DICTIONARY LEARNING FOR SPARSE DECOMPOSITION: A NEW CRITERION AND ALGORITHM

Zahra Sadeghipoor, Massoud Babaie-Zadeh[*]

Electrical Engineering Department, Sharif University of Technology, Tehran, Iran
z.sadeghipoor@gmail.com
mbzadeh@yahoo.com

Christian Jutten

GIPSA-lab, Grenoble, France
and Institut Universitaire de France
christian.jutten@inpg.fr

ABSTRACT

During the last decade, there has been a growing interest toward the problem of sparse decomposition. A very important task in this field is dictionary learning, which is designing a suitable dictionary that can sparsely represent a group of training signals. In most dictionary learning algorithms, the cost function to determine the the optimum dictionary is the ℓ_0 norm of the matrix of decomposition coefficients of the training signals. However, we believe that this cost function fails to fully express the goal of dictionary learning, because it only sparsifies the *whole* set of coefficients for all training signals, rather than the coefficients for *each* training signal individually. Thus, in this paper we present a new criterion for dictionary learning. We then propose a new dictionary learning algorithm that solves our proposed optimization problem for the case of complete dictionaries. The proposed algorithm follows the idea of smoothed ℓ_0 (SL0) algorithm for sparse recovery. Simulation results emphasize the efficiency of the proposed cost function and algorithm.

Index Terms- Sparse Decomposition, Dictionary learning, Compressed Sensing.

1. INTRODUCTION

Consider the following signal decomposition:

$$\boldsymbol{x}^{n\times 1} = \boldsymbol{D}^{n\times m}.\boldsymbol{s}^{m\times 1} \qquad m \geq n \tag{1}$$

in which, the signal $\boldsymbol{x}$ is being expressed as a linear combination of the columns of $\boldsymbol{D}$, the vector $\boldsymbol{s}$ collects the coefficients of this decomposition and the superscripts denote dimensions. After [1], the matrix $\boldsymbol{D}$ is usually called the *dictionary* and its columns are called *atoms*. The dictionary can be complete ($m = n$) or overcomplete ($m > n$). For complete dictionaries the above decomposition is unique. For overcomplete dictionaries the decomposition is not unique, but one usually looks for the sparsest possible decomposition, that is, the decomposition that uses the minimum number of atoms to express the signal [2].

In the above problem, it has been assumed that the dictionary, $\boldsymbol{D}$, is known. However, having a suitable dictionary for a class of signals is crucial for having sparser representations, which is desired in many applications like image denoising and compression [2]. Such dictionaries are either chosen predetermined [*e.g.* complete or overcomplete Discrete Cosine Transform (DCT) for image signals], or they are learned for a given set of signals (the 'training' signals) coming from the desired class [3]. The 'dictionary learning' problem is important for both complete and over-complete dictionaries. For example, instead of using the complete DCT dictionary to denoise or compress a family of image signals (say facial pictures), one may be interested to use a dictionary learned from a set of given facial pictures. In this paper, we mainly address the problem of learning complete dictionaries. More precisely, although the discussion of Section 2 is general, the algorithm and simulations of Sections 3 and 4 are for the complete case only.

It is usually stated in the literature [3, 4] that the optimum dictionary for sparse decomposition can be designed by solving:

$$\{\boldsymbol{D}^*, \boldsymbol{S}^*\} = \underset{\boldsymbol{D},\boldsymbol{S}}{\operatorname{argmin}} \|\boldsymbol{S}\|_0 \text{ subject to } \boldsymbol{X}^{n\times N} = \boldsymbol{D}^{n\times m}.\boldsymbol{S}^{m\times N}, \tag{2}$$

where $\|.\|_0$ denotes the ℓ^0 quasi-norm of a vector or a matrix (*i.e.* the number of its non-zero entries), each column of $\boldsymbol{X}$ is a training signal, each column of $\boldsymbol{S}$ is the representation of the corresponding column of $\boldsymbol{X}$ based on the dictionary $\boldsymbol{D}$, and N is the number of training signals. The number of training signals is usually much more than the number of atoms, that is, $N \gg m$ (in fact, for $N = m$, the trivial solution is $\boldsymbol{D} = \boldsymbol{X}$, in which each signal is represented by exactly one non-zero coefficient; a similar thing is true when $N < m$).

