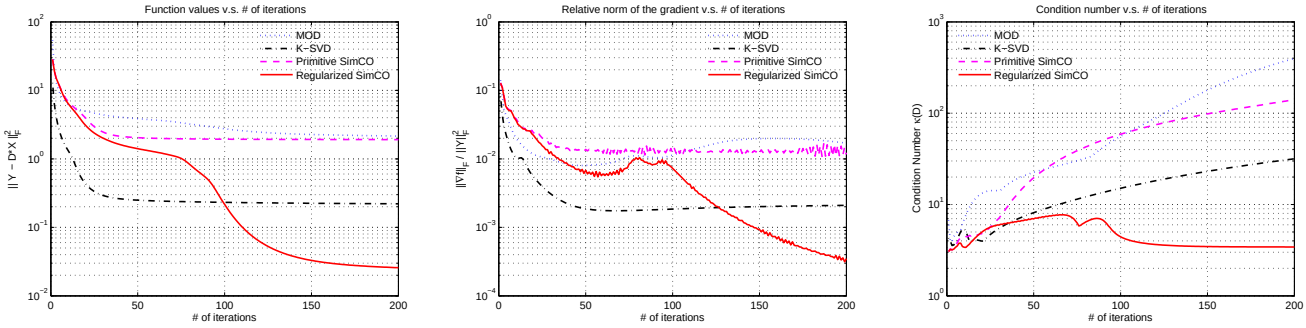
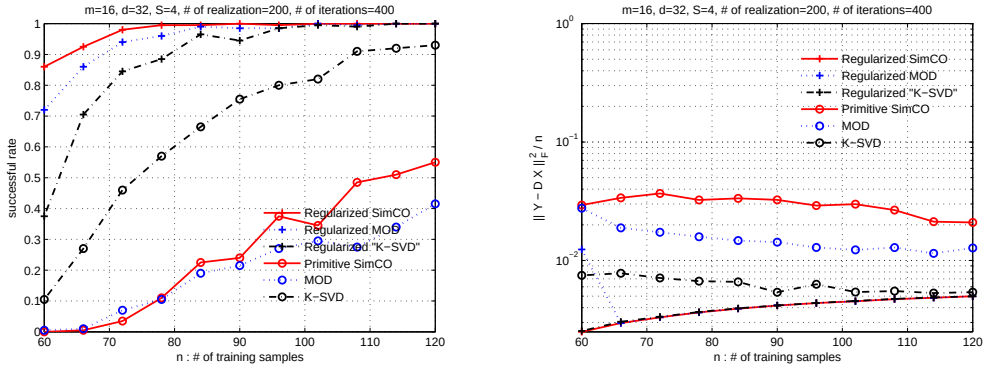


Figure 1: Starting with the same point, the convergence behaviors of MOD, K-SVD, primitive SimCO and regularized SimCO are different. In this particular example, only regularized SimCO avoids converging to a singular point.

1) When the number of iterations exceeds 50, MOD, K-SVD and primitive SimCO stop improving the training performance. Surprisingly, the gradient $\nabla f_{\mu=0}$ in these methods does not converge to zero. This implies that these methods *do not converge to a local minimizer*. A more careful study reveals that these algorithms converge to singular points where $\kappa\left(\boldsymbol{D}\right)$ becomes large ($\kappa\left(\boldsymbol{D}\right) > 10$ for MOD, K-SVD, and primitive SimCO).
2) By adding a regularization term and choosing the regularization constant properly, regularized SimCO avoids the convergence to an ill-conditioned dictionary, hence improves the performance.

It is worth noting that the SimCO formulation is crucial for distinguishing between singular points and local minimizers. In the SimCO formulation, the objective function only involves the dictionary, and the gradient of the objective function can be easily computed via (8) and (10). If the search process converges to a local minimizer, the gradient should converge to zero. When the gradient does not vanish and changes rapidly in a neighborhood, the convergence point must be a singular point.

We also observe that the singular points, rather than the local minima, are the bottleneck. Towards this end, we randomly pick converged dictionaries in MOD, K-SVD, and primitive SimCO (from the case where there is no noise and Ω_{true} is priorly known). Surprisingly, we found that these algorithms either converge to a global minimizer or a singular point. Among the randomly picked converged dictionaries, no local minimizer has been found yet. Furthermore, as we will show in the next subsection, by adding the regularization term and forcing the search path away from singular points, substantial performance improvement can be achieved. All these suggest that singular points tend to be the major obstacle preventing these algorithms from converging to a global minimizer.



(a) Noiseless case.

(b) Noisy case: SNR of training samples is 20 dB. Note that there always exists a floor in the reconstruction error which is proportional to noise.

Figure 2: Performance comparison of dictionary update (no sparse coding step).

