Weighted SimCO: A Novel Algorithm for Dictionary Update

Xiaochen Zhao*, Guangyu Zhou*, Wenwu Wang†, Wei Dai*
*Department of Electrical and Electronic Engineering, Imperial College London
†Department of Electronic Engineering, University of Surrey
{xiaochen.zhao10, g.zhou11, wei.dai1}@imperial.ac.uk
w.wang@surrey.ac.uk

***Abstract*—Algorithms aiming at solving dictionary learning problem usually involve iteratively performing two stage operations: sparse coding and dictionary update. In the dictionary update stage, codewords are updated based on a given sparsity pattern. In the ideal case where there is no noise and the true sparsity pattern is known a priori, dictionary update should produce a dictionary that precisely represent the training samples. However, we analytically show that benchmark algorithms, including MOD, K-SVD and regularized SimCO, could not always guarantee this property: they may fail to converge to a global minimum. The key behind the failure is the singularity in the objective function. To address this problem, we propose a weighted technique based on the SimCO optimization framework, hence the term weighted SimCO. Decompose the overall objective function as a sum of atomic functions. The crux of weighted SimCO is to apply weighting coefficients to atomic functions so that singular points are zeroed out. A second order method is implemented to solve the corresponding optimization problem. We numerically compare the proposed algorithm with the benchmark algorithms for noiseless and noisy scenarios. The empirical results demonstrate the significant improvement in the performance.**

I. Introduction

Dictionary learning is a technique to find an over-complete dictionary, from which the signals can be approximated from a linear combination of only a few columns, which are often referred to as codewords. A good dictionary plays a decisive role in sparse signal representation, which recently receives wide attention in a large number of applications, such as signal denoising, image inpainting and data classification. While pre-defined dictionaries have been widely adopted in many applications (e.g. DCT and wavelet dictionaries for image compression), learning a dictionary directly from data often leads to a better sparse representation, and has been successful in the applications where pre-defined dictionaries are not available, for example, blind source separation or certain military applications. Usually benchmark algorithms addressing the dictionary learning problem focus on solving two problems iteratively: sparse coding and dictionary update.

Sparse coding aims at finding an optimal sparse linear coefficients to approximating the training samples by linearly combining codewords from a given dictionary. Algorithms developed for sparse coding include Basis Pursuit (BP) [1], Matching Pursuit (MP) [2], Orthogonal Matching Pursuit (OMP) [3], Subspace Pursuit (SP) [4], Gradient Pursuit (GP) [5], etc.

Used in combination with these sparse coding algorithms, MOD [6] and K-SVD [7] are two early efficient benchmark mechanisms for dictionary update. Method of Optimal Directions (MOD), designed by Engan et al. [6], updates dictionary in each iteration by fixing the sparse coefficients, i.e., in dictionary update stage, only the dictionary is updated. Aharon et al. [7] generalized the K-means algorithm and developed the so called K-SVD algorithm. K-SVD considers a structure where each time a single codeword is simultaneously updated with the corresponding row of the sparse coefficient matrix. Each column and row update is actually a singular value decomposition calculation. Simultaneous Codeword Optimization (SimCO), a very recent algorithm designed by Dai, et al. [8], allows simultaneously updating a few specified codewords with the corresponding rows of the sparse coefficients. SimCO can also be considered as a generalization of both MOD and K-SVD.

In [8], the authors found that ill-conditioned dictionary may occur during the dictionary update process which leads the algorithms to fail to converge to a global minimum. To address this problem, Dai, et al. proposed Regularized SimCO. The main idea is to add l_2 norm of the sparse coefficients as a penalty term to the objective function of dictionary update. Numerical results verified that regularized SimCO improves the successful rate of the dictionary update.

In this paper, we proposed a weighted technique based on SimCO framework to address the singularity problem. The new algorithm admits the following advantages.

- The crux of weighted SimCO is to apply weighting coefficients to atomic functions so that singular points are zeroed out and learning performance is improved.
- A second order method is implemented to solve the corresponding optimization problem to achieve fast convergence rate.

The rest of this paper is arranged as following: Section II introduces the SimCO framework which plays a crucial role in explaining the singularity problem during the dictionary update process. Then in Section III, an explicit example is constructed to explain the failure of the benchmark algorithms. As our solution on this dilemma, weighted SimCO is introduced in Section IV. Section V gives a pseudo-code of the

second order method which is ready for speeding up our novel algorithm and some derived results are appended. In Section VI, numerical comparisons between the new algorithm and the benchmark algorithms are presented in several scenarios. Finally, we conclude our work in the last section.

