Accelerated Dictionary Learning for Sparse Signal Representation

Fateme Ghayem[1], Mostafa Sadeghi[1],
Massoud Babaie-Zadeh[1], and Christian Jutten[2] *

[1]Department of Electrical Engineering, Sharif University of Technology, Tehran, Iran
[2]GIPSA-Lab, Institut Universitaire de France, Grenoble, France
{fateme.ghayem,m.saadeghii}@gmail.com,
mbzadeh@yahoo.com,
christian.jutten@gipsa-lab.grenoble-inp.fr

Abstract. Learning sparsifying dictionaries from a set of training signals has been shown to have much better performance than pre-designed dictionaries in many signal processing tasks, including image enhancement. To this aim, numerous practical dictionary learning (DL) algorithms have been proposed over the last decade. This paper introduces an accelerated DL algorithm based on iterative proximal methods. The new algorithm efficiently utilizes the iterative nature of DL process, and uses accelerated schemes for updating dictionary and coefficient matrix. Our numerical experiments on dictionary recovery show that, compared with some well-known DL algorithms, our proposed one has a better convergence rate. It is also able to successfully recover underlying dictionaries for different sparsity and noise levels.

Keywords: Sparse representation, compressed sensing, dictionary learning, proximal algorithms.

1 Introduction

The information contents of natural signals are usually significantly less than their ambient dimensions. This fact has been extensively used in many applications, including compressed sensing [1]. Let $\mathbf{y} \in \mathbb{R}^m$ be a given (natural) signal. Then, its representation over a set of signal building blocks, $\{\mathbf{d}_1, \cdots, \mathbf{d}_n\}$ (called atoms) is written as $\mathbf{y} = \sum_{i=1}^{n} x_i \cdot \mathbf{d}_i = \mathbf{D}\mathbf{x}$, where $\mathbf{D} \in \mathbb{R}^{m \times n}$ is called dictionary, which contains $\mathbf{d}_i$'s as its columns, and $\mathbf{x} \in \mathbb{R}^n$ is the vector of coefficients. If the dictionary is chosen appropriately, then the coefficients vector $\mathbf{x}$ is expected to be very sparse. So, an important question is how to choose the sparsifying dictionary $\mathbf{D}$. Discrete cosine transform (DCT) and wavelets are two well-known predesigned dictionaries which can be used in sparsity-based applications. However, such fixed transforms (dictionaries) are suitable for only particular class

* This work has been funded by ERC project 2012-ERC-AdG- 320684 CHESS and by the Center for International Scientific Studies and Collaboration (CISSC).

of signals. An alternative and more efficient way to choose the dictionary is to learn it from a set of training signals. This process is called *dictionary learning* (DL), which has received a lot of attention over the last decade [2]. Given a set of training signals $\{\mathbf{y}_i\}_{i=1}^{L}$ collected as the columns of the matrix $\mathbf{Y} \in \mathbb{R}^{m\times L}$, a dictionary $\mathbf{D}$ and the corresponding coefficient matrix $\mathbf{X}$ are optimized such that the representation error $\|\mathbf{Y}-\mathbf{D}\mathbf{X}\|_F^2$ becomes small and $\mathbf{X}$ has sufficiently sparse columns. This problem can be formulated as

$$\min_{\mathbf{D}\in\mathcal{D},\,\mathbf{X}} \left\{ \frac{1}{2}\|\mathbf{Y}-\mathbf{D}\mathbf{X}\|_F^2 + \lambda\|\mathbf{X}\|_0 \right\}, \tag{1}$$

where $\lambda > 0$ is a regularization parameter, $\|\cdot\|_0$ denotes the so-called ℓ_0 pseudo-norm which counts the number of non-zero entries, and $\mathcal{D}$ is defined as follows:

$$\mathcal{D} \triangleq \left\{ \mathbf{D} \in \mathbb{R}^{m\times n} \;\middle|\; \forall\, i:\ \|d_i\|_2 = 1 \right\}. \tag{2}$$

Many algorithms have been proposed to solve (1) or its variants [3–9]. Most of these algorithms follow an alternating minimization approach, consisting of two main steps: sparse representation (SR) and dictionary update (DU). In the first step, $\mathbf{D}$ is kept fixed and the minimization is done over $\mathbf{X}$. There exist many efficient algorithms to perform this step, e.g., orthogonal matching pursuit (OMP) [10]. In the DU step, $\mathbf{X}$ is set to its current estimate obtained in the SR step, and $\mathbf{D}$ is updated. We refer to one round of performing these two steps as one *DL iteration*.