B. Experiments on Synthetic Data

The setting for synthetic data tests is summarized as follows. The training samples are generated via $\boldsymbol{Y} = \boldsymbol{D}_{\text{true}}\boldsymbol{X}_{\text{true}}$. Here, the columns of $\boldsymbol{D}_{\text{true}}$ are randomly generated from the uniform distribution on 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 the tests, we fix $m = 16$, $d = 32$, and $S = 4$, and change n, i.e., the number of training samples. Generally speaking, the fewer training samples there are, the more challenging the dictionary update is. In our experiments, we intentionally choose the challenging case with small n.

We first focus on the performance of dictionary update by assuming that the true sparsity pattern Ω_{true} is available. In regularized methods, the regularization constant μ is sequentially reduced to zero: the total number of iterations is set to 400; we change μ from $1e-1$ to $1e-2$, $1e-3$, and $1e-4$, for every 100 iterations. Experiments for both noiseless and noisy cases are performed. Note that in the noiseless case, the sparse representation distortion $\|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2/n$ can approach zero. It is more indicative to use success rate rather than distortion: a success is claimed when $\|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2/n \leq 10^{-7}$ and a failure is claimed otherwise. For the noisy case, there always exists a floor in the representation distortion that is proportional to noise. The normalized distortion $\|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2/n$ serves as a good performance measure. The simulation results are presented in Figure 2. It is evident that regularization significantly improves the performance and that among all the regularized methods, regularized SimCO is consistently better than others.

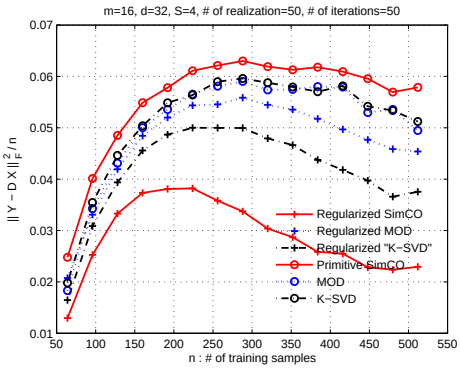


Figure 3: Performance comparison of dictionary learning using OMP for sparse coding.

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 used for testing the K-SVD method in [10], [67]. The overall dictionary learning procedure is given in 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 tests, the number of outer-iterations is set to 50, and the number of inner-iterations of is set to 1. Furthermore, in 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 normalized learning performance $\|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2/n$ is depicted in Figure 3. Again, the average performance of regularized SimCO is consistently better than that of other methods.

C. Numerical Results for Image Denoising

As we mentioned in the introduction, dictionary learning methods have many applications. In this subsection, we look at one particular application, i.e., image denoising. Here, an image corrupted by noise was used to train the dictionary: we take 1,000 (significantly less than 65,000 used in [67]) blocks (of size 8×8) of the corrupted image as training samples. The number of codewords in the training dictionary is $d = 256$. For dictionary learning, we iterate the sparse coding and dictionary update stages 10 times. The sparse coding stage is based on the OMP algorithm implemented in [67]. In the dictionary update stage, different algorithms are tested and the number of iterations in dictionary update is set to 50. The regularization constant is set to $\mu = 0.05$. After the whole process of dictionary learning, we use the learned dictionary to reconstruct the image. The reconstruction results are presented in Figure 4. While all dictionary learning methods significantly improves the image SNRs, the largest gain was obtained from regularized SimCO.

Figure 4: Example of the image denoising using dictionary learning. PSNR values in dB are given in sub-figure titles.

D. Comments on the Running Time

This subsection compares the computational complexity of MOD, K-SVD, and SimCO. As detailed in Sections III and V, MOD uses an approximation to the Hessian while Algorithm 3 is based on the exact Hessian (without explicitly computed). As a result, the complexity of MOD and SimCO is on the same level: the computational cost for each MOD iteration is less than that for each SimCO iteration, but the number of iterations required for convergence in MOD is larger than that in SimCO. As opposed to MOD and SimCO where all codewords are simultaneously updated, K-SVD updates codewords individually. Despite the fact that a closed form solution can be obtained for each update via SVD, the speed of K-SVD is often slower than MOD and SimCO because of the individual update. The actual running time in practice is compared for different algorithms in Table I. The numerical comparison is consistent with the qualitative analysis above.

VIII. Conclusions

We have presented a new framework for dictionary update. It allows not only a simultaneous update of all codewords and the corresponding coefficients but also the observation that singular points rather than local minima are the bottleneck for the dictionary update. To mitigate the effects of singularity, regularized

Table I: Comparison of running time (in seconds) for dictionary learning. Note that sparse coding step was included in producing Figures 3 and 4.

	MOD	K-SVD	Prim. SimCO	Reg. MOD	Reg. "K-SVD"	Reg. SimCO
Fig. 2(a)	2.4×10^4	2.0×10^5	5.1×10^4	5.9×10^3	2.0×10^5	2.3×10^4
Fig. 2(b)	2.3×10^4	1.9×10^5	5.0×10^4	6.3×10^3	8.6×10^4	2.1×10^4
Fig. 3	1.5×10^4	3.7×10^4	3.1×10^4	4.2×10^4	9.7×10^4	8.7×10^4
Fig. 4	-	-	-	7.50	18.13	14.68

SimCO has been proposed. First and second order optimization procedures have been implemented. Numerical experiments verify that regularization substantially improves the performance.