II. Dictionary Learning and the Framework of SimCO

In Dictionary learning problem, let $\boldsymbol{Y} \in \mathbb{R}^{m\times n}$ be the training data, where the columns represent for the training signals. One needs to find a dictionary $\boldsymbol{D} \in \mathbb{R}^{m\times d}$ and sparse coefficient matrix $\boldsymbol{X} \in \mathbb{R}^{d\times n}$ so that the objective function is minimized. i.e.,

$$\min_{\boldsymbol{D},\boldsymbol{X}} \|\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X}\|_F^2, \text{ s.t. } \forall i,\ \|\boldsymbol{D}_{:,i}\|_2^2 = 1,$$

where $\|\cdot\|_F^2$ represents for the Frobenius norm. Each column of $\boldsymbol{D}$ is termed as a codeword in the dictionary and is usually normalized. In practical applications the dictionaries are generally over-complete ($m < d$), which results in the non-unique solution unless certain constraints are posed. One most widely used constraint is that $\boldsymbol{X}$ is sparse, i.e., a large number of entries in the matrix $\boldsymbol{X}$ are zero.

Dictionary Learning algorithms usually includes two stages: sparse coding and dictionary update. In sparse coding stage, large number of l_1 minimization and greedy algorithms (e.g. BP [1], OMP [3] and SP [4]) have been designed in order to solve the same question: fix dictionary $\boldsymbol{D}$ and update coefficient $\boldsymbol{X}$ by $\min_{\boldsymbol{X}} \|\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X}\|_F^2$. However algorithms on dictionary update concern about different targets to update. MOD [6] focuses on updating $\boldsymbol{D}$ by fixing $\boldsymbol{X}$ and in K-SVD [7] each codeword of the dictionary is updated with the corresponding row of the sparse coefficients at one time. SimCO, as the name suggests, allows a few codewords with the corresponding rows in sparse coefficients being updated simultaneously.

Let set $\mathcal{I}$ represent for the indexes of the codewords chosen to be updated, i.e., $\boldsymbol{D}_{:,\mathcal{I}}$ is the sub-matrix of $\boldsymbol{D}$. One can 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 the optimization formulation of SimCO can be written as

$$\min_{\boldsymbol{D}_{:,\mathcal{I}},\boldsymbol{X}_{:,\mathcal{I}}} \|\boldsymbol{Y}_r - \boldsymbol{D}_{:,\mathcal{I}}\boldsymbol{X}_{\mathcal{I},:}\|_F^2.$$

In SimCO, if the sparse pattern is known, the optimal sparse coefficient for any well-conditioned dictionary can be immediately obtained. Denoted Ω as the index set of non-zero elements in $\boldsymbol{X}$, $\Omega(:,j)$ represents for the index set of non-zero elements in $\boldsymbol{X}_{:,j}$. Then for a given sparse pattern of $\boldsymbol{X}$, the optimal $\boldsymbol{X}^*$ can be calculated by

$$\begin{cases} \boldsymbol{X}^*_{\Omega(:,j),j} = \boldsymbol{D}^{\dagger}_{:,\Omega(:,j)}\boldsymbol{Y}_{:,j} \\ \boldsymbol{X}^*_{i,j} = 0 \qquad\qquad\qquad\qquad \forall (i,j)\notin\Omega \end{cases}.$$

$\boldsymbol{D}^{\dagger}$ is the pseudo-inverse of matrix $\boldsymbol{D}$. Thus the objective function of SimCO can be represented as a function of $\boldsymbol{D}$,

$$f(\boldsymbol{D}) = \|\boldsymbol{Y}_r - \boldsymbol{D}_{:,\mathcal{I}}\boldsymbol{X}^*_{\mathcal{I},:}\|_F^2.$$

Therefore, simultaneously updating $\boldsymbol{D}_{:,\mathcal{I}}$ and $\boldsymbol{X}_{:,\mathcal{I}}$ through line search method along the gradient descent direction becomes applicable, where the gradient is computed by

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

Consider the more common situation, when the whole dictionary is to be updated, the gradient is simplified to

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

III. An Explicit Example

In this section, we construct an explicit example to show how singularity affects the dictionary update stage. The example is designed in such a way that MOD, K-SVD, SimCO and regularized SimCO will fail. In particular, we consider an under-determined dictionary in this example, which may not be practical. Nevertheless the analysis is applicable to the general cases. Due to the space constraints, only main results are given in this paper, while the detailed analysis will be presented in its journal version [10].

Consider a dictionary learning problem where a dictionary will be trained from given data $\boldsymbol{Y}$ and a prior given true sparsity pattern $\boldsymbol{\Omega}$, which is, for the sake of notation simplification, denoted by the '0-1' matrix below:

$$\boldsymbol{Y} = \begin{bmatrix} 1 & 0 & 0.7 & 0 \\ 0 & 1 & 0.7 & 0 \\ 0 & 0 & -0.1 & 1 \\ 0 & 0 & -0.1 & 1 \end{bmatrix},\ \boldsymbol{\Omega} = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix}.$$

