DICTIONARY LEARNING AND UPDATE BASED ON SIMULTANEOUS CODEWORD OPTIMIZATION (SIMCO)

Wei Dai[⋆] *Tao Xu*[†] *Wenwu Wang*[†]

[⋆]Department of Electrical and Electonic Engineering, Imperial College London, London, United Kingdom
Email:wei.dai1@imperial.ac.uk
[†]Centre for Vision, Speech and Signal Processing, University of Surrey, Guildford, United Kingdom
Emails:[t.xu; w.wang]@surrey.ac.uk

ABSTRACT

Dictionary learning aims to adapt elementary codewords directly from training data so that each training signal can be best approximated by a linear combination of only a few codewords. Following the two-stage iterative processes: sparse coding and dictionary update, that are commonly used, for example, in the algorithms of MOD and K-SVD, we propose a novel framework that allows one to update an arbitrary set of codewords and the corresponding sparse coefficients simultaneously, hence termed *simultaneous codeword optimization (SimCO)*. Under this framework, we have developed two algorithms, namely the primitive and the regularized SimCO. Simulations are provided to show the advantages of our approach over the K-SVD algorithm in terms of both learning performance and running speed.

1. INTRODUCTION

The basic assumption underlying sparse coding is that a natural signal can be approximated by the combination of only a small number of elementary components, called *codewords* or *atoms*, chosen from a dictionary (i.e., the collection of all the codewords). The issue of dictionary design is of practical importance in many applications. As compared with predefined dictionaries based on e.g. discrete cosine transform (DCT), dictionaries learned from training data have the potential to offer better performance, as the codewords are derived to capture the salient information directly from the signals.

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

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

where the matrices $\boldsymbol{D}$ and $\boldsymbol{X}$ are often referred to as the dictionary and the corresponding coefficients respectively, and $\boldsymbol{D}_{:,i}$ denotes the i^{th} codeword of the dictionary. The problem is usually solved via a two-stage iterative process: sparse coding and dictionary update, such as in the well-known algorithms of MOD [1] and K-SVD [2], among many others.

In this paper, we focus on the dictionary update stage and propose a novel framework for dictionary learning. Specifically, we formulate dictionary update as an optimization problem on manifolds. Different from the existing algorithms, such as the MOD and the K-SVD, this framework allows us to update an *arbitrary* subset of the codewords and the corresponding coefficients simultaneously, hence termed *simultaneous codeword optimization (SimCO)*. We develop two algorithms: the primitive and the regularized SimCO (in Sections 2 and 3 respectively). We study numerically (in Section 4) the problem of ill-conditioned dictionary associated with the K-SVD and the primitive SimCO, and show that the regularized SimCO can overcome such a problem. We also provide empirical results (in Section 4) to show the advantages of the proposed technique.

2. PRIMITIVE SIMCO

The goal of the sparse coding stage in dictionary learning 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 [3] or greedy algorithms, e.g., the OMP [4] and SP [5] algorithms. The focus of this paper is on the dictionary update stage. Different from the MOD and K-SVD algorithms, the key characteristic of our approach is to update the arbitrary set of codewords and the corresponding non-zero coefficients *simultaneously*. Similar to K-SVD, however, we fix the sparsity pattern, which refers to the support set $\Omega \subset [d]\times[n]$ containing the indices of non-zero entries in $\boldsymbol{X}$, i.e., $X_{i,j} \neq 0$ for all $(i,j)\in\Omega$ and $X_{i,j} = 0$ for all $(i,j)\notin\Omega$. In dictionary learning algorithms, $\Omega \subset [d]\times[n]$ is often obtained from the sparse coding stage. Let $\mathcal{I} \subset [d]$ be an index set, $\boldsymbol{D}_{:,\mathcal{I}}$ denote the sub-matrix of $\boldsymbol{D}$ formed by the columns of $\boldsymbol{D}$ indexed by $\mathcal{I}$, and $\boldsymbol{X}_{\mathcal{I},:}$ be the sub-matrix of $\boldsymbol{X}$ consisting of the rows of $\boldsymbol{X}$ indexed by

$\mathcal{I}$. Then the optimization problem that only updates $\boldsymbol{D}_{:,\mathcal{I}}$ and $\boldsymbol{X}_{\mathcal{I},:}$ becomes