APPENDIX

A. *Proof of Theorem 1*

The following notations are repeatedly used in the proofs. Consider the singular value decomposition $\boldsymbol{A} = \sum_{i=1}^{m} \lambda_i \boldsymbol{u}_{\boldsymbol{A},i} \boldsymbol{v}_{\boldsymbol{A},i}^T$, where $\lambda_1 > \lambda_2 \geq \cdots \geq \lambda_m \geq 0$ are the singular values, and $\boldsymbol{u}_{\boldsymbol{A},i}$ and $\boldsymbol{v}_{\boldsymbol{A},i}$ are the left and right singular vectors corresponding to λ_i respectively. It is clear that the objective function $f(\boldsymbol{u}) = \inf_{\boldsymbol{w}\in\mathbb{R}^n} \left\|\boldsymbol{A} - \boldsymbol{u}\boldsymbol{w}^T\right\|_F^2$ has two global minimizers $\pm\boldsymbol{u}_{\boldsymbol{A},1}$. For a given $\boldsymbol{u} \in \mathcal{U}_{m,1}$, the angle between $\boldsymbol{u}$ and the closest global minimizer is defined as

$$\theta = \cos^{-1} \left|\langle \boldsymbol{u}, \boldsymbol{u}_{\boldsymbol{A},1} \rangle\right|.$$

The crux of the proof is that along the gradient descent path, the angle θ is monotonically decreasing. Suppose that the starting angle is less than $\pi/2$. Then the only stationary points are when the angle θ is zero. Hence, the gradient descent search converges to a global minimizer. The probability one part comes from that the starting angle equals to $\pi/2$ with probability zero.

To formalize the idea, it is assumed that the starting point $\boldsymbol{u}_0 \in \mathcal{U}_{m,1}$ is randomly generated from the uniform distribution on the Stiefel manifold. Define a set $\mathcal{B} \subset \mathcal{U}_{m,1}$ to describe the set of "bad" starting points. It is defined by

$$\mathcal{B} = \left\{\boldsymbol{u} \in \mathcal{U}_{m,1} : \ \boldsymbol{u}^T \boldsymbol{u}_{\boldsymbol{A},1} = 0\right\},$$

which contains all unit vectors that are orthogonal to $\boldsymbol{u}_{\boldsymbol{A},1}$. According to [68], under the uniform measure on $\mathcal{U}_{m,1}$, the measure of the set $\mathcal{B}$ is zero. As a result, the starting point $\boldsymbol{u}_0 \notin \mathcal{B}$ with probability one. The reason that we refer to $\mathcal{B}$ as the set of "bad" starting points is explained by the following lemma.

Lemma 4. *Starting from any $\boldsymbol{u}_0 \in \mathcal{B}$, a gradient descent path stays in the set $\mathcal{B}$.*

Proof: This lemma can be proved by computing the gradient of f at a $\boldsymbol{u} \in \mathcal{B}$. Let $\boldsymbol{w}_{\boldsymbol{u}} \in \mathbb{R}^n$ be the optimal solution of the least squares problem in $f(\boldsymbol{u}) = \inf_{\boldsymbol{w}\in\mathbb{R}^n} \left\|\boldsymbol{A} - \boldsymbol{u}\boldsymbol{w}^T\right\|_F^2$. It can be verified that $\boldsymbol{w}_{\boldsymbol{u}} = \boldsymbol{A}^T\boldsymbol{u}$ and $\nabla f = -2\left(\boldsymbol{A} - \boldsymbol{u}\boldsymbol{w}_{\boldsymbol{u}}^T\right)\boldsymbol{w}_{\boldsymbol{u}}$. It is clear that

$$\begin{aligned}\nabla f &= -2\left(\boldsymbol{A} - \boldsymbol{u}\boldsymbol{w}_{\boldsymbol{u}}^T\right)\boldsymbol{w}_{\boldsymbol{u}} = -2\left(\boldsymbol{A} - \boldsymbol{u}\boldsymbol{u}^T\boldsymbol{A}\right)\boldsymbol{A}^T\boldsymbol{u} \\ &= -2\sum_i \lambda_i^2 \boldsymbol{u}_{\boldsymbol{A},i}\boldsymbol{u}_{\boldsymbol{A},i}^T\boldsymbol{u} + 2\boldsymbol{u}\left(\boldsymbol{u}^T\boldsymbol{A}\boldsymbol{A}^T\boldsymbol{u}\right)\end{aligned}$$