When SR and DU steps are solved using iterative algorithms, the iterations of DL can be efficiently utilized to reduce the work needed for updating $\mathbf{D}$ and $\mathbf{X}$. In other words, the final estimates of $\mathbf{D}$ and $\mathbf{X}$ obtained in each DL iteration can be used to initialize the SR and DU steps of the next DL iteration. Besides accelerating the whole DL procedure, this so-called *warm-starting* may also avoid undesired local minima. Earlier works, including method of optimal directions (MOD) [3] and K-singular value decomposition (K-SVD) [4], do not take full advantage of this fact. In fact, both of them use OMP to perform the SR step, which does not efficiently use current estimate of the coefficient matrix[1]. Moreover, MOD finds the unconstrained least-squares solution of the DU step

$$\mathbf{D} = \mathbf{Y}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1} \tag{3}$$

which is followed by a column normalization. So, the previous estimate of $\mathbf{D}$ is not used to find the next one. K-SVD, on the other hand, updates the dictionary atom-by-atom together with the non-zero entries of the coefficient matrix. In this way, the previous estimates of the atoms are used, in someway, to get the new estimates. Instances of the DL algorithms that utilize iterative algorithms to update $\mathbf{D}$ and $\mathbf{X}$ include [8], which uses iterative majorization minimization,

[1] It should be mentioned that, a modification to OMP has been proposed in [11], which reuses the coefficients obtained in each DL iteration in order to initialize OMP for the next DL iteration.

and [9], which proposes a multi-block hybrid proximal alternating (MBHPA) algorithm to solve the DL problem in (1).

In this paper, new iterative schemes based on proximal gradient algorithms [12] are proposed to perform SR and DU steps. Unlike the algorithm proposed in [9], which is also based on proximal approach, our algorithm is equipped with accelerated extrapolation and inertial techniques [13, 14]. Moreover, ℓ_1 norm is used here, as the sparsity measure in contrast to [9] that uses ℓ_0 pseudo-norm. Another difference between our algorithm and the one proposed in [9] is that, we update the whole dictionary using iterative proximal gradient method, while [9] updates the atoms of the dictionary sequentially. As will be shown in Section 3, our proposed algorithm, which we call accelerated dictionary learning (ADL), outperforms K-SVD and the algorithm introduced in [9].

The rest of the paper is organized as follows. In Section 2, our new iterative algorithms for performing SR and DU steps are introduced. Then, Section 3 presents the simulation results.

2 Proposed Method

2.1 Main problem

We target the following problem in order to learn overcomplete dictionaries:

$$\min_{\mathbf{D}\in\mathcal{D},\mathbf{X}} \left\{\|\mathbf{Y}-\mathbf{D}\mathbf{X}\|_F^2+\lambda\|\mathbf{X}\|_1\right\}. \tag{4}$$

In the same way as usual methods, solving (4) consists of SR and DU steps, which are performed alternatively. In the rest of this section, we will separately illustrate how each step is realized to update the coefficient matrix, $\mathbf{X}$, and the dictionary, $\mathbf{D}$. Before proceeding, let us review some notations and terminologies related to proximal algorithms [12].

Definition 1 ([12]). *The projection of a point $\mathbf{x}\in\mathbb{R}^n$ onto a non-empty set $\mathcal{S}\subseteq\mathbb{R}^n$ is defined as*

$$P_{\mathcal{S}}\left\{\mathbf{x}\right\}\triangleq\underset{\mathbf{u}\in\mathcal{S}}{\operatorname{argmin}}\ \frac{1}{2}\|\mathbf{x}-\mathbf{u}\|_2^2.$$

Definition 2 ([12]). *The proximal mapping of a convex function $g:\ \mathrm{dom}_g\longrightarrow\mathbb{R}$ is defined as*

$$\mathrm{Prox}_g\left\{\mathbf{x}\right\}\triangleq\underset{\mathbf{u}\in\mathrm{dom}_g}{\operatorname{argmin}}\ \left\{\frac{1}{2}\|\mathbf{u}-\mathbf{x}\|_2^2+g(\mathbf{u})\right\}.$$