In this paper, we discuss that *(2) is not well suited for dictionary learning, and needs to be modified*. The reason is that the goal of dictionary learning is to design a dictionary that can represent *each* training signal sparsely, whereas solving (2) only ensures that the *whole* matrix $\boldsymbol{S}$ is sparse and it does not imply that each column of $\boldsymbol{S}$ is *individually* sparse. Actually, previous dictionary learning algorithms (*e.g.* [4, 5, 6, 7]) have not encountered a problem with the above cost function, because they do not solve (2) directly, instead they use a two-step alternating minimization approach to learn the dictionary: In the first step, sparse representations of the training signals with a fixed dictionary are found, and in the second step the dictionary is updated to better fit the representations found in the previous phase. This process continues until convergence. This approach automatically avoids a representation matrix with all non-zero elements concentrated in a few columns, because in the first step the representation of each training signal is computed independently, and this adds a tendency to the algorithm to produce individually sparse representations. However, *such a tendency does not exist in the original cost function (2)*, and if we have a non-convex algorithm to directly solve it, then (2) is not a good criterion for dictionary learning and needs to be corrected. The algorithm that will be presented in Section 3 is such a direct minimization algorithm, and hence this correction to the criterion (2) will be crucial.

In this paper, we firstly modify the optimization problem (2) to

[*]This work has been partially funded by Iran NSF (INSF).

create the tendency to distribute non-zeros entries over all columns of $\boldsymbol{S}$. Secondly, for the complete case, we present an algorithm for solving the resulting optimization problem. This algorithm is inspired by the idea of Smoothed ℓ^0 (SL0) approach for sparse recovery [8].

The rest of the paper is organized as follows. In Section 2, we present the proposed criterion and discuss why this criterion is more appropriate than (2) for dictionary learning. Section 3 is devoted to describe the algorithm for solving the new optimization problem for the complete case. Finally, Section 4 studies the resulting algorithm numerically.

2. THE NEW CRITERION FOR DICTIONARY LEARNING

Firstly, let us show by using a toy example that solving (2) may result in a solution that is intuitively less desired. Assume that (2) is solved with respect to $\boldsymbol{S}$ for two fixed dictionaries $\boldsymbol{D}_1$ and $\boldsymbol{D}_2$, and the optimum solutions for these dictionaries are $\boldsymbol{S}_1^*$ and $\boldsymbol{S}_2^*$, respectively. Suppose that in the case of $m = 4$ and $N = 5$, $\boldsymbol{S}_1^*$ and $\boldsymbol{S}_2^*$ are as the following:

$$\boldsymbol{S}_1^* = \begin{pmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{pmatrix}$$

$$\boldsymbol{S}_2^* = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

The ℓ^0 norm of $\boldsymbol{S}_1^*$ is smaller than that of $\boldsymbol{S}_2^*$. Hence, between $(\boldsymbol{D}_1, \boldsymbol{S}_1^*)$ and $(\boldsymbol{D}_2, \boldsymbol{S}_2^*)$ pairs, the optimization process of problem (2) would choose the $(\boldsymbol{D}_1, \boldsymbol{S}_1^*)$ pair as the dictionary that is better for sparse representation of $\boldsymbol{X}$. However, $\boldsymbol{D}_1^*$ results in highly sparse representations for three signals, while two other signals (first and second columns of $\boldsymbol{S}_1^*$) are not sparse in the dictionary domain. So intuitively, $(\boldsymbol{D}_2, \boldsymbol{S}_2^*)$ is a better solution for the problem of dictionary learning because *all* training signals have fairly sparse representations using this pair.