When $\boldsymbol{u}_0 \in \mathcal{B}$, it holds that $\langle \boldsymbol{u}_0, \boldsymbol{u}_{\boldsymbol{A},1}\rangle = 0$ and $\langle \nabla f(\boldsymbol{u}_0), \boldsymbol{u}_{\boldsymbol{A},1}\rangle = 0$. Since both $\boldsymbol{u}_0$ and the gradient descent direction are orthogonal to $\boldsymbol{u}_{\boldsymbol{A},1}$, the gradient descent path starting from $\boldsymbol{u}_0 \in \mathcal{B}$ stays in $\mathcal{B}$. ■

Now consider a starting points $\boldsymbol{u}_0 \notin \mathcal{B}$. We shall show that the angle θ is monotonically decreasing along the gradient descent path. Towards this end, the notions of directional derivative play an important role. View θ as a function of $\boldsymbol{u} \in \mathcal{U}_{m,1}$. The directional derivative of θ at $\boldsymbol{u} \in \mathcal{U}_{m,1}$ along a direction vector $\boldsymbol{h} \in \mathbb{R}^m$, denoted by $\nabla_{\boldsymbol{h}}\theta \in \mathbb{R}$, is defined as

$$\nabla_{\boldsymbol{h}}\theta = \lim_{\epsilon\to 0}\frac{\theta(\boldsymbol{u}+\epsilon\boldsymbol{h}) - \theta(\boldsymbol{u})}{\epsilon}.$$

Note the relationship between the directional derivative and the gradient given by $\nabla_{\boldsymbol{h}}\theta = \langle \nabla\theta, \boldsymbol{h}\rangle$. With this definition, the following lemma plays the central role in establishing Theorem 1.

Lemma 5. *Consider a $\boldsymbol{u} \in \mathcal{U}_{m,1}$ such that $\theta(\boldsymbol{u}) := \cos^{-1}(|\langle \boldsymbol{u}, \boldsymbol{u}_{\boldsymbol{A},1}\rangle|) \in (0, \pi/2)$. Let $\boldsymbol{h}_f = -\nabla f(\boldsymbol{u})$ be the gradient of the objective function f at $\boldsymbol{u}$. Then it holds $\nabla_{\boldsymbol{h}_f}\theta < 0$.*

The proof of this lemma is detailed in Appendix B.

The implications of this lemma are twofold. First, it implies that $\boldsymbol{h}_f = -\nabla f \neq \boldsymbol{0}$ for all $\boldsymbol{u}$ such that $\theta(\boldsymbol{u}) \in (0, \pi/2)$. Hence, the only possible stationary points in $\mathcal{U}_{m,1}\backslash\mathcal{B}$ are $\boldsymbol{u}_{\boldsymbol{A},1}$ and $-\boldsymbol{u}_{\boldsymbol{A},1}$. Second, starting from $\boldsymbol{u}_0 \in \mathcal{B}$, the angle θ decreases along the gradient descent path. As a result, a gradient descent path will not enter $\mathcal{B}$. It will converge to $\boldsymbol{u}_{\boldsymbol{A},1}$ or $-\boldsymbol{u}_{\boldsymbol{A},1}$. Theorem 1 is therefore proved.

B. Proof of Lemma 5

This appendix is devoted to prove Lemma 5, i.e., $\nabla_{\boldsymbol{h}_f}\theta < 0$. Note that $\nabla_{\boldsymbol{h}_f}\theta = \langle \boldsymbol{h}_f, \nabla\theta\rangle = \langle -\nabla f, \nabla\theta\rangle = \nabla_{-\nabla\theta} f$. It suffices to show that $\nabla_{-\nabla\theta} f < 0$.

Towards this end, the following definitions are useful. Define $s = \text{sign}\left(\boldsymbol{u}^T\boldsymbol{u}_{\boldsymbol{A},1}\right)$. Then the vector $s\boldsymbol{u}_{\boldsymbol{A},1}$

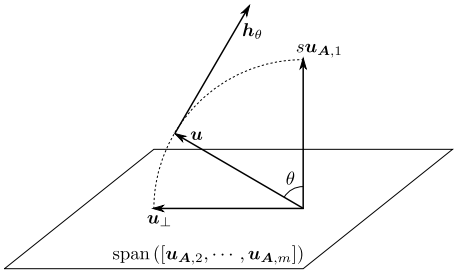


Figure 5: Illustration of $\boldsymbol{u}$, $\boldsymbol{u}_{\boldsymbol{A},1}$, $\boldsymbol{h}_\theta$ and $\boldsymbol{u}_\perp$.

is one of the two global minimizers that is the closest to $\boldsymbol{u}$. It can be also verified that $\theta = \cos^{-1}\langle \boldsymbol{u}, s\boldsymbol{u}_{\boldsymbol{A},1}\rangle$. Furthermore, suppose that $\theta \in (0, \pi/2)$. Define

$$\boldsymbol{h}_\theta = \frac{s\boldsymbol{u}_{\boldsymbol{A},1} - \boldsymbol{u}\cos\theta}{\sin\theta}, \text{ and } \boldsymbol{u}_\perp = \frac{\boldsymbol{u} - s\boldsymbol{u}_{\boldsymbol{A},1}\cos\theta}{\sin\theta}.$$