Let the dictionary $\boldsymbol{D} \in \mathbb{R}^{4\times 3} : \|\boldsymbol{D}_{:,i}\|_2 = 1,\ i = 1,2,3$. And the sparse signal $\boldsymbol{X} \in \mathbb{R}^{3\times 4} : \boldsymbol{X}_{i,j} = 0,\ \forall (i,j)\in\boldsymbol{\Omega}^c$. One needs to find optimal dictionary $\boldsymbol{D}^*$ and $\boldsymbol{X}^*$, s.t. $f(\boldsymbol{D}^*) = \|\boldsymbol{Y}-\boldsymbol{D}^*\boldsymbol{X}^*\|_F^2 = 0$, i.e., solve the optimization problem

$$\min_{\boldsymbol{D}}\min_{\boldsymbol{X}} \|\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X}\|_F^2.$$

In this example, the optimal $\boldsymbol{D}^*$ and its corresponding $\boldsymbol{X}^*$ can be obtained by checking the front three columns of $\boldsymbol{Y}$ and $\boldsymbol{\Omega}$:

$$\boldsymbol{D}^* = \begin{bmatrix} 1 & 0 & 0.7 \\ 0 & 1 & 0.7 \\ 0 & 0 & -0.1 \\ 0 & 0 & -0.1 \end{bmatrix},\ \boldsymbol{X}^* = \begin{bmatrix} 1 & 0 & 0 & 7 \\ 0 & 1 & 0 & 7 \\ 0 & 0 & 1 & -10 \end{bmatrix}.$$

In order to facilitate the discussion, we consider the case that $\boldsymbol{D}^*_{:,1:2}$ is known a priori. Hence, the optimization is only with respect to $\boldsymbol{D}_{:,3}$, i.e., the third column of $\boldsymbol{D}$.

$$\min_{\boldsymbol{D}_{:,3},\boldsymbol{X}|\boldsymbol{X}_{\boldsymbol{\Omega}^C}=0} \|\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X}\|_F^2,\ s.t.\ \|\boldsymbol{D}_{:,3}\|_2 = 1,\ \boldsymbol{D}_{:,1:2} = \boldsymbol{D}^*_{:,1:2}. \tag{1}$$

We denote $\boldsymbol{D}^0$ as the initial dictionary and $\boldsymbol{D}^k$ as the updated dictionary after the k^{th} iteration. And we suppose that $\boldsymbol{D}^0_{:,3}$ and $\boldsymbol{D}^k_{:,3}$ are of the form

$$\boldsymbol{D}^0_{:,3} = \begin{bmatrix} \sqrt{1-\epsilon_0^2} & \sqrt{1-\epsilon_0^2} & \epsilon_0 & \epsilon_0 \end{bmatrix}^T/\sqrt{2},\ \epsilon_0\in(0,1),$$

$$\boldsymbol{D}^k_{:,3} = \begin{bmatrix} \sqrt{1-\epsilon_k^2} & \sqrt{1-\epsilon_k^2} & \epsilon_k & \epsilon_k \end{bmatrix}^T/\sqrt{2},\ \epsilon_k\in(-1,1),$$

such that $\left\|\boldsymbol{D}_{:,3}^k\right\|_2^2 = 1$. Note that $\boldsymbol{D}^k$ is fully determined by ϵ_k. In our analysis, we only track ϵ_k. Further more when $\epsilon_k = 0$, $\boldsymbol{D}^k$ becomes column rank deficient and the objective function (1) becomes singular.

The analytical results of benchmark algorithms are presented in the following. Note that the optimization problem at hand (1) is sightly different from that for the general setting considered in [6], [7] and [8]. Adaptations of benchmark algorithms have to be made. Nevertheless, the insight and conclusions obtained for this particular example can be applied to the general case.

(a) $MOD\,algorithm$. MOD employs an alternative optimization procedure: first fix the dictionary and optimize the sparse coefficients; then fix the sparse coefficients and optimize the dictionary. We derived that after the kth iteration, the updated dictionary $\boldsymbol{D}^k$ will have

$$\epsilon_k = \epsilon_{k-1}\left(1 - 0.07\epsilon_{k-1} - 0.48\epsilon_{k-1}^2 + o(\epsilon_{k-1}^2)\right),$$

where $\epsilon_k \in (0, 1)$ since it always keep the sign after each iteration. This implies when $k \to \infty$, ϵ_k converges to some value between $(0, \epsilon_0)$ rather than $-0.1\sqrt{2}$. i.e., finally the dictionary $\boldsymbol{D}$ will not converge to $\boldsymbol{D}^*$.