To cope with this deficiency of (2), we propose using the following optimization problem to learn the optimum dictionary:

$$\{\boldsymbol{D}^*, \boldsymbol{S}^*\} = \underset{\boldsymbol{D},\boldsymbol{S}}{\operatorname{argmin}} \sum_{j=1}^{N} \|\boldsymbol{s}_j\|_0^2 \quad \text{subject to } \boldsymbol{X}^{n\times N} = \boldsymbol{D}^{n\times m}.\boldsymbol{S}^{m\times N} \tag{3}$$

where $\boldsymbol{s}_j$ is the j^{th} column of $\boldsymbol{S}$. Note that the previous cost function in (2) can be written as $\|\boldsymbol{S}\|_0 = \sum_{j=1}^{N} \|\boldsymbol{s}_j\|_0$. By adding the power two to the ℓ^0 norms of individual columns in our cost function, we have created a tendency to not produce too different $\|\boldsymbol{s}_j\|_0$'s. The value of our cost function in the above example for $\boldsymbol{S}_1^*$ is 21 and for $\boldsymbol{S}_2^*$ it is 20, so (3) does not choose $\boldsymbol{S}_1^*$ that contains columns with too different ℓ^0 norms.

To justify the efficiency of (3) in dictionary learning from a different point of view, consider the following lemma.

Lemma 1. *Consider a matrix $\boldsymbol{S}$ with a constant ℓ^0 norm. The cost function of (3) is minimum if and only if the ℓ^0 norms of all of its columns are equal.*

Proof. Cauchy inequality states that for numbers x_i and y_i, $i = 1, \ldots, N$, we have $(\sum_{j=1}^{N} x_j y_j)^2 \leqslant (\sum_{j=1}^{N} x_j^2)(\sum_{j=1}^{N} y_j^2)$, and the equality holds if and only if $\exists \alpha \forall i : x_i = \alpha y_i$. Hence for $x_i \triangleq \|\boldsymbol{s}_j\|_0$ and $y_i = 1$ we have:

$$\sum_{j=1}^{N} \|\boldsymbol{s}_j\|_0^2 \geqslant \frac{1}{N}(\sum_{j=1}^{N} \|\boldsymbol{s}_j\|_0)^2 = \frac{\|\boldsymbol{S}\|_0^2}{N} \tag{4}$$

where $\boldsymbol{s}_j$ is the j^{th} column of $\boldsymbol{S}$, and the equality holds if and only if all $\|\boldsymbol{s}_j\|_0$'s are equal. Consequently, for a constant $\|\boldsymbol{S}\|_0$, $\sum_{j=1}^{N} \|\boldsymbol{s}_j\|_0^2$ is minimum if and only if the ℓ^0 norms of all columns are equal. □

It was also possible to propose other optimization problems that ensure homogeneous sparsity over the columns of $\boldsymbol{S}$, by applying other functions on the ℓ^0 norms of the columns of $\boldsymbol{S}$. For instance, instead of the power two, we could apply power 4. In fact, the key point in defining an adequate optimization problem for dictionary learning is to impose a strictly convex and increasing function on $[0, +\infty]$ on the ℓ^0 norms of the columns of $\boldsymbol{S}$, to create a tendency of producing a kind of homogeneous sparsity over the representations of different training signals. Of course this tendency should not be too much, which is probably the case for power 4 or larger. However, among all possible optimization problems, we prefer to use (3), because the resulting nonlinear function is quadratic and easier to optimize. In this paper, we continue the discussion with the cost function proposed in (3), and studying the choice of an optimal convex nonlinearity that could be applied on the ℓ^0 norms of the columns is not the subject of this paper.

3. AN ALGORITHM FOR SOLVING THE PROPOSED OPTIMIZATION PROBLEM

In this paper, we propose an algorithm to solve (3) in the case of complete (square) dictionaries[1]. This case is simpler, because $\boldsymbol{D}$ uniquely determines $\boldsymbol{S}$ through $\boldsymbol{S} = \boldsymbol{D}^{-1}\boldsymbol{X}$, and hence we have no optimization to do with respect to $\boldsymbol{S}$. Thus, (3) is simplified as:

$$\boldsymbol{D}^* = \underset{\boldsymbol{D}}{\operatorname{argmin}} \sum_{j=1}^{N} \|(\boldsymbol{D}^{-1}\boldsymbol{X})_j\|_0^2 \tag{5}$$

where $(\boldsymbol{D}^{-1}\boldsymbol{X})_j$ is the j^{th} column of $\boldsymbol{D}^{-1}\boldsymbol{X}$.