Clearly, vectors $\boldsymbol{h}_\theta$ and $\boldsymbol{u}_\perp$ are well-defined when $\theta \in (0, \pi/2)$. The relationship among $\boldsymbol{u}$, $\boldsymbol{u}_{\boldsymbol{A},1}$, $\boldsymbol{h}_\theta$ and $\boldsymbol{u}_\perp$ is illustrated in Figure 5. Intuitively, the vector $\boldsymbol{h}_\theta$ is the tangent vector that pushes $\boldsymbol{u}$ towards the global minimizer $s\boldsymbol{u}_{\boldsymbol{A},1}$.

In the following, we show that $\nabla_{-\nabla\theta} f = \nabla_{\boldsymbol{h}_\theta} f$ if we restrict $\boldsymbol{u} \in \mathcal{U}_{m,1}$. By the definition of the directional derivative, one has[6]

$$\nabla_{-\nabla\theta}\boldsymbol{u} = \lim_{\epsilon\to 0}\frac{\boldsymbol{u} - \epsilon\nabla\theta}{\|\boldsymbol{u} - \epsilon\nabla\theta\|}.$$

Note that

$$\begin{aligned}\nabla\theta &= \nabla\left(\cos^{-1}(\cos\theta)\right)\\ &= -\frac{1}{\sqrt{1-\cos^2\theta}}\nabla\langle \boldsymbol{u}, s\boldsymbol{u}_{\boldsymbol{A},1}\rangle = -\frac{1}{\sin\theta}\left(s\boldsymbol{u}_{\boldsymbol{A},1}\right).\end{aligned}$$

Since $s\boldsymbol{u}_{\boldsymbol{A},1} = \boldsymbol{u}\cos\theta + \boldsymbol{h}_\theta\sin\theta$, one has

$$\begin{aligned}\boldsymbol{u} - \epsilon\nabla\theta &= \boldsymbol{u} + \frac{\epsilon}{\sin\theta}\left(s\boldsymbol{u}_{\boldsymbol{A},1}\right)\\ &= \boldsymbol{u}\left(1 + \epsilon\cos\theta/\sin\theta\right) + \epsilon\boldsymbol{h}_\theta.\end{aligned}$$

Substitute it back to $\nabla_{-\nabla\theta}\boldsymbol{u}$. One has $\nabla_{-\nabla\theta}\boldsymbol{u} = \boldsymbol{h}_\theta$. In other words, if $\boldsymbol{u} \in \mathcal{U}_{m,1}$, then $\nabla_{-\nabla\theta} f = \nabla_{\boldsymbol{h}_\theta} f$.

[6]The denominator comes from the restriction that $\boldsymbol{u} \in \mathcal{U}_{m,1}$.

To compute $\nabla_{\boldsymbol{h}_\theta} f$, note that $f(\boldsymbol{u}) = \left\|\boldsymbol{A} - \boldsymbol{u}\boldsymbol{w}_{\boldsymbol{u}}^T\right\|_F^2 = \|\boldsymbol{A}\|_F^2 - \left\|\boldsymbol{u}^T\boldsymbol{A}\right\|_2^2$. Now define

$$g(\boldsymbol{u}) = \left\|\boldsymbol{u}^T\boldsymbol{A}\right\|_2^2.$$

Then clearly $\nabla_{\boldsymbol{h}_\theta} f = -\nabla_{\boldsymbol{h}_\theta} g$. To proceed, we also decompose $\boldsymbol{A}$ as follows. Recall the SVD of $\boldsymbol{A}$ given by $\boldsymbol{A} = \sum_{i=1}^m \lambda_i \boldsymbol{u}_{\boldsymbol{A},i}\boldsymbol{v}_{\boldsymbol{A},i}^T$. Let $\boldsymbol{U}_{\boldsymbol{A},\perp} \in \mathcal{U}_{m,m-1}$ contain the left singular vectors corresponding to $\lambda_2, \cdots, \lambda_m$, i.e., $\boldsymbol{U}_{\boldsymbol{A},\perp} = [\boldsymbol{u}_{\boldsymbol{A},2}, \cdots, \boldsymbol{u}_{\boldsymbol{A},m}]$. Similarly define $\boldsymbol{V}_{\boldsymbol{A},\perp}$. Then,

$$\begin{aligned}\boldsymbol{A} &= [\boldsymbol{u}_{\boldsymbol{A},1}, \boldsymbol{U}_{\boldsymbol{A},\perp}]\,\mathrm{diag}\left([\lambda_1, \cdots, \lambda_m]\right)\begin{bmatrix}\boldsymbol{v}_{\boldsymbol{A},1}^T \\ \boldsymbol{V}_{\boldsymbol{A},\perp}^T\end{bmatrix} \\ &= [\boldsymbol{u}_{\boldsymbol{A},1}, \boldsymbol{U}_{\boldsymbol{A},\perp}]\left[\boldsymbol{w}_{\boldsymbol{A},1}, \boldsymbol{W}_{\boldsymbol{A},\perp}\right]^T,\end{aligned}$$