(b) $K\text{-}SVD\,algorithm$. Use the same initializations as in the MOD case. Because $\boldsymbol{D}_{:,1:2}^0\boldsymbol{X}_{1:2,:}^0$ is already optimized, it is reasonable to just consider optimizing $\boldsymbol{D}_{:,3}^0\boldsymbol{X}_{3,:}^0$ and $\boldsymbol{D}_{:,4}^0\boldsymbol{X}_{4,:}^0$. Furthermore, notice that $(\boldsymbol{Y} - \boldsymbol{D}_{:,1:2}^0\boldsymbol{X}_{1:2,:}^0)_{:,1:2}$ is zero, we can just look at the term $(\boldsymbol{Y} - \boldsymbol{D}_{:,1:2}^0\boldsymbol{X}_{1:2,:}^0)_{:,3:4}$and find its best approximation. The third column of the dictionary can be updated through finding its largest right singular vector. For small positive ϵ_k, the dictionaries before and after the k^{th} iteration: $\boldsymbol{D}^{k-1}$ and $\boldsymbol{D}^k$, coincidentally has

$$\epsilon_k = \epsilon_{k-1}\left(1 - 0.07\epsilon_{k-1} - 0.48\epsilon_{k-1}^2 + o(\epsilon_{k-1}^2)\right).$$

Therefore with the same analysis in the MOD case, K-SVD does not make the dictionary converge to $\boldsymbol{D}^*$, neither.

(c) $SimCO\,\&\,Regularized\,SimCO$. In SimCO on solves the optimization (1) directly by using optimization methods, for example, the gradient descent method, on manifolds. Realizing the singularity problem, the authors in [8] proposed to solve the regularized optimization problem

$$\begin{aligned}\min_{\boldsymbol{D}} f_\mu(\boldsymbol{D}) = \min_{\boldsymbol{D}}\min_{\boldsymbol{X}} &\|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2 + \mu\|\boldsymbol{X}\|_F^2 \\ s.t.\,\forall i,\ &\|\boldsymbol{D}_{:,i}\|_2 = 1,\ \boldsymbol{\Omega}_{\boldsymbol{X}^C} = 0. \end{aligned} \tag{2}$$

Numerical results demonstrate the empirical performance improvement. Note that ideally one needs to solve a series optimization problems (2) where $\mu \to 0$.

Let $\boldsymbol{X}^*$ be the optimal solution of $f_\mu(\boldsymbol{D})$ for a given $\boldsymbol{D}$. It can be verified that

$$\begin{aligned} f_\mu(\boldsymbol{D}^k) &= \min_{\boldsymbol{X}}\|\boldsymbol{Y} - \boldsymbol{D}^k\boldsymbol{X}^*\|_F^2 + \mu\|\boldsymbol{X}_{:,3:4}^*\|_F^2 \\ &= 3 - \frac{2\epsilon_k^2}{\mu + 1 - \frac{1-\epsilon_k^2}{\mu+1}} - 2(0.7\sqrt{1-\epsilon_k^2} - 0.1\epsilon_k)^2. \end{aligned}$$

It can be shown [10] that whenever $\epsilon_k \in (0, 1\sqrt{2})$ and $\mu \in (0, \min(\sqrt{1+100\epsilon_k^2}, \sqrt{2}-1))$, there always exist a $\bar{\epsilon} \in (0, \epsilon_k)$ with its regularized ovjective function being larger than $f_\mu(\boldsymbol{D}^k)$. Hence, the infinitesimal gradient descent method (the true gradient descent method) cannot pass through the singular point where $\boldsymbol{D}_{:,3} = \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix}^T/\sqrt{2}$ to reach the global optimizer where $\boldsymbol{D}_{:,3}^* = \begin{bmatrix} 0.7 & 0.7 & -0.1 & -0.1 \end{bmatrix}^T$. $\boldsymbol{D}$ finally converges the singular point as $\mu \to 0$.

To conclude, when the updated dictionary $\boldsymbol{D}$ is close to be column rank deficient, i.e., in a neighborhood of a singular point on the manifolds, after iterations of the benchmark optimization algorithms, it will come increasingly closer to the singular point.

IV. A Weighted Technique

In this section, a weighted technique is developed to address the singularity problem.

Write the objective function of the dictionary learning problem as a summation of atomic functions.

$$\begin{aligned} f(\boldsymbol{D}) &= \|\boldsymbol{Y} - \boldsymbol{D}\boldsymbol{X}\|_F^2 \\ &= \sum_i \|\boldsymbol{Y}_{:,i} - \boldsymbol{D}_{:,\boldsymbol{\Omega}(:,i)}\boldsymbol{X}_{\boldsymbol{\Omega}(:,i),i}\|_F^2 \\ &= \sum_i f_i(\boldsymbol{D}_{:,\boldsymbol{\Omega}(:,i)}), \end{aligned}$$