Let $\delta_{\mathcal{S}}(\mathbf{x})$ denote the indicator function of the set $\mathcal{S}$, i.e.,

$$\delta_{\mathcal{S}}(\mathbf{x})\triangleq\begin{cases}0 & \mathbf{x}\in\mathcal{S}\\ \infty & \mathbf{x}\notin\mathcal{S}\end{cases}. \tag{5}$$

The proximal mapping of $\delta_{\mathcal{S}}$ is, then, the projection onto $\mathcal{S}$ [12]

$$\mathrm{Prox}_{\delta_{\mathcal{S}}}\left\{\mathbf{x}\right\}=P_{\mathcal{S}}\left\{\mathbf{x}\right\}. \tag{6}$$

2.2 Sparse representation

To perform the SR step, let us define

$$f(\mathbf{D},\mathbf{X}) \triangleq \frac{1}{2}\|\mathbf{Y}-\mathbf{D}\mathbf{X}\|_F^2. \tag{7}$$

The SR step for the i-th DL iteration is then

$$\mathbf{X}^{(i)} = \underset{\mathbf{X}}{\operatorname{argmin}} \ \left\{ f(\mathbf{D}^{(i-1)},\mathbf{X}) + \lambda\|\mathbf{X}\|_1 \right\}. \tag{8}$$

To solve the above problem using iterative proximal gradient algorithms [12], $f(\mathbf{D}^{(i-1)},\mathbf{X})$ is replaced with its quadratic approximation around the previous estimate of $\mathbf{X}^{(i)}$ [12]. It is straightforward to show that the final problem is

$$\begin{aligned}\mathbf{X}_{k+1}^{(i)} &= \underset{\mathbf{X}}{\operatorname{argmin}} \left\{ \frac{1}{2\mu_x}\|\mathbf{X}-\widehat{\mathbf{X}}_k^{(i)}\|_F^2 + \lambda\|\mathbf{X}\|_1 \right\} \\ &= \operatorname{Prox}_{\mu_x\lambda\|.\|_1}\left\{\widehat{\mathbf{X}}_k^{(i)}\right\},\end{aligned} \tag{9}$$

where, $\widehat{\mathbf{X}}_k^{(i)} \triangleq \mathbf{X}_k^{(i)} - \mu_x\nabla_X f(\mathbf{D}^{(i-1)},\mathbf{X}_k^{(i)})$, k stands for the iteration index, and μ_x is a step-size which is set as $\mu_x = 1/\|(\mathbf{D}^{(i-1)})^T\mathbf{D}^{(i-1)}\|$, with $\|.\|$ being the matrix spectral norm [15]. The proximal mapping of the ℓ_1 norm is the so-called soft-thresholding operation [12]. The component-wise soft-thresholding function, denoted by $\operatorname{Soft}_\lambda$, is defined as [16]

$$\operatorname{Soft}_\lambda\{x\} \triangleq \operatorname{sgn}(x)\cdot\max(|x|-\lambda,0). \tag{10}$$

The iterative proximal gradient algorithm to solve (8) can be compactly written as

$$\mathbf{X}_{k+1}^{(i)} = \operatorname{Soft}_{\mu_x\lambda}\left\{\mathbf{X}_k^{(i)} - \mu_x\nabla_X f(\mathbf{D}^{(i-1)},\mathbf{X}_k^{(i)})\right\}. \tag{11}$$

In order to accelerate the algorithm, we add an extrapolation step [14] to the above algorithm as follows,

$$\begin{cases}\mathbf{Z}_k^{(i)} = \mathbf{X}_k^{(i)} + w_x(\mathbf{X}_k^{(i)} - \mathbf{X}_{k-1}^{(i)}) \\ \mathbf{X}_{k+1}^{(i)} = \operatorname{Soft}_{\mu_x\lambda}\left\{\mathbf{Z}_k^{(i)} - \mu_x\nabla_X f(\mathbf{D}^{(i-1)},\mathbf{Z}_k^{(i)})\right\}\end{cases}, \tag{12}$$

in which, $w_x \geq 0$ is a weighting parameter which controls the convergence rate of the algorithm. The above iterations are repeated until $\|\mathbf{X}_{k+1}^{(i)} - \mathbf{X}_k^{(i)}\|_F \leq \tau_x$, where τ_x is a given tolerance. This accelerated iterative scheme has already been discussed in some previous works, including [13, 17] to solve the vector form of the ℓ_1-based sparse representation problem.