The ℓ^0 norm of a vector is discontinuous, so the problem (5) is probably NP-hard. Thus, to minimize the cost function of (5), we use the idea of smoothed ℓ^0 norm (SL0) [8].

In [8], Mohimani *et al.* proposed to approximate the ℓ_0 norm of a vector with a smooth function, and then minimize the smooth function to find the sparsest solution of an underdetermined system of linear equations. They proposed to use the following equality to approximate the ℓ_0 norm:

$$\|\boldsymbol{s}\|_0 = \lim_{\sigma \to 0} \sum_{i=1}^{n} (1 - f_\sigma(s_i)) \tag{6}$$

[1] One may argue that in this case the optimal sparsifying dictionary is the Karhunen-Loève Transform (KLT) given by Principal Component Analysis (PCA). Note however that firstly, contrary to PCA, (3) is not limited to produce unitary dictionaries. Secondly, PCA does not maximize the 'sparsity' of the representation, it maximizes the 'energy compaction' of the representation. In this point of view, even by forgetting the unitary constraint, the use of KLT as the sparsifying transform is somehow like using the idea of Matching Pursuit (MP) [1] instead of minimizing the ℓ^0 norm in sparse recovery.

where $f_\sigma(s_i) \triangleq \exp(-\frac{s_i^2}{\sigma^2})$ in which s_i is the i^{th} element of $\boldsymbol{s}$ and n is its length. Thus, the ℓ^0 norm of a vector can be approximated by the following continuous function:

$$\|\boldsymbol{s}\|_0 \cong n - \sum_{i=1}^{n} \exp(-\frac{s_i^2}{\sigma^2}) \tag{7}$$

In $f_\sigma(s_i)$, σ determines the accuracy of approximation. The larger σ, the smoother $\sum_{i=1}^{n}(1 - f_\sigma(s_i))$ but worse approximation of the ℓ^0 norm; and the smaller σ, the better approximation of the ℓ^0 norm but the less smooth $\sum_{i=1}^{n}(1 - f_\sigma(s_i))$. The approximation tends to equality for $\sigma \to 0$. Moreover, for $\sigma \to \infty$, $\sum_{i=1}^{n}(1 - f_\sigma(s_i))$ tends to a very smooth and convex function (contrary to the ℓ^0 norm itself) [8]. The goal is then to minimize $\sum_{i=1}^{n}(1 - f_\sigma(s_i))$ for a very small value of σ. However, this is difficult, because for small values of σ, $\sum_{i=1}^{n}(1 - f_\sigma(s_i))$ is highly non-smooth with lots of local minima. To escape from these local minima the "Graduated Non-Convexity (GNC)" procedure can be used [9]: The algorithm starts with a very large σ (for which there is no local minima), and decreases σ gradually to zero. *The minimizer of $\sum_{i=1}^{n}(1 - f_\sigma(s_i))$ is used as the starting point to locate the minimum of this function for the next (smaller) σ, using a steepest descent approach*. Since the value of σ is just slightly decreased, it is likely that the minimizer of $\sum_{i=1}^{n}(1 - f_\sigma(s_i))$ for the new σ is not too far from the minimizer of the function for the previous (larger) σ, and hence it is likely that the algorithm escapes from getting trapped into a local minimum (for more details about this algorithm refer to [8]).