where $f_i(\boldsymbol{D}_{:,\boldsymbol{\Omega}(:,i)})$ is referred to as an atomic function. Because of the sparsity, $\boldsymbol{D}_{:,\boldsymbol{\Omega}(:,i)}$ is usually a tall matrix. Singular points of dictionary learning objective function correspond to the ill-conditioned $\boldsymbol{D}_{:,\boldsymbol{\Omega}(:,i)}$. For an ill-conditioned $\boldsymbol{D}_{:,\boldsymbol{\Omega}(:,i)}$, its minimum singular value is much smaller than the maximum one [9]. An extreme situation is the column rank deficient $\boldsymbol{D}_{:,\boldsymbol{\Omega}(:,i)}$, where the minimum singular value is zero. Based on this property, we are able to determine whether $\boldsymbol{D}_{:,\boldsymbol{\Omega}(:,i)}$ is near a singular point by checking how close its minimum singular value is to zero. Notice that it is the summation of all atomic functions to represent the objective function, which implies that even some of the atomic functions have singularity, the rest can still have good update. As a result, utilizing weighting functions of $\boldsymbol{D}_{:,\boldsymbol{\Omega}(:,i)}$ to mitigate the influence degree of atomic functions with ill-conditioned $\boldsymbol{D}_{:,\boldsymbol{\Omega}(:,i)}$ is feasible.

Let λ_r be the minimum singular value of $\boldsymbol{D}_{:,\boldsymbol{\Omega}(:,i)}$. Let $0 \leqslant \delta_d \leqslant \delta_u \leqslant \frac{\pi}{2}$ be two threshold constants. We want a weighting coefficient w_i to have a directly reflection of the singularity of $\boldsymbol{D}_{:,\boldsymbol{\Omega}(:,i)}$: when $\lambda_r < \delta_d$ which means $\boldsymbol{D}_{:,\boldsymbol{\Omega}(:,i)}$ is very close to a singular point, we set w_i to zero; when $\lambda_r > \delta_u$ which means $\boldsymbol{D}_{:,\boldsymbol{\Omega}(:,i)}$ is well conditioned, we set w_i to 1; when $\delta_d \leqslant \lambda_r \leqslant \delta_u$, the weighting coefficient is chosen between 0 and 1. To better define w_i, we introduce notation $\boldsymbol{D}_{:,\boldsymbol{\Omega}(:,i)}^k$ to be distinguished from $\boldsymbol{D}_{:,\boldsymbol{\Omega}(:,i)}$, where $\boldsymbol{D}_{:,\boldsymbol{\Omega}(:,i)}^k$ is the output dictionary after the kth dictionary update iteration. w_i is a coefficient of $\boldsymbol{D}_{:,\boldsymbol{\Omega}(:,i)}^k$ and is updated only at the start of each dictionary update stage, i.e., it is fixed during the each dictionary update iteration. Then a choice of w_i is specified as follows,

$$w_i = \begin{cases} 0 & a \leqslant 0 \\ 6a^5 - 15a^4 + 10a^3 & 0 < a < 1 \\ 1 & a \geqslant 1 \end{cases},$$

where a is a function of $\boldsymbol{D}^k_{:,\boldsymbol{\Omega}(:,i)}$,

$$a\left(\boldsymbol{D}^k_{:,\boldsymbol{\Omega}(:,i)}\right)=\frac{\lambda_r-\delta_d}{\delta_u-\delta_d}.$$

Apply the weighting coefficient to all atomic functions, the modified problem formulation becomes

$$\sum_i \min_{\boldsymbol{D}_{:,\boldsymbol{\Omega}(:,i)},\boldsymbol{X}_{\boldsymbol{\Omega}(:,i),i}} \|\boldsymbol{Y}_{:,i}-\boldsymbol{D}_{:,\boldsymbol{\Omega}(:,i)}\boldsymbol{X}_{\boldsymbol{\Omega}(:,i),i}\|_F^2\cdot w_i.$$

For any i, with the decreasing of the minimum singular value of $\boldsymbol{D}^k_{:,\boldsymbol{\Omega}(:,i)}$, the w_i is reducing the proportion of ith atomic function relative to the overall objective function. For some minimum singular being very close to zero, the proportion is reduced to zero. While for some $\boldsymbol{D}^k_{:,\boldsymbol{\Omega}(:,i)}$ with large minimum singular value, w_i maintains one which brings no impact. Therefore in the framework of this weighted model, singular points will not appear on the optimizing path.

V. A Newton Method

Second order information helps numerical optimization algorithm find better directions than the gradient descent method so as to accelerate the convergence to a large extent. In particular, a line search Newton CG method is adapted to solve the weighted SimCO problem.

In Newton CG method the searching path is determined not only by the gradient but the Hessian as well. However, note that the Hessian of $f(\boldsymbol{D})$ has size $(m\times d)\times(m\times d)$ and the overall computational cost is huge. Therefore we consider conjugate gradient method to avoid the expensive computations. Use notation $\boldsymbol{D}_i$ to represent $\boldsymbol{D}_{:,\Omega(:,i)}$ to simplify the expressions. Referring to [9], one needs to calculate the Hessian of the function $\sum_i f_i(\boldsymbol{D}_i)\cdot w_{i_i}$ along some searching direction $\boldsymbol{\eta}$, denoted as $\nabla_{\boldsymbol{\eta}}\left(\nabla\left(f_i(\boldsymbol{D}_i)\cdot w_i\right)\right)$, where $\nabla_{\boldsymbol{\eta}}(\cdot)$ is the directional derivative [11] along $\boldsymbol{\eta}$. $\boldsymbol{D}_i\in\mathbb{R}^{m\times s}$ and $\boldsymbol{\eta}\in\mathbb{R}^{m\times s}$, $m>s$.