2.3 Dictionary update

The problem to be solved in the DU step is as fallows:

$$\mathbf{D}^{(i+1)} = \underset{\mathbf{D}\in\mathcal{D}}{\operatorname{argmin}} \ f(\mathbf{D},\mathbf{X}^{(i)}), \tag{13}$$

which can be equivalently written as

$$\mathbf{D}^{(i+1)} = \underset{\mathbf{D}}{\operatorname{argmin}} \ \left\{ f(\mathbf{D}, \mathbf{X}^{(i)}) + \delta_{\mathcal{D}}(\mathbf{D}) \right\}. \tag{14}$$

Following the same approach used to solve (8), the iterative proximal gradient algorithm to solve (14) becomes as

$$\begin{aligned} \mathbf{D}_{k+1}^{(i+1)} &= \underset{\mathbf{D}}{\operatorname{argmin}} \left\{ \frac{1}{2\mu_d} \|\mathbf{D} - \widehat{\mathbf{D}}_k^{(i+1)}\|_F^2 + \delta_{\mathcal{D}}(\mathbf{D}) \right\} \\ &= \operatorname{Prox}_{\mu_d \delta_{\mathcal{D}}} \left\{ \widehat{\mathbf{D}}_k^{(i+1)} \right\}, \end{aligned} \tag{15}$$

where, $\widehat{\mathbf{D}}_k^{(i+1)} \triangleq \mathbf{D}_k^{(i+1)} - \mu_d \nabla_D f(\mathbf{D}_k^{(i+1)}, \mathbf{X}^{(i)})$, and μ_d is a step-size, which is set as $\mu_d = 1/\|(\mathbf{X}^{(i)})^T \mathbf{X}^{(i)}\|$. According to (6), the proximal mapping in (15) is the projection onto $\mathcal{D}$. So, the iterative proximal gradient algorithm to solve (14) can be compactly written as

$$\mathbf{D}_{k+1}^{(i+1)} = \mathcal{P}_{\mathcal{D}} \left\{ \mathbf{D}_k^{(i+1)} - \mu_d \nabla_D f(\mathbf{D}_k^{(i+1)}, \mathbf{X}^{(i)}) \right\}. \tag{16}$$

To accelerate the above algorithm, we add an inertial term [14] to the above algorithm as follows:

$$\begin{cases} \mathbf{C}_k^{(i+1)} = \mathbf{D}_k^{(i+1)} - \mu_d \nabla_D f(\mathbf{D}_k^{(i+1)}, \mathbf{X}^{(i)}) \\ \mathbf{D}_{k+1}^{(i+1)} = \mathcal{P}_{\mathcal{D}} \left\{ \mathbf{C}_k^{(i+1)} \right\} + w_d (\mathbf{D}_k^{(i+1)} - \mathbf{D}_{k-1}^{(i+1)}) \end{cases}, \tag{17}$$

in which, $w_d \geq 0$ is a weighting parameter, which controls the convergence rate of the algorithm. Similar to the $\mathbf{X}$ update step, the above iterations are repeated until $\|\mathbf{D}_{k+1}^{(i+1)} - \mathbf{D}_k^{(i+1)}\|_F \leq \tau_d$, where τ_d is a given tolerance. Here, one may wonder why an inertial scheme is not used for the SR step, or similarly, why an extrapolated scheme, like the one used in (12), is not used instead of (17). In fact, it was observed in our simulations that for the SR step, the extrapolation, and for the DU step, the inertial technique result in the fastest convergence.

2.4 Non-zero coefficients update

Similar to K-SVD, in order to further accelerate the whole DL algorithm, the non-zero entries of $\mathbf{X}$ are also updated after the DU step. Let $\mathbf{x}_{[l]}$ denote the l-th row of $\mathbf{X}$, and $\Omega_l \triangleq \left\{ j \mid \mathbf{x}_{[l]}(j) \neq 0 \right\}$ be the indexes of non-zeros in $\mathbf{x}_{[l]}$. Then, $\mathbf{x}_{[l]}(\Omega_l)$ is updated as follows