To solve (5), we use a similar smooth approximation for the ℓ^0 norm and a GNC approach. By defining $\boldsymbol{B} = \boldsymbol{D}^{-1}$ and using the approximation in (7), (5) can be rewritten as follows:

$$\boldsymbol{B}^* = \underset{\boldsymbol{B}}{\operatorname{argmin}} \sum_{j=1}^{N} \left[n - \sum_{i=1}^{n} \exp(-\frac{(\boldsymbol{b}_i^T.\boldsymbol{x}_j)^2}{\sigma^2}) \right]^2, \tag{8}$$

where $\boldsymbol{b}_i^T$ denotes the i^{th} row of $\boldsymbol{B}$ and $\boldsymbol{x}_j$ is the j^{th} training signal. Using the steepest descent approach, the update rule of the algorithm is:

$$\boldsymbol{B}^{k+1} = \boldsymbol{B}^k - \mu_\sigma \nabla_{\boldsymbol{B}} F(\boldsymbol{X}, \boldsymbol{B}^k), \tag{9}$$

where $\boldsymbol{B}^k$ stands for the approximation of inverse dictionary at the k^{th} iteration of the algorithm, and $F(\boldsymbol{X}, \boldsymbol{B})$ is the cost function of (8), that is:

$$F(\boldsymbol{X}, \boldsymbol{B}) \triangleq \sum_{j=1}^{N} \left[n - \sum_{i=1}^{n} \exp(-\frac{(\boldsymbol{b}_i^T.\boldsymbol{x}_j)^2}{\sigma^2}) \right]^2. \tag{10}$$

In (9), $\nabla_{\boldsymbol{B}} F(\boldsymbol{X}, \boldsymbol{B})$ is a matrix whose $(i,j)^{\text{th}}$ element is the derivative of $F(\boldsymbol{X}, \boldsymbol{B})$ with respect to the $(i,j)^{\text{th}}$ element of $\boldsymbol{B}$. By simple mathematical calculations, we can show that the i^{th} row of $\nabla_{\boldsymbol{B}} F(\boldsymbol{X}, \boldsymbol{B})$ is equal to:

$$\sum_{j=1}^{N} \left[n - \sum_{l=1}^{n} \exp(-\frac{s_{lj}^2}{\sigma^2}) \right] \frac{2s_{ij}}{\sigma^2} \exp(-\frac{s_{ij}^2}{\sigma^2}) \boldsymbol{x}_j^T, \tag{11}$$

where s_{ij} is the $(i,j)^{\text{th}}$ element of the matrix $\boldsymbol{S}$.

Note that without any constraints on $\boldsymbol{B}$, the minimizer of (5) tends to decrease the values of matrix $\boldsymbol{B}$, to decrease the values of s_{ij} and hence to decrease the cost function of (3). In order to avoid this situation, we normalize the rows of $\boldsymbol{B}$ in each iteration of the algorithm.

Finally, as stated in [8], μ_σ should be decreased proportional to σ^2, and hence we put $\mu_\sigma = \mu\sigma^2$. The pseudo-code of the final algorithm is shown in Fig. 1.

- *User sets the following parameters*:
  1. c is the factor by which σ is decreased.
  2. σ_{min} is the minimum σ.
  3. L is the number of iterations of the inner loop as in SL0 [8].
  4. μ is the step size in the updating equation.
- *Initialization*:
  1. Generate a random matrix with normalized rows as $\boldsymbol{B}_0$.
  2. Set $\boldsymbol{S} = \boldsymbol{B}_0.\boldsymbol{X}$.
  3. Set $\sigma = 2\max(\text{abs}(\boldsymbol{S}))$ as suggested in [8].
- *Repeat the following steps until $\sigma > \sigma_{min}$.*
  1. Repeat the following steps L times.
     (a) Calculate $\nabla_{\boldsymbol{B}} F(\boldsymbol{X}, \boldsymbol{B})$ using (11).
     (b) Update $\boldsymbol{B}$ as the following:
     $$\boldsymbol{B} \leftarrow \boldsymbol{B} - (\mu\sigma^2)\nabla_{\boldsymbol{B}} F(\boldsymbol{X}, \boldsymbol{B})$$
     (c) Normalize each row of $\boldsymbol{B}$.
  2. Change σ to $\sigma = c \times \sigma$ and go to step 1.

Fig. 1. The proposed dictionary learning algorithm.

4. SIMULATION RESULTS

In this section, we first study the effectiveness of the proposed optimization problem (3) for dictionary learning and compare it with (2). To this end, we implemented the SL0-based dictionary learning algorithm proposed in Section 3 for both optimization problems (3) and (2). Note that the only change required in Fig. 1 to solve (2) is the gradient. We refer to the algorithm that solves (3) as $\ell_0\ell_2$ and call the other one $\ell_0\ell_1$.