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

Define

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

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

where $\mathcal{I}^c$ is a set complementary to $\mathcal{I}$. It is clear that

$$f_{\mathcal{I}}(\boldsymbol{D}) = \min_{\boldsymbol{X}_{\mathcal{I},:}:\, X_{i,j}=0,\, \forall (i,j)\notin\Omega} \|\boldsymbol{Y}_r-\boldsymbol{D}_{:,\mathcal{I}}\boldsymbol{X}_{\mathcal{I},:}\|_F^2. \tag{3}$$

Hence, the optimization problem (2) can be written as

$$\min_{\boldsymbol{D}_{:,\mathcal{I}}} f_{\mathcal{I}}(\boldsymbol{D})\ \text{ subject to}\, \|\boldsymbol{D}_{:,i}\|_2 = 1,\ \forall i \in \mathcal{I}. \tag{4}$$

The gradient descent method is used to solve (4), which contains two steps: respectively gradient computation and line search. First, the gradient of $f_{\mathcal{I}}(\boldsymbol{D})$ with respect to $\boldsymbol{D}_{:,i}$, $i \in \mathcal{I}$, can be computed as

$$\begin{aligned}\nabla_{\boldsymbol{D}_{:,i}} f_{\mathcal{I}}(\boldsymbol{D}) &= -2\left(\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X}^*\right)_{:,\Omega(i,:)} \boldsymbol{X}^{*T}_{i,\Omega(i,:)} \\ &= -2\left(\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X}^*\right)\boldsymbol{X}^{*T}_{i,:}.\end{aligned} \tag{5}$$

where $\Omega(i,:) = \{j :\ (i,j)\in\Omega\}$ which gives the columns of $\boldsymbol{Y}$ whose sparse representation involves the codeword $\boldsymbol{D}_{:,i}$. The optimal $\boldsymbol{X}^*$ admits the following closed-from

$$\begin{aligned}&\boldsymbol{X}^*_{i,j} = 0,\ \forall (i,j)\notin\Omega,\ \boldsymbol{X}^*_{\mathcal{I}^c,:} = \boldsymbol{X}_{\mathcal{I}^c,:} \\ &\boldsymbol{X}^*_{\mathcal{I}\cap\Omega(:,j),j} = \boldsymbol{D}^{\dagger}_{:,\mathcal{I}\cap\Omega(:,j)}\left(\boldsymbol{Y}_r\right)_{:,j},\ \forall j \in [n],\end{aligned} \tag{6}$$

where the superscript † denotes the pseudo-inverse of a matrix.

Significantly different from the standard line search mechanism for the Euclidean space, we perform the line search over the product space of Grassmann manifolds, as it can be shown that $f_{\mathcal{I}}$ is indeed a function defined on the product of Grassmann manifolds. For convenience, we use the symbol $\boldsymbol{g}_i$ to denote $\nabla_{\boldsymbol{D}_{:,i}} f_{\mathcal{I}}(\boldsymbol{D})$, and further define

$$\bar{\boldsymbol{g}}_i = \boldsymbol{g}_i - \boldsymbol{D}_{:,i}\boldsymbol{D}^T_{:,i}\boldsymbol{g}_i,\ \forall i \in \mathcal{I}. \tag{7}$$

According to [6], $\bar{\boldsymbol{g}}_i$ is in fact the gradient of f with respect to $\boldsymbol{D}_{:,i}$ on the Grassmann manifold. The line search path for dictionary update, say $\boldsymbol{D}(t)$, $t \geq 0$, is therefore defined as

$$\begin{cases} \boldsymbol{D}_{:,i}(t) = \boldsymbol{D}_{:,i} & \text{if } i \notin \mathcal{I} \text{ or } \|\bar{\boldsymbol{g}}_i\|_2 = 0, \\ \boldsymbol{D}_{:,i}(t) = \boldsymbol{D}_{:,i}\cos\left(\|\bar{\boldsymbol{g}}_i\|_2 t\right) - \left(\bar{\boldsymbol{g}}_i/\|\bar{\boldsymbol{g}}_i\|_2\right)\sin\left(\|\bar{\boldsymbol{g}}_i\|_2 t\right) & \\ \qquad\qquad \text{if } i \in \mathcal{I} \text{ and } \|\bar{\boldsymbol{g}}_i\|_2 \neq 0. & \end{cases} \tag{8}$$

where $\boldsymbol{g}_i = \nabla_{\boldsymbol{D}_{:,i}} f_{\mathcal{I}}(\boldsymbol{D})$ is computed via (5).