where $\boldsymbol{w}_{\boldsymbol{A},i} = \lambda_i \boldsymbol{v}_{\boldsymbol{A},i}$ for $i = 1, \cdots, m$, and $\boldsymbol{W}_{\boldsymbol{A},\perp} = [\boldsymbol{w}_{\boldsymbol{A},2}, \cdots, \boldsymbol{w}_{\boldsymbol{A},m}]$. It is straightforward to verify that $\boldsymbol{w}_{\boldsymbol{A}}^T\boldsymbol{W}_{\boldsymbol{A},\perp} = \boldsymbol{0}$.

The function $g(\boldsymbol{u})$ can be decomposed into two parts. Note that

$$\begin{aligned}g(\boldsymbol{u}) &= \left\|\boldsymbol{u}^T\left[\boldsymbol{u}_{\boldsymbol{A},1}, \boldsymbol{U}_{\boldsymbol{A},\perp}\right]\left[\boldsymbol{w}_{\boldsymbol{A},1}, \boldsymbol{W}_{\boldsymbol{A},\perp}\right]^T\right\|_2^2 \\ &= \left\|\boldsymbol{u}^T\boldsymbol{u}_{\boldsymbol{A},1}\boldsymbol{w}_{\boldsymbol{A},1}^T\right\|_2^2 + \left\|\boldsymbol{u}^T\boldsymbol{U}_{\boldsymbol{A},\perp}\boldsymbol{W}_{\boldsymbol{A},\perp}^T\right\|_2^2 \\ &\quad + 2\left\langle\boldsymbol{u}^T\boldsymbol{u}_{\boldsymbol{A},1}\boldsymbol{w}_{\boldsymbol{A},1}^T, \boldsymbol{u}^T\boldsymbol{U}_{\boldsymbol{A},\perp}\boldsymbol{W}_{\boldsymbol{A},\perp}^T\right\rangle \\ &= \left\|\boldsymbol{u}^T\boldsymbol{u}_{\boldsymbol{A},1}\boldsymbol{w}_{\boldsymbol{A},1}^T\right\|_2^2 + \left\|\boldsymbol{u}^T\boldsymbol{U}_{\boldsymbol{A},\perp}\boldsymbol{W}_{\boldsymbol{A},\perp}^T\right\|_2^2,\end{aligned}$$

where the last equality follows from that $\boldsymbol{W}_{\boldsymbol{A},\perp}^T\boldsymbol{w}_{\boldsymbol{A}} = \boldsymbol{0}$ and hence

$$\left\langle\boldsymbol{u}^T\boldsymbol{u}_{\boldsymbol{A},1}\boldsymbol{w}_{\boldsymbol{A},1}^T, \boldsymbol{u}^T\boldsymbol{U}_{\boldsymbol{A},\perp}\boldsymbol{W}_{\boldsymbol{A},\perp}^T\right\rangle = 0.$$

To further simplify $g(\boldsymbol{u})$, note that $\cos\theta = \left|\boldsymbol{u}^T\boldsymbol{u}_{\boldsymbol{A}}\right|$. Furthermore, it is straightforward to verify that the projection of $\boldsymbol{u}$ on $\mathrm{span}\,(\boldsymbol{U}_{\boldsymbol{A},\perp})$ is given by $\boldsymbol{U}_{\boldsymbol{A},\perp}\boldsymbol{U}_{\boldsymbol{A},\perp}^T\boldsymbol{u} = \boldsymbol{u}_\perp \sin\theta$. Define $\boldsymbol{u}_R = \boldsymbol{U}_{\boldsymbol{A},\perp}^T\boldsymbol{u}_\perp \in \mathbb{R}^{m-1}$. Then, $\|\boldsymbol{u}_R\| = 1$ and

$$\begin{aligned}&\left\|\boldsymbol{u}^T\boldsymbol{U}_{\boldsymbol{A},\perp}\boldsymbol{W}_{\boldsymbol{A},\perp}^T\right\|_2^2 \\ &= \sin^2\theta\left\|\boldsymbol{u}_\perp^T\boldsymbol{U}_{\boldsymbol{A},\perp}\boldsymbol{W}_{\boldsymbol{A},\perp}^T\right\|_2^2 \\ &= \sin^2\theta\boldsymbol{u}_R^T\mathrm{diag}\left(\left[\lambda_2^2, \cdots, \lambda_m^2\right]\right)\boldsymbol{u}_R^T.\end{aligned}$$