$$\mathbf{x}_{[l]}(\Omega_l) = \underset{\mathbf{x}_{[l]}^r}{\operatorname{argmin}} \frac{1}{2} \|\mathbf{E}(\Omega_l) - \mathbf{d}_l \mathbf{x}_{[l]}^r\|_F^2 = \mathbf{d}_l^T \mathbf{E}(\Omega_l) \tag{18}$$

in which, $\mathbf{E}(\Omega_l)$ contains those columns of $\mathbf{E} = \mathbf{Y} - \sum_{i \neq l} \mathbf{d}_i \mathbf{x}_{[i]}$ indexed in Ω_l, and $\mathbf{x}_{[l]}^r$ is a row vector of length $|\Omega_l|$. This process is repeated for all the rows of $\mathbf{X}$.

A detailed description of the proposed DL algorithm, which we call accelerated dictionary learning (ADL), is given in Algorithm 1.

Algorithm 1 ADL

Require: $\mathbf{Y}$, $\mathbf{D}^{(1)}=\mathbf{D}_0$, $\mathbf{X}^{(1)}=\mathbf{X}_0$, λ, I, τ_X, τ_D, w_x, w_d.
for $i=1,2,\cdots,I$ **do**
  1. Sparse representation:
  $\mathbf{D}=\mathbf{D}^{(i)}$, $k=1$.
  $\mu_x=1/\|\mathbf{D}^T\mathbf{D}\|$.
  while $\|\mathbf{X}_{k+1}^{(i)}-\mathbf{X}_k^{(i)}\|_F>\tau_X$ **do**
    $\widehat{\mathbf{X}}_k^{(i)}=\mathbf{X}_k^{(i)}+w_x(\mathbf{X}_k^{(i)}-\mathbf{X}_{k-1}^{(i)})$
    $\mathbf{X}_{k+1}^{(i)}=\mathrm{Soft}_{\lambda\mu_X}\left\{\widehat{\mathbf{X}}_k^{(i)}-\mu_x\nabla_{\mathbf{X}}f(\mathbf{D},\widehat{\mathbf{X}}_k^{(i)})\right\}$
    $k=k+1$
  end while
  $\mathbf{X}^{(i)}=\mathbf{X}_{k+1}^{(i)}$.
  2. Dictionary update:
  $\mathbf{X}=\mathbf{X}^{(i)}$, $k=1$.
  $\mu_d=1/\|\mathbf{X}^T\mathbf{X}\|$.
  while $\|\mathbf{D}_{k+1}^{(i)}-\mathbf{D}_k^{(i)}\|_F>\tau_d$ **do**
    $\widehat{\mathbf{D}}_k^{(i)}=\mathbf{D}_k^{(i)}-\mu_d\nabla_{\mathbf{D}}f(\mathbf{D}_k^{(i)},\mathbf{X})$
    $\mathbf{D}_{k+1}^{(i)}=\mathcal{P}_{\mathcal{D}}(\widehat{\mathbf{D}}_k^{(i)})+w_d(\mathbf{D}_k^{(i)}-\mathbf{D}_{k-1}^{(i)})$
    $k=k+1$
  end while
  $\mathbf{D}^{(i)}=\mathbf{D}_{k+1}^{(i)}$.
  3. Non-zero coefficients update:
  for $l=1,2,\cdots,n$ **do**
    $\Omega_l=\left\{i\ \middle|\ \mathbf{x}_{[l]}(i)\neq 0\right\}$
    $\mathbf{E}_l=\mathbf{Y}-\sum_{i\neq l}\mathbf{d}_i\mathbf{x}_{[i]}$
    $\mathbf{x}_{[l]}(\Omega_l)=\mathbf{d}_l^T\mathbf{E}_l(\Omega_l)$
  end for
end for
Output: $\mathbf{D}=\mathbf{D}^i$, $\mathbf{X}=\mathbf{X}^i$.

3 Simulations

In this section, the performance of our proposed DL algorithm is compared with K-SVD[2] and the algorithm proposed in [9], which is referred to as MBHPA-DL[3]. We consider a dictionary recovery experiment, in which, given a set of training signals generated by sparse linear combinations of the atoms in a known dictionary, the goal is to recover the underlying dictionary. Our simulations were performed in MATLAB R2013a environment on a system with 3.8 GHz Intel cori 7 CPU and 8 GB RAM, under Microsoft Windows 7 operating system. As a rough measure of complexity, we will report the runtimes of the algorithms.