All experiments were done on synthetic data. A dictionary of size 20×20 is generated with zero mean Gaussian random elements with unit variance. We generate matrix $\boldsymbol{S}$ such that each of its columns has exactly k non-zero elements (k is changed in different experiments). Locations of non-zero elements are uniformly random and their values are zero mean Gaussian random variables with unit variance. The matrix of training signals is generated as $\boldsymbol{X} = \boldsymbol{D}.\boldsymbol{S}$. The number of training signals is 1000.

Parameters used for both algorithms are $\sigma_{min} = 10^{-2}, c = 0.8, L = 2000$, and $\mu = 0.01$. In each experiment, we used the same initial point for both $\ell_0\ell_1$ and $\ell_0\ell_2$ algorithms. The estimated dictionary was compared against the original dictionary. To determine how many columns of the dictionary are estimated correctly, we use the success rate proposed in [4]. The success rate is the number of correctly estimated columns divided by the number of all dictionary columns.

Each reported result is an average over 20 trials (with different dictionaries and different training signals).

Figure 2 summarizes the success rate of both algorithms for different numbers of non-zero coefficients (k). It can be immediately seen in this figure that by directly minimizing (2) the initial dictionary almost never converges to the original one, while solving our

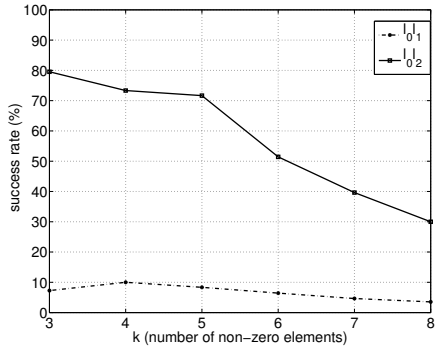

Fig. 2. The success rate of the algorithm that uses our optimization problem ($\ell_0\ell_2$) and the one that minimizes the conventional dictionary learning cost function ($\ell_0\ell_1$).

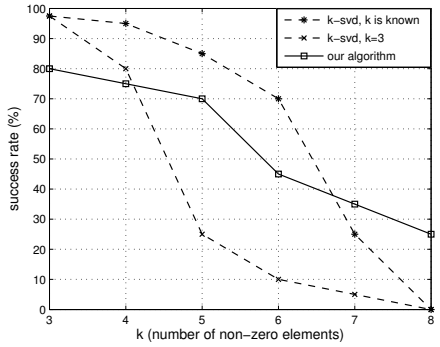

Fig. 3. The success rate of our algorithm and the K-SVD dictionary learning method in estimating a synthetic dictionary.

proposed optimization problem performs fairly satisfactory.

Next we compare the performance of our algorithm in solving the proposed optimization problem with the well-known K-SVD algorithm [4] for dictionary learning. To simulate the K-SVD algorithm, we have used the MATLAB toolbox available at `http://www.cs.technion.ac.il/~elad/Various/KSVD_Matlab_ToolBox.zip`. The experimental setup and the parameters of our algorithm are the same as the first experiment.

Figure 3 shows the results of our algorithm and K-SVD for different values of k. This figure shows the success rate of both algorithms. Figure 3 shows the performance of the K-SVD algorithm in two cases. First, when k (the number of non-zero elements of each column of $\boldsymbol{S}$) is known as a prior information and used as an input to the algorithm, second when only an underestimated value of k ($k = 3$) is available as the input. Note that our algorithm does not need to know the value of k and it is not one of the input parameters of the algorithm.

As can be seen in Fig. 3, the K-SVD algorithm when the value of k is assumed to be known in advance outperforms our algorithm. However, if only an estimation of k is accessible, our algorithm can perform the same or even better than K-SVD.