Let $f_i(\boldsymbol{D}_i)=\|\boldsymbol{y}_i-\boldsymbol{D}_i\boldsymbol{x}_i\|_2^2$, where $\boldsymbol{y}_i=\boldsymbol{Y}_{:,i}$ and $\boldsymbol{x}_i=\boldsymbol{X}_{:,i}$. Note that w_i only changes at the start of each dictionary update iteration. Term $\nabla\left(f_i(\boldsymbol{D}_i)\cdot w_i\right)$ and $\nabla_{\boldsymbol{\eta}}\left(\nabla\left(f_i(\boldsymbol{D}_i)\cdot w_i\right)\right)$ can be written as

$$\nabla\left(f_i(\boldsymbol{D}_i)\cdot w_i\right)=\sum_i\nabla f_i(\boldsymbol{D}_i)\cdot w_i,$$

$$\nabla_{\boldsymbol{\eta}}\left(\nabla\left(f_i(\boldsymbol{D}_i)\cdot w_i\right)\right)=\sum_i\nabla_{\boldsymbol{\eta}}\left(\nabla f_i(\boldsymbol{D}_i)\right)\cdot w_i.$$

Denote $\boldsymbol{D}_i^\dagger$ as the pseudo-inverse of $\boldsymbol{D}_i$. The optimal $\boldsymbol{x}_i=\boldsymbol{D}_i^\dagger\boldsymbol{y}_i$. Then $\nabla f_i(\boldsymbol{D}_i)$ can be written as

$$\nabla f_i(\boldsymbol{D}_i)=\frac{\partial f_i}{\partial\boldsymbol{D}_i}+\frac{\partial f_i}{\partial\boldsymbol{x}_i}\frac{\partial\boldsymbol{x}_i}{\partial\boldsymbol{D}_i}=-2\left(\boldsymbol{y}_i-\boldsymbol{D}_i\boldsymbol{x}_i\right)\boldsymbol{x}_i^T+\boldsymbol{0}.$$

Also we are able to obtain $\nabla_{\boldsymbol{\eta}}\boldsymbol{x}_i$ and $\nabla_{\boldsymbol{\eta}}\left(\nabla f_i(\boldsymbol{D}_i)\right)$.

$$\begin{aligned}\nabla_{\boldsymbol{\eta}}\boldsymbol{x}_i&=\nabla_{\boldsymbol{\eta}}(\boldsymbol{D}_i^\dagger\boldsymbol{y}_i)\\&=\nabla_{\boldsymbol{\eta}}\left((\boldsymbol{D}_i^T\boldsymbol{D}_i)^{-1}\right)\boldsymbol{D}_i^T\boldsymbol{y}_i+(\boldsymbol{D}_i^T\boldsymbol{D}_i)^{-1}\nabla_{\boldsymbol{\eta}}\boldsymbol{D}_i^T\boldsymbol{y}_i\\&=-(\boldsymbol{D}_i^T\boldsymbol{D}_i)^{-1}\left(\left(\boldsymbol{D}_i^T\boldsymbol{\eta}+\boldsymbol{\eta}^T\boldsymbol{D}_i\right)\boldsymbol{D}_i^\dagger-\boldsymbol{\eta}_i^\dagger\right)\boldsymbol{y}_i,\end{aligned}$$

$$\begin{aligned}\nabla_{\boldsymbol{\eta}}\left(\nabla f_i(\boldsymbol{D}_i)\right)&=2\nabla_{\boldsymbol{\eta}}\left(\boldsymbol{D}_i\boldsymbol{x}_i-\boldsymbol{y}_i\right)\boldsymbol{x}_i^T+2\left(\boldsymbol{D}_i\boldsymbol{x}_i-\boldsymbol{y}_i\right)\nabla_{\boldsymbol{\eta}}\boldsymbol{x}_i^T\\&=2\left(\boldsymbol{\eta}\boldsymbol{x}_i+\boldsymbol{D}_i\nabla_{\boldsymbol{\eta}}\boldsymbol{x}_i\right)\boldsymbol{x}_i^T+2\left(\boldsymbol{D}_i\boldsymbol{x}_i-\boldsymbol{y}_i\right)\nabla_{\boldsymbol{\eta}}\boldsymbol{x}_i^T.\end{aligned}$$

In the following a detailed pseudo-code of the Newton CG algorithm is listed in Algorithm 1. The line search method on deciding the step size is listed in Algorithm 2.