[2] For K-SVD and OMP, we have used K-SVD-Box v10 and OMP-Box v10 available at `http://www.cs.technion.ac.il/~ronrubin/software.html`.

[3] The MATLAB implementation of our proposed algorithm together with those of the other compared algorithms will be made available at `https://sites.google.com/site/msaadeghii/`.

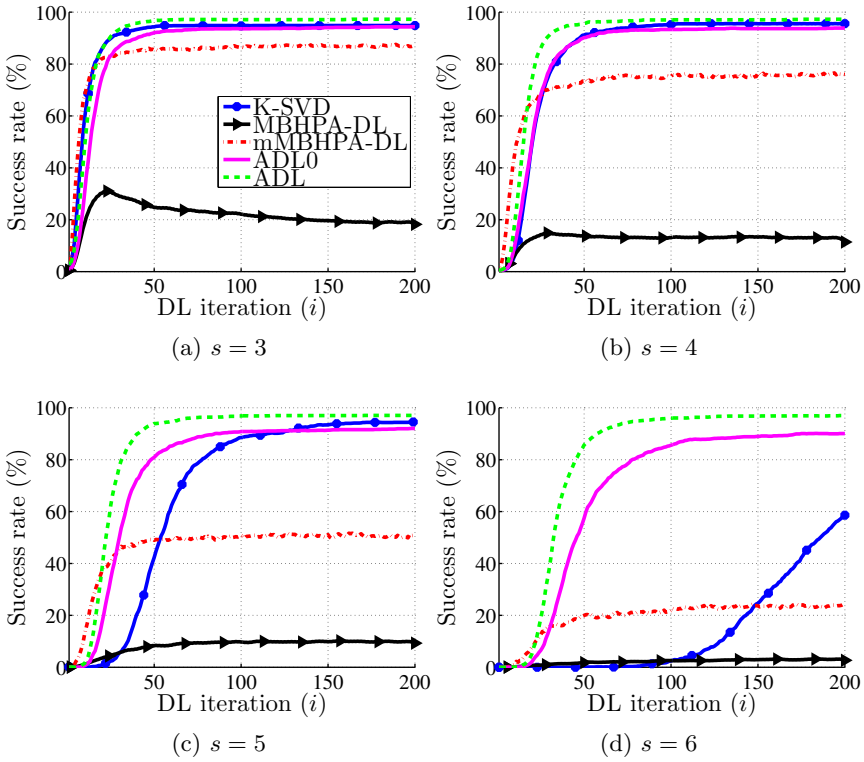

Fig. 1: Success rates in recovery of 20×50 dictionaries from a set of 1500 training signals, for different sparsity levels. The SNR is 100 dB. In this figure, ADL0 corresponds to ADL for $w_d = w_x = 0$.

Similar to [4], the underlying dictionary was generated as a random matrix of size 20×50, with zero-mean and unit-variance independent and identically distributed (i.i.d.) Gaussian entries. The dictionary was then normalized to have unit-norm columns. A collection of 1500 training signals, $\{\mathbf{y}_i\}_{i=1}^{1500}$, were produced, each as a linear combination of s different columns of the dictionary, with zero-mean and unit-variance i.i.d. Gaussian coefficients in uniformly random and independent positions. We varied s from 3 to 6. We then added white Gaussian noise with signal to noise ratio (SNR) levels of 100 dB and 20 dB. The algorithms were then applied onto these noisy training signals, and the resulting recovered dictionaries were compared to the generating dictionary as follows. Assume that $\mathbf{d}_i$ is a known atom and $\bar{\mathbf{d}}_i$ is the atom in the recovered dictionary that best matches $\mathbf{d}_i$ among the others. We say that the recovery is successful if $|\mathbf{d}_i^T \bar{\mathbf{d}}_i|$ is above 0.99 [4]. To evaluate the performance of the algorithms, we calculated the percentage of recovered atoms. We performed 200 iterations between the SR and DU steps for all the algorithms. The initial dictionary was