We observed that if a good estimation of the dictionary is available as the initial point, our algorithm would converge to the optimal dictionary. Figure 4 shows the estimation results of our algorithm

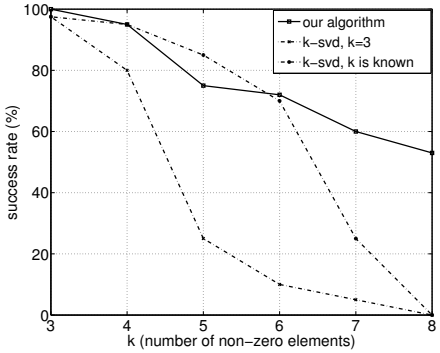

Fig. 4. The success rate of our method and the K-SVD algorithm. The initial point for our algorithm is $\boldsymbol{B}_0 = \boldsymbol{B} + 0.1\boldsymbol{\Lambda}$ and $k = 3$ is used as the input parameter of K-SVD.

when the initial inverse matrix ($\boldsymbol{B}_0$) is close to the original one, i.e.,:

$$\boldsymbol{B}_0 = \boldsymbol{B} + 0.1\boldsymbol{\Lambda}, \tag{12}$$

where $\boldsymbol{B}$ is the inverse matrix of the original dictionary, $\boldsymbol{\Lambda}$ is a matrix with zero mean Gaussian entries with unit variance. Note that the rows of $\boldsymbol{B}$ in (12) have unit ℓ^2 norm. Figure 4 also shows the results of the K-SVD algorithm in two cases (when k is known, and when only an underestimation of k is available to the algorithm). As can be seen in this figure, our algorithm when the initial point is close to the original dictionary outperforms K-SVD when a good estimation of the number of non-zero elements in the representation matrix (k) does not exist. It can also be seen that even if the K-SVD algorithm knows the exact value of k, in most cases, our method (starting from a good initial point) performs the same or even better than the state-of-the-art.

In summary, in comparison to K-SVD, the problem of local minima is more severe in our algorithm, but it results in a better estimation near the true answer. Consequently, one may start with K-SVD and then refines the quality of estimation by our algorithm. Moreover, it has to be noted that K-SVD has been developed to design complete or over-complete dictionaries, whereas our algorithm is only for designing complete dictionaries.

5. CONCLUSION

In this paper, we presented a new cost function for dictionary learning. We showed that the proposed cost function is more suitable than the traditional one, because contrary to the traditional one, it has the tendency of producing individual sparse representations for all training data. Simulation results by using an algorithm for minimizing our proposed cost function showed that having such a tendency is crucial, and without it the algorithm fails to retrieve the correct dictionary. Actually, it seems that previous dictionary learning algorithms have not encountered such a problem, because of the alternating-minimization approach that they use to minimize the traditional cost function, which creates the mentioned tendency as a side-effect.

6. REFERENCES

[1] S. Mallat and Z. Zhang, "Matching pursuits with time frequency dictionaries," *IEEE Transactions on Signal Processing*, vol. 41,

no. 12, pp. 3397–3415, 1993.

[2] M. Elad, *Sparse and Redundant Representations: From Theory to Applications in Signal and Image Processing*, Springer, 2010.

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

[4] M. Aharon, M. Elad, and A.M. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, November 2006.

[5] K. Engan, S. O. Aase, and J. H. Hakon-Husoy, "Method of optimal direction for frame design," *IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 5, pp. 2443–2446, 1999.

[6] K. Kreutz-Delgado, J. F. Murray, B. D. Rao, K. Engan, T. Lee, and T. J. Sejnowsk, "Dictionary learning algorithms for sparse representation," *Neural Computation*, vol. 15, no. 2, pp. 349–396, 2003.

[7] M. S. Lewicki and T. J. Sejnowski, "Learning overcomplete representations," *Neural Computation*, vol. 12, pp. 337–365, 2000.

[8] H. Mohimani, M. Babaie-Zadeh, and C. Jutten, "A fast approach for overcomplete sparse decomposition based on smoothed l^0-norm," *IEEE Transactions on Signal Processing*, vol. 57, no. 1, pp. 289–301, January 2009.

[9] A. Blake and A. Zisserman, *Visual Reconstruction*, MIT Press, 1987.