3. REGULARIZED SIMCO

As will be detailed in Section 4.1, both K-SVD and the primitive SimCO may result in ill-conditioned dictionaries. We say the dictionary $\boldsymbol{D}$ is ill-conditioned with respect to the fixed sparsity pattern Ω if

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

for some $j \in [n]$. Here, the matrix $\boldsymbol{D}_{:,\Omega(:,j)}$ contains the codewords for representing $\boldsymbol{Y}_{:,j}$, and $\lambda_{\min}(\cdot)$ and $\lambda_{\max}(\cdot)$ give the smallest and largest singular values of a matrix, respectively. To mitigate the problem of the ill-conditioned $\boldsymbol{D}$, we propose to optimise a regularized objective function

$$\begin{aligned}&\widetilde{f}_{\mathcal{I}}(\boldsymbol{D}) \\ &= \min_{\boldsymbol{X}_{\mathcal{I},:}:\, X_{i,j}=0,\, \forall (i,j)\notin\Omega} \left(\|\boldsymbol{Y}-\boldsymbol{D}\boldsymbol{X}\|_F^2 + \mu\|\boldsymbol{X}_{\mathcal{I},:}\|_F^2\right) \\ &= \sum_{j=1}^{n} \underbrace{\min_{\boldsymbol{X}_{\mathcal{I}\cap\Omega(:,j),j}} \left(\left\|\left(\boldsymbol{Y}_r\right)_{:,j} - \boldsymbol{D}_{:,\mathcal{I}\cap\Omega(:,j)}\boldsymbol{X}_{\mathcal{I}\cap\Omega(:,j),j}\right\|_2^2 + \mu\left\|\boldsymbol{X}_{\mathcal{I}\cap\Omega(:,j),j}\right\|_2^2\right)}_{\widetilde{f}_{\mathcal{I},j}(\boldsymbol{D})}.\end{aligned} \tag{9}$$

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

As compared with the primitive SimCO in Section 2, the only changes need to be made for the regularized SimCO are the computation of $\widetilde{f}_{\mathcal{I}}(\boldsymbol{D})$ and the corresponding gradient $\nabla_{\boldsymbol{D}}\widetilde{f}_{\mathcal{I}}(\boldsymbol{D})$. Let $m_j = |\mathcal{I}\cap\Omega(:,j)|$. It is clear that $\boldsymbol{D}_{:,\boldsymbol{X}_{\mathcal{I}\cap\Omega(:,j),j}} \in \mathbb{R}^{m\times m_j}$ and $\boldsymbol{X}_{\mathcal{I}\cap\Omega(:,j),j} \in \mathbb{R}^{m_j}$. Define

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

where $\boldsymbol{0}_{m_j}$ is the zero vector of length m_j, and $\boldsymbol{I}_{m_j}$ is the $m_j \times m_j$ identity matrix. The optimal $\widetilde{\boldsymbol{X}}^*_{\mathcal{I}\cap\Omega(:,j),j}$ to solve (9) is given by

$$\widetilde{\boldsymbol{X}}^*_{\mathcal{I}\cap\Omega(:,j),j} = \widetilde{\boldsymbol{D}}^{\dagger}_j\widetilde{\boldsymbol{Y}}_{r,j} \tag{10}$$

Hence, $\widetilde{f}_{\mathcal{I}}(\boldsymbol{D})$ and $\nabla_{\boldsymbol{D}}\widetilde{f}_{\mathcal{I}}(\boldsymbol{D})$ are computed as follows

$$\widetilde{f}_{\mathcal{I}}(\boldsymbol{D}) = \left\|\boldsymbol{Y}_r - \boldsymbol{D}\widetilde{\boldsymbol{X}}^*\right\|_F^2 + \mu\cdot\left\|\widetilde{\boldsymbol{X}}^*_{\mathcal{I},:}\right\|_F^2. \tag{11}$$

$$\nabla_{\boldsymbol{D}_{:,\mathcal{I}}}\widetilde{f}_{\mathcal{I}}(\boldsymbol{D}) = -2\left(\boldsymbol{Y}-\boldsymbol{D}\widetilde{\boldsymbol{X}}^*\right)\widetilde{\boldsymbol{X}}^{*T}_{\mathcal{I},:}. \tag{12}$$