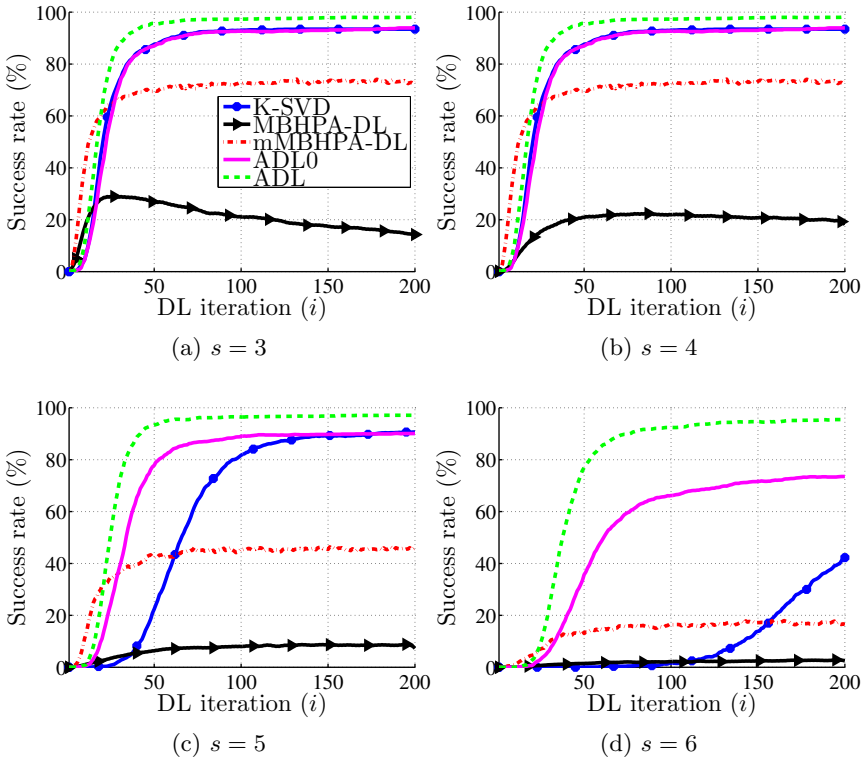

Fig. 2: Success rates in recovery of 20×50 dictionaries from a set of 1500 training signals, for different sparsity levels. The SNR is 20 dB. In this figure, ADL0 corresponds to ADL for $w_d = w_x = 0$.

made by randomly choosing different columns of the training signals followed by a normalization.

It should be noted that the original MBHPA-DL algorithm proposed in [9] performs only one iteration to update the coefficient matrix in SR step. This leads to a very slow convergence rate, usually ending up with a dictionary far away from the underlying one, which is confirmed by our simulations. As a solution, we modified MBHPA-DL's SR step by running it until a stopping criterion, similar to the one used in Algorithm 1, is satisfied. We call this version of the algorithm mMBHPA-DL, for modified MBHPA-DL.

The initial coefficient matrix for both ADL, MBHPA-DL, and mMBHPA-DL was set to the zero matrix. It was also observed that, in average, for all values of s, $\lambda = 0.1$ works well. The tolerances for terminating the SR and DU steps were set as $\tau_x = \tau_d = 0.005$. The weight parameters, w_x and w_d, in ADL were set to 0.85. In addition, to see the effect of using extrapolation and inertial accelerating

Table 1: Average runtimes (in second). In this table, ADL0 corresponds to ADL for $w_d = w_x = 0$.

Algorithm	K-SVD	mMBHPA-DL	ADL0	ADL
Runtime (s)	2.69	15.52	18.29	11.80

schemes, we executed ADL for $w_x = w_d = 0$. We refer to this version of ADL as ADL0.

The resulting success rates of the algorithms, averaged over 50 trials, versus DL iterations, and for two different noise levels of 100 dB and 20 dB, are shown in Figs. 1 and 2, respectively. The averaged runtimes are also reported in Table 1. Inspection of these results leads to several conclusions as follows:

- As clearly demonstrated in Figs. 1 and 2, mMBHPA-DL considerably outperforms the original MBHPA-DL algorithm.
- The proposed algorithm, for both cases of zero (ADL0) and non-zero weight parameters (ADL), is more successful in recovery of underlying dictionaries than K-SVD and mMBHPA-DL. Moreover, the use of non-zero weight parameters in ADL increases the convergence rate of the algorithm, especially for $s = 5$ and $s = 6$ and high noise level.
- While the performances of K-SVD and mMBHPA-DL deteriorate for $s = 5$ and $s = 6$, with that of mMBHPA-DL being more severe, ADL and ADL0 have promising recovery performances for all values of s. Indeed, ADL recovers the underlying dictionaries almost perfectly for all values of s, and both low and high noise levels.
- In terms of runtimes, K-SVD is the fastest algorithm. This is mainly due to the fact that, it uses the batch OMP algorithm [18] in its SR step, which is optimized for large training matrices. Moreover, ADL runs faster than mMBHPA-DL. Another noticeable remark is that, ADL0 has higher runtime than ADL. In fact, it takes more time for ADL0 to satisfy the stopping criteria of the SR and DU steps, because it does not utilize accelerated schemes.