Algorithm 1 Line search Newton CG algorithm

Input: $\nabla f_0=\nabla f(\boldsymbol{D})$, $\boldsymbol{D}$, x_0.
Output: $\bar{\boldsymbol{\eta}}$.
For $k=0,\ 1,\ 2,\ ...$
  Define tolerance $\epsilon_k=\min(0.5,\ \sqrt{\|\nabla f_k\|})\,\|\nabla f_k\|$.
  Set $\boldsymbol{z}_0=\boldsymbol{0}$, $\boldsymbol{r}_0=\nabla f_k$, $\boldsymbol{d}_0=-\boldsymbol{r}_0=-\nabla f_k$.
  For $\forall i$, $\overline{\nabla_{\boldsymbol{d}_j}\nabla f_k}_{:,i}=(\boldsymbol{I}-\boldsymbol{D}_{:,i}\boldsymbol{D}_{:,i}^T)\boldsymbol{\eta}_{:,i}$.
  For $j=0,\ 1,\ 2,\ ...$
    If $\boldsymbol{d}_j^T\left(\overline{\nabla_{\boldsymbol{d}_j}\nabla f_k}\right)\leqslant 0$
      If $j=0$
        return $\boldsymbol{\eta}=-\nabla f_k$.
      else
        return $\boldsymbol{\eta}=\boldsymbol{z}_j$.
    Set $\alpha_j=\boldsymbol{r}_j^T\boldsymbol{r}_j/\boldsymbol{d}_j^T\left(\overline{\nabla_{\boldsymbol{d}_j}\nabla f_k}\right)$.
    Set $\boldsymbol{r}_{j+1}=\boldsymbol{r}_j+\alpha_j\left(\overline{\nabla_{\boldsymbol{d}_j}\nabla f_k}\right)$.
    If $\|\boldsymbol{r}_{j+1}\|<\epsilon_k$
      return $\boldsymbol{\eta}=\boldsymbol{z}_{j+1}$.
    Set $\beta_{j+1}=\boldsymbol{r}_{j+1}^T\boldsymbol{r}_{j+1}/\boldsymbol{r}_j^T\boldsymbol{r}_j$.
    Set $\boldsymbol{d}_{j+1}=-\boldsymbol{r}_{j+1}+\beta_{j+1}\boldsymbol{d}_j$.
end
For $\forall i$, $\bar{\boldsymbol{\eta}}_{:,i}=(\boldsymbol{I}-\boldsymbol{D}_{:,i}\boldsymbol{D}_{:,i}^T)\boldsymbol{\eta}_{:,i}$.

Algorithm 2 Line search method on deciding the step size

Input: $f_0=f(\boldsymbol{D})$, ∇f_0,$\bar{\boldsymbol{\eta}}$, $\boldsymbol{D}^0=\boldsymbol{D}$, $t=1$, $c=1e-6$.
Output: $\boldsymbol{D}=\boldsymbol{D}^k$.
For $k=1,\ 2,\ 3,\ ...$
  Do compact SVD $\forall i$, $\bar{\boldsymbol{\eta}}_{:,i}=\boldsymbol{U}_i\boldsymbol{\Sigma}_i\boldsymbol{V}_i^T$.
  Compute $\forall i$, $\boldsymbol{D}^k_{:,i}=(\boldsymbol{D}^{k-1}_{:,i}\boldsymbol{V}_i,\boldsymbol{U}_i)\begin{pmatrix}cos\boldsymbol{\Sigma}_i\\ sin\boldsymbol{\Sigma}_i\end{pmatrix}\boldsymbol{V}_i^T$.
  If $f_k\leqslant f_{k-1}+c\cdot t\cdot\nabla f_{k-1}^T\bar{\boldsymbol{\eta}}$ (Armijo condition [9])
    return $\boldsymbol{D}^k$.
  Set $t^k=0.8t^{k-1}$.
end

VI. Empirical Study

The setting for the synthetic data tests is as follows. Training samples $\boldsymbol{Y}=\boldsymbol{D}_{true}\boldsymbol{X}_{true}$, where true dictionary $\boldsymbol{D}_{true}\in\mathbb{R}^{m\times d}$ and true sparse coefficients $\boldsymbol{X}_{true}\in\mathbb{R}^{d\times n}$ One supposes that each element of $\boldsymbol{D}_{true}$ and each non-zero element of $\boldsymbol{X}_{true}$ is randomly generated from standard Gaussian distribution. One also suppose that in each column of $\boldsymbol{X}_{true}$ there is s many non-zero elements, where the positions are randomly generated from the uniform distribution. In the numerical tests, we fix $m=16$, $d=32$ and $s=4$. The number of training samples varies from $n=30\sim120$. Compared with the new algorithm, the three benchmark algorithms are involved. In fairness, we set the maximum number of iterations of all the four algorithms equal to 1000. For regularized

SimCO, the initial regularization constant $\mu = 0.1$. Then for every 100 iterations, μ is decreased to $\mu/10$. For weighted SimCO, the threshold $[\delta_d,\ \delta_u]$ is initialized as $[1e-3,\ 2e-1]$. Numerical results show that if there are enough training samples, singular points will not appear in a small enough neighborhood of the global minimum. Therefore, we can zero the thresholds when the objective function is small enough so that the final objective function of weighted SimCO is equivalent to the original objective function. For simplicity, in our test the thresholds are zeroed after 500 iterations.