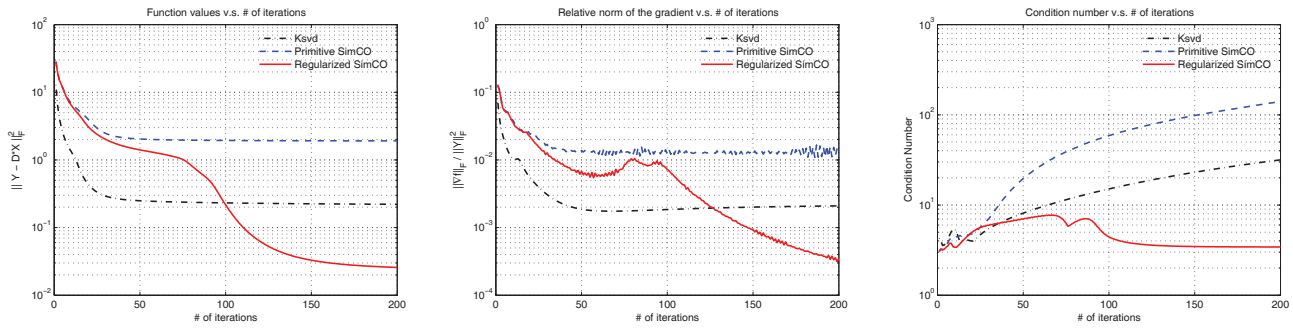


Fig. 1: Starting with the same point, the behaviors of the K-SVD, the primitive SimCO and the regularized SimCO are different.

The regularized SimCO is obtained by replacing (3) and (5) in the primitive SimCO with (11) and (12) respectively, while remaining other steps unchanged. If $\mu = 0$, the regularized SimCO reduces to the primitive one. As shown in the next section, the two versions of SimCO can be used jointly in practice.

4. EMPIRICAL TESTS

We numerically test the proposed algorithms, i.e., the primitive and the regularized SimCO, using synthetic data[1], and compare them with the baseline method K-SVD. To simplify the comparison, for both the primitive and the regularized SimCO, we set $\mathcal{I} = [d]$. In Section 4.1, we show that both the K-SVD and the primitive SimCO may result in an ill-conditioned dictionary while adding a regularized term can avoid this problem. Empirical experiments on synthetic data are detailed in Section 4.2. The results demonstrate the excellent learning performance of the regularized SimCO.

4.1. Ill-Conditioned Dictionaries

We handpick a particular example to show that both the K-SVD and the primitive SimCO may converge to an ill-conditioned dictionary. In the example, the training samples $\boldsymbol{Y} \in \mathbb{R}^{16\times 78}$ are computed via $\boldsymbol{Y} = \boldsymbol{D}_{\text{true}}\boldsymbol{X}_{\text{true}}$, where $\boldsymbol{D}_{\text{true}} \in \mathbb{R}^{16\times 32}$ is a dictionary, $\boldsymbol{X}_{\text{true}} \in \mathbb{R}^{32\times 78}$ is the corresponding sparse coefficient matrix, and each column of $\boldsymbol{X}$ contains exactly 4 nonzero elements. To test the performance of the three different algorithms, we randomly generate the initial dictionary $\boldsymbol{D}_0$ from the uniform distribution on the product of the Stiefel manifolds $\prod^{32}\mathcal{U}_{16,1}$, and the initial coefficient matrix $\boldsymbol{X}_0$ from the standard Gaussian distribution so that $\boldsymbol{X}_0$ and $\boldsymbol{X}$ have the exactly same sparsity pattern. All the tested algorithms start with the same input $\boldsymbol{Y}$, $\boldsymbol{D}_0$ and $\boldsymbol{X}_0$. For the regularized SimCO, μ is set to 0.01.