Hence,

$$g(\boldsymbol{u}) = \cos^2\theta \cdot \lambda_1 + \sin^2\theta \boldsymbol{u}_R^T \text{diag}\left(\left[\lambda_2^2, \cdots, \lambda_m^2\right]\right) \boldsymbol{u}_R.$$

We are now ready to decide the sign of $\nabla_{\boldsymbol{h}_\theta} g$. It is straightforward to verify that

$$\nabla_{\boldsymbol{h}_\theta} \cos\theta = \lim_{\epsilon \to 0} \left\langle \frac{\boldsymbol{u} + \epsilon \boldsymbol{h}_\theta}{\sqrt{1+\epsilon^2}}, s\boldsymbol{u}_{\boldsymbol{A},1} \right\rangle = \sin\theta,$$

and similarly $\nabla_{\boldsymbol{h}_\theta} \sin\theta = -\cos\theta$. Therefore,

$$\begin{aligned} \nabla_{\boldsymbol{h}_\theta} \boldsymbol{u}_\perp &= \nabla_{\boldsymbol{h}_\theta} \left(\frac{\boldsymbol{u} - -\cos\theta s\boldsymbol{u}_{\boldsymbol{A},1}}{\sin\theta} \right) \\ &= \frac{\boldsymbol{h}_\theta \sin\theta + \boldsymbol{u}\cos\theta - s\boldsymbol{u}_{\boldsymbol{A},1}}{\sin^2\theta} \\ &= \frac{s\boldsymbol{u}_{\boldsymbol{A},1} - s\boldsymbol{u}_{\boldsymbol{A},1}}{\sin^2\theta} = \boldsymbol{0}, \end{aligned}$$

and $\nabla_{\boldsymbol{h}_\theta} \boldsymbol{u}_R = \nabla_{\boldsymbol{h}_\theta} \left(\boldsymbol{U}_{\boldsymbol{A},\perp}^T \boldsymbol{u}_\perp \right) = \boldsymbol{0}$. Hence, one has

$$\nabla_{\boldsymbol{h}_\theta} g = \sin 2\theta \left(\lambda_1 - \boldsymbol{u}_R^T \text{diag}\left(\left[\lambda_2^2, \cdots, \lambda_m^2\right]\right) \boldsymbol{u}_R \right).$$

Note that

$$\begin{aligned} &\boldsymbol{u}_R^T \text{diag}\left(\left[\lambda_2^2, \cdots, \lambda_m^2\right]\right) \boldsymbol{u}_R \\ &\leq \boldsymbol{u}_R^T \text{diag}\left(\left[\lambda_2^2, \cdots, \lambda_2^2\right]\right) \boldsymbol{u}_R = \lambda_2 < \lambda_1. \end{aligned}$$

It can be concluded that when $\theta \in (0, \pi/2)$, $\nabla_{\boldsymbol{h}_\theta} g > 0$ and $\nabla_{\boldsymbol{h}_\theta} f = -\nabla_{\boldsymbol{h}_\theta} g < 0$. Lemma 5 is therefore proved.

ACKNOWLEDGMENTS

We thank Mr. Xiaochen Zhao for his help in Matlab implementation of Algorithm 3, Miss Qingju Liu for running parts of the numerical tests, Dr Mark Barnard and Dr Fei Yan for proofreading the manuscript. We are also very grateful for the reviewers' comments which have helped us considerably improving this paper.

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 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] R. Rubinstein, A. M. Bruckstein, and M. Elad, "Dictionaries for sparse representation modelling," *Proceedings of IEEE*, vol. 98, no. 6, pp. 1045–1057, 2010.

[44] 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.

[45] K. Skretting and K. Engan, "Recursive least squares dictionary learning algorithm," *IEEE Trans. Signal Process.*, vol. 58, no. 4, pp. 2121–2130, 2010.

[46] 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.

[47] M. Yaghoobi, L. Daudet, and M. E. Davies, "Parametric dictionary design for sparse coding," *IEEE Trans. Signal Process.*, vol. 57, no. 12, pp. 4800–4810, 2009.

[48] P. Sallee and B. A. Olshausen, "Learning sparse multiscale image representations," in *Conf. Neural Information Processing Systems*, 2003.

[49] C. Cadieu and B. A. Olshausen, "Learning transformational invariants from time-varying natural images," in *Proc. Conf. Neural Information Processing Systems*, 2088.

[50] T. Blumensath and M. Davies, "Sparse and shift-invariant representations of music," *IEEE Trans. Speech Audio Process.*, vol. 14, no. 1, pp. 50–57, 2006.

[51] B. Mailhé, S. Lesage, R. Gribonval, F. Bimbot, and P. Vandergheynst, "Shift invariant dictionary learning for sparse representations: Extending K-SVD," in *European Signal Processing Conf.*, vol. 4, 2008.

[52] M. Aharon and M. Elad, "Sparse and redundant modeling of image content using an image-signature-dictionary," *SIAM J. Imaging Sci.*, vol. 1, no. 3, pp. 228–247, 2008.