We show the noiseless case in Figure 1. Simulation results ($n = 60 \sim 120$) show that the average performance of weighted SimCO is comprehensively better than the rest three algorithms. In Figure 2, 20dB white Gaussian noise is added on the training samples $\boldsymbol{Y}$ and the rest setup is the same as the noiseless case. The figure shows that weighted SimCO and regularized SimCO have similar performance and meanwhile both better than MOD and K-SVD. Finally in Figure 3 we draw the successful rate of the four algorithms. We vary n from 30 to 84. When $n = 30$, the number of training samples is smaller than the number of codewords, which means that the each training sample can always be represented by only one codeword. Cases of $n = 36 \sim 84$ are the most difficult cases since the number of samples is slightly larger than the number of codewords. The average results show that weighted SimCO always keep the best performance of the four. And with the increase of the training samples, weighted SimCO is the first to reach 100% success rate ($n \geqslant 66$).

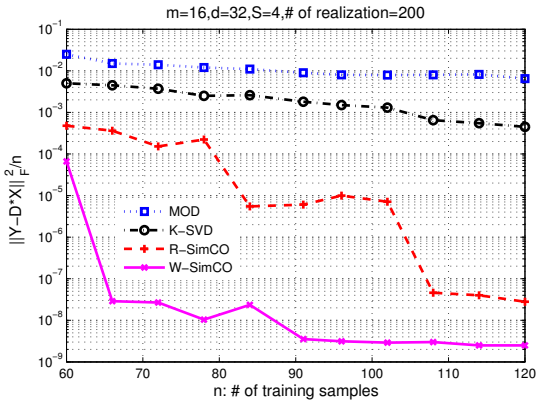

Figure 1. Noiseless Case

VII. Conclusion

In this paper, we showed the impact to the convergence of dictionary update caused by the ill-conditioned dictionaries during the optimization process. An explicit example was constructed with explanations on how the benchmark algorithms failed to find a global minimum. Analyzing the problem on the manifolds, the ill-condition happens corresponding to the singular points. To address this problem, we proposed weighted SimCO, which successfully avoids the optimization stagnating around singular points. Numerical results showed that the new algorithm maintains better performance and higher successful rate under dictionaries with different number of training samples.

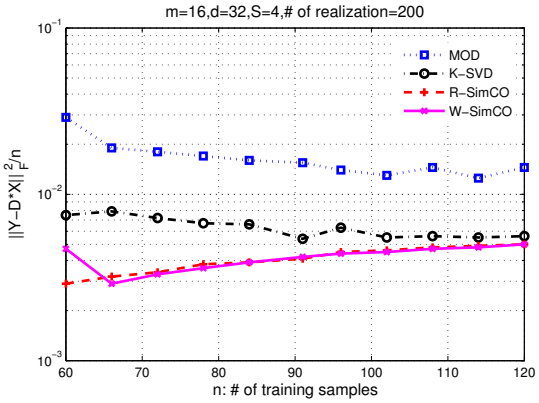

Figure 2. Noisy Case: SNR of training samples is 20dB.

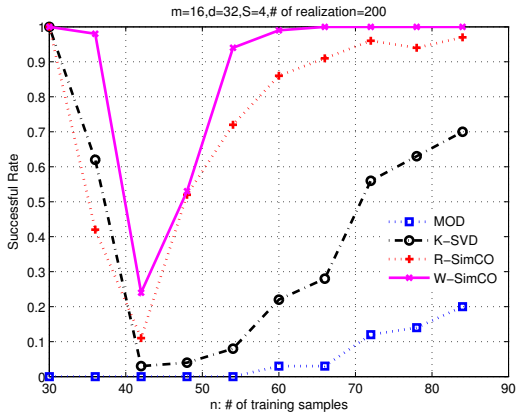

Figure 3. Success Probability of 200 realizations

References

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

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

[3] 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.

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

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

[6] 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.

[7] 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.

[8] W. Dai, T. Xu, and W. Wang, "Simultaneous codeword optimization (SimCO) for dictionary update and learning", IEEE Trans. Signal Process., 2012, accepted.

[9] J. Nocedal and S. J. Wright, "Numerical optimization", Springer, 2006.

[10] X. Zhao, G. Zhou and W. Dai, "Dictionary learning: a singularity problem and how to handle it", In preparation.

[11] F. B. Hildebrand, "Advanced calculus for applications". Prentice Hall, 1976.