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

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

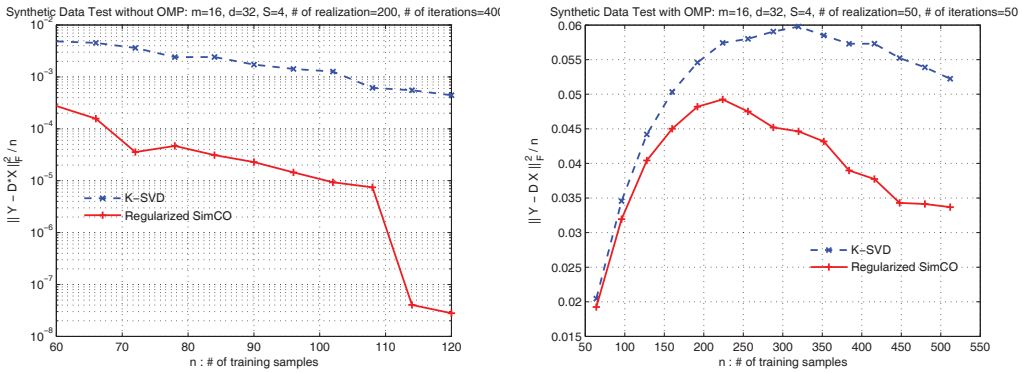
Here, note that $\kappa(\boldsymbol{D}_{\text{true}}) = 3.39$. Figure 1 shows that the regularized SimCO avoids the convergence to an ill-conditioned dictionary as compared with the other two algorithms. In addition, when the number of iterations exceeds 50, the gradients in both the K-SVD and the primitive SimCO surprisingly increase slightly with further iterations. This implies that these two methods do not converge to local minimizers.

4.2. Experiments on Synthetic Data

In the synthetic data tests, we assume that $\boldsymbol{Y} = \boldsymbol{D}_{\text{true}}\boldsymbol{X}_{\text{true}}$ where the columns of $\boldsymbol{D}_{\text{true}}$ are randomly generated from the Stiefel manifold $\mathcal{U}_{m,1}$, and each column of $\boldsymbol{X}_{\text{true}}$ contains exactly S many non-zeros that are Gaussian distributed. We fix $m = 16$, $d = 32$, and $S = 4$. We change the number of training samples n. For each value of n, we run 100 random tests. In each random test, we also randomly generate an initial dictionary $\boldsymbol{D}_0$ and an initial coefficient matrix $\boldsymbol{X}_0$.

We first test the performance of dictionary update without considering the effect of sparse coding. In particular, we assume the true sparsity is known by setting the sparsity pattern of $\boldsymbol{X}_0$ the same as that of $\boldsymbol{X}_{\text{true}}$. Noting the relation between the primitive and the regularized SimCO, the ideal way to test the regularized SimCO is to sequentially decrease μ to zero and let the regularized SimCO converge for each value of μ. In practice, we choose the following simple strategy: the total number of iterations is set to 400; we set μ to 0.1, 0.01, 0.001, and 0.0001, for iterations 1-100, 101-200, 201-300, and 301-400, respectively. For fair comparison, we

[1]Experiments on real data and theoretical analysis of the proposed algorithms can be found from http://arxiv.org/abs/1109.5302.



(a) Dictionary update only

(b) With sparse coding

Fig. 2: Performance comparison of the K-SVD and the regularized SimCO.

also set the number of iterations in K-SVD dictionary update to 400. The numerical results of $\|\boldsymbol{Y}-\boldsymbol{DX}\|_F^2/n$ versus n are presented in Figure 2. The average performance of the regularized SimCO is consistently better than that of K-SVD.

Then we evaluate the overall dictionary learning performance by combining the dictionary update and sparse coding stages. For sparse coding, we adopt the OMP algorithm [4] as it has been intensively used for testing the K-SVD method in [2, 7]. We refer to the iterations between sparse coding and dictionary learning stages as outer-iterations, and the iterations within the dictionary update stage as inner-iterations. In our test, the numbers of outer-iterations are set to 50 for both the K-SVD and the regularized SimCO, and in each outer iteration, the numbers of inner-iterations of both algorithms are set to 1. Furthermore, in the regularized SimCO, the regularized constant is set to $\mu = 0.1$ during the first 30 outer-iterations, and $\mu = 0$ during the rest 20 outer-iterations. The simulation results of $\|\boldsymbol{Y}-\boldsymbol{DX}\|_F^2/n$ versus n are depicted in Figure 2. Again, the average performance of the regularized SimCO is consistently better than that of the K-SVD.

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

5. CONCLUSION

We have presented a new codeword optimization algorithm and its extended version for dictionary learning, where an arbitrary set of codewords and their corresponding coefficients are allowed to be updated simultaneously. The numerical results, measured for the learning performance and the running speed, have shown that the proposed technique, in particular, the regularized SimCO, outperforms the K-SVD algorithm.

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

6. REFERENCES

[1] 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.

[2] 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.

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

[4] 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.

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

[6] 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.

[7] 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.