[53] P. Smaragdis, B. Raj, and M. Shashanka, "Sparse and shift-invariant feature extraction from non-negative data," in *IEEE Int. Conf. on Audio and Speech Signal Processing*, 2008, pp. 2069–2072.

[54] R. Gribonval and M. Nielsen, "Sparse representations in unions of bases," *IEEE Trans. Inf. Theory*, vol. 49, no. 12, pp. 3320–3325, 2003.

[55] P. Jost, P. Vandergheynst, S. Lesage, and R. Gribonval, "BMoTIF: An efficient algorithm for learning translation invariant dictionaries," in *IEEE Int. Conf. Acoust., Speech, Signal Process.*, vol. 5, 2006, pp. 857–860.

[56] K. Labusch, E. Barth, and T. Martinetz, "Robust and fast learning of sparse codes with stochastic gradient descent," *IEEE J. Selected Topics in Signal Process.*, vol. 5, no. 5, pp. 1048–1060, 2011.

[57] J. Mairal, F. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *J. Mach. Learn. Res.*, vol. 11, pp. 16–60, 2010.

[58] G. Monaci, P. Vandergheynst, and F. T. Sommer, "Learning bimodal structure in audio-visual data," *IEEE Trans. Neural Netw.*, vol. 20, no. 12, pp. 1898–1910, 2009.

[59] A. L. Casanovas, G. Monaci, P. Vandergheynst, and R. Gribonval, "Blind audiovisual source separation based on sparse redundant representations," *IEEE Trans. Multimedia*, vol. 12, no. 5, pp. 358–371, 2010.

[60] I. Tošić and P. Frossard, "Dictionary learning: what is the right representation for my signal," *IEEE Signal Process. Mag.*, vol. 28, no. 2, pp. 27–38, March 2011.

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

[62] 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.

[63] W. Dai, E. Kerman, and O. Milenkovic, "A geometric approach to low-rank matrix completion," *IEEE Trans. Inform. Theory*, vol. 58, no. 1, pp. 237–247, Jan. 2012.

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

[65] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes: The Art of Scientific Computing*, 3rd ed. Cambridge University Press, 2007, ch. 10.2. Golden Section Search in One Dimension.

[66] P.-A. Absil, R. Mahony, and R. Sepulchre, *Optimization Algorithms on Matrix Manifolds*. Princeton University Press, 2008.

[67] 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.

[68] W. Dai, Y. Liu, and B. Rider, "Quantization bounds on Grassmann manifolds and applications to MIMO systems," *IEEE Trans. Inform. Theory*, vol. 54, no. 3, pp. 1108–1123, March 2008.

Wei Dai (S'01, M'07) received his B.E. degree in electronic engineering from Tsinghua University in 1999, and his Ph.D. degree in electrical and computer engineering from the University of Colorado at Boulder in 2007. From 2007 to 2010, he was a Postdoctoral Research Associate at the University of Illinois at Urbana-Champaign. Since 2011, he has been a Lecturer (an Assistant Professor) in the Department of Electrical and Electronic Engineering, Imperial College London. His interdisciplinary research interests include sparse signal processing, wireless communications, and applications of information theory and signal processing to biology.

Tao Xu (S'10) was born in Beijing, China, in 1983. He received the B.E. degree in 2005 from Xidian University and the M.E. degree in 2008 from Beijing University of Posts and Telecommunications, both in software engineering. He is currently studying the Ph.D. degree in electronic engineering at University of Surrey, funded by the Centre for Vision Speech and Signal Processing and the China Scholarship Council. His current research interests are in the areas of machine learning and signal processing.

Wenwu Wang (M'03, SM'11) was born in Anhui, China, in 1974. He received the B.Sc. degree in automatic control in 1997, the M.E. degree in control science and control engineering in 2000, and the Ph.D. degree in navigation guidance and control in 2002, all from Harbin Engineering University, China.

He joined the Department of Electronic Engineering, King's College London, U.K., as a Postdoctoral Research Associate in May 2002, and then transferred to the Cardiff School of Engineering, Cardiff University, U.K., in January 2004. In May 2005, he joined the Tao Group Ltd. (now Antix Labs Ltd.), U.K., as a DSP engineer. In September 2006, he moved to the Creative Technology Ltd. working at the Sensaura Division, U.K., as a Software R&D Engineer. Since May 2007, he has been with the Centre for Vision Speech and Signal Processing, University of Surrey, U.K., where he is currently a Lecturer. He is also a member of both the Ministry of Defence (MoD) University Defence Research Centre in Signal Processing and the BBC Audio Research Partnership.

His current research interests are in the areas of blind signal processing, machine audition (listening), audio-visual signal processing, sparse signal processing, machine learning, and perception. His research is funded by the Engineering and Physical Sciences Research Council, Ministry of Defence, Defence Sciences and Technology Laboratory, Home Office, Royal Academy of Engineering, and the University Research Support Fund.