4 Conclusion

This paper presented an accelerated dictionary learning (DL) algorithm based on iterative proximal algorithms to be used in sparse representation-based applications. Our proposed approach combines first-order proximal algorithms with accelerating inertial and extrapolation schemes to update coefficient matrix and dictionary alternatively, resulting in a simple, yet efficient DL algorithm. It was demonstrated through a dictionary recovery experiment that, compared with the well-known K-SVD [4] and a recently introduced algorithm [9], the proposed algorithm is more successful in recovering underlying dictionaries from a set of,

possibly noisy, training signals with different sparsity levels. Future works include applying the proposed algorithm to real-world applications such as image denoising, and establishing its convergence.

References

1. Candès, E.J., Wakin, M.B.: An introduction to compressive sampling. IEEE Signal Proc. Magazine **25**(2) (2008) 21–30
2. Rubinstein, R., Bruckstein, A.M., Elad, M.: Dictionaries for sparse representation modeling. Proceedings of the IEEE **98**(6) (2010) 1045–1057
3. Engan, K., Aase, S.O., Hakon Husoy, J.: Method of optimal directions for frame design. In: Proceedings of IEEE ICASSP. (1999)
4. Aharon, M., Elad, M., Bruckstein, A.: K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. IEEE Trans. on Signal Processing **54**(11) (2006) 4311–4322
5. Rubinstein, R., Zibulevsky, M., Elad, M.: Double sparsity: Learning sparse dictionaries for sparse signal approximation. IEEE Trans. on Signal Processing **58**(3) (2010) 1553–1564
6. Sadeghi, M., Babaie-Zadeh, M., Jutten, C.: Learning over-complete dictionaries based on atom-by-atom updating. IEEE Trans. on Si **62**(4) (2014) 883–891
7. Sadeghi, M., Babaie-Zadeh, M., Jutten, C.: Dictionary learning for sparse representation: A novel approach. IEEE Signal Proc. Letters **20**(12) (2013) 1195–1198
8. Yaghoobi, M., Blumensath, T., Davies, M.E.: Dictionary learning for sparse approximations with the majorization method. IEEE Trans. on Signal Processing **57**(6) (2009) 2178 – 2191
9. Bao, C., Ji, H., Quan, Y., Shen, Z.: Dictionary learning for sparse coding: Algorithms and convergence analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence **38**(7) (2015) 1356–1369
10. Pati, Y.C., Rezaiifar, R., Krishnaprasad, P.S.: Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition. In: In Proc. Asilomar Conf. Signal Syst. Comput. (1993)
11. Smith, L.N., Elad, M.: Improving dictionary learning: Multiple dictionary updates and coefficient reuse. IEEE Signal Proc. Letters **20**(1) (2013) 79–82
12. Parikh, N., Boyd, S.: Proximal algorithms. Foundations and Trends in Optimization **1**(3) (2014) 123–231
13. Xu, Y., Yin, W.: A globally convergent algorithm for nonconvex optimization based on block coordinate update. (2015)
14. Ochs, P., Chen, Y., Brox, T., Pock, T.: iPiano: Inertial proximal algorithm for nonconvex optimization. SIAM J. Imag. Sci. **7**(2) (2014) 388–1419
15. Meyer, C.D.: Matrix Analysis and Applied Linear Algebra. Society for Industrial and Applied Mathematics (SIAM) (2000)
16. Elad, M.: Sparse and Redundant Representations. Springer (2010)
17. Beck, A., Teboulle, M.: A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J. Imag. Sci. **2**(1) (2009) 183–202
18. Rubinstein, R., Zibulevsky, M., Elad, M.: Efficient implementation of the K-SVD algorithm using batch orthogonal matching pursuit. Technical report, Technion University (2008)