A Proximal Method for Convolutional Dictionary Learning with Convergence Property

Guan-Ju Peng
Department of Applied Mathmatics
National Chung Hsing University
South District, Taichung City 40227, Taiwan
Email: gjpeng@email.nchu.edu.tw

***Abstract*—The convolutional sparse coding (CSC) is superior in representing signals, and to obtain the best performance of CSC, the dictionary is usually learned from data. The so-called convolution dictionary learning (CDL) problem is therefore formulated for the purpose. Most of the solvers for CDL alternately update the coefficients and dictionary in an iterative manner, and as a consequence, numerous redundant iterations incur slow speed in achieving convergence. Moreover, their convergence properties can hardly be analyzed even though the ℓ_0 sparse inducing function is approximated by the convex ℓ_1 norm. In this article, we propose an algorithm which directly deals with the non-convex non-smooth ℓ_0 constraint and provides a sound convergence property. The experimental results show that, the proposed method achieves the convergence point with less time and a smaller final function value compared to the existing convolutional dictionary learning algorithms.**

I. Introduction

Given the data signals $\{\mathbf{y}_l\}$ and the dictionary elements $\{\mathbf{d}_m\}$, convolutional sparse representation represents a signal $\mathbf{y}_l$ as follows:

$$\mathbf{y}_l = \sum_m \mathbf{d}_m * \boldsymbol{\alpha}_{l,m}, \tag{1}$$

where $\boldsymbol{\alpha}_{l,m}$ is the coefficient map for $\mathbf{d}_m$ to approximate $\mathbf{y}_l$. The set of coefficient maps, denoted by $\{\boldsymbol{\alpha}_{l,m}\}$, can be derived by solving the *convolutional sparse coding* (CSC) problem, which is defined as follows:

$$\underset{\{\boldsymbol{\alpha}_{l,m}\}}{\arg\min} \frac{1}{2}\|\mathbf{y}_l - \sum_m \mathbf{d}_m * \boldsymbol{\alpha}_{l,m}\|_F^2 + \sum_m \Omega(\boldsymbol{\alpha}_{l,m}), \tag{2}$$

where Ω is the sparse inducing function. When Ω is the ℓ_0 function, the greedy algorithms can be used to solve CSC [1], [2]. In contrast, when the non-convex ℓ_0 function is approximated by the convex ℓ_1 norm [3], [4], the convex optimization algorithms, such as fast-iterative-shrinkage-thresholding-algorithm (FISTA) [5] and feature-sign algorithm [6], can be adopted to solve CSC. More algorithms solving CSC using convex optimization can be found in [7], [8], [9].

To improve the performance of representing signals, one of the major concerns for convolutional sparse representation is the selection of dictionary. When the property of signals is known and unchanged, a dictionary describing a fixed morphology, such as wavelets [10] and curvelets [11], can be used. In contrast, when there is no assumption on the morphology of signals, the dictionary can be *learned* from the received signals by solving the *convolutional dictionary learning* (CDL) problem, which is defined as follows:

$$\begin{aligned}\underset{\{\boldsymbol{\alpha}_{l,m}\},\{\mathbf{d}_m\}}{\arg\min} \quad & \sum_l \{\frac{1}{2}\|\sum_m \mathbf{d}_m * \boldsymbol{\alpha}_{l,m} - y_l\|^2 \\ & +\lambda_1 \sum_m \Omega(\boldsymbol{\alpha}_{l,m})\} + \lambda_2 \sum_m \Gamma_{\mathcal{C}}(\mathbf{d}_m),\end{aligned} \tag{3}$$

where $\mathcal{C}$ denotes the unit-norm sphere (i.e., $\|\mathbf{d}_m\| = 1$), and $\Gamma_{\mathcal{C}}$ is an indicator function defined as follows:

$$\Gamma_{\mathcal{C}}(\mathbf{x}) = \begin{cases} 0 & \text{if } \mathbf{x} \in \mathcal{C}, \\ \infty & \text{otherwise.} \end{cases} \tag{4}$$

The mainstream algorithms solve CDL in an iterative manner, and each iteration comprises of two steps: convolutional sparse coding step and convolutional dictionary updating step [7], [9], [12]. In the first step, the dictionary is kept unchanged, and the coefficient maps for each signal can be updated using the algorithms for CSC. In the second step, the coefficient maps derived in the first step is used to update the dictionary, and $\mathcal{C}$, a non-convex unit-norm sphere, is approximated by a convex unit-norm ball (i.e., $\|\mathbf{d}_m\| \leqslant 1$) such that convex optimization algorithms can adopted to solve the dictionary updating step.

To improve the computational efficiency, convolution is usually calculated in Fourier domain, and thereby two additional constraints, which are $\Gamma_{\mathcal{C}_\alpha}$ and $\Gamma_{\mathcal{C}_\mathbf{d}}$, are inserted to the CDL problem to maintain the equivalence of the variables in the spatial and the Fourier domains. The resulting alternative optimization problem of CDL can be formulated as follows:

$$\begin{aligned}\underset{\{\boldsymbol{\alpha}_{l,m}\},\{\mathbf{d}_m\}}{\arg\min} \quad & \sum_l \{\frac{1}{2}\|\sum_m \mathbf{d}_m * \boldsymbol{\alpha}_{l,m} - y_l\|^2 \\ & + \sum_m \lambda_1(\Omega + \Gamma_{\mathcal{C}_\alpha})(\boldsymbol{\alpha}_{l,m})\} + \\ & + \sum_m \{\lambda_2(\Gamma_{\mathcal{C}} + \Gamma_{\mathcal{C}_\mathbf{d}})(\mathbf{d}_m)\},\end{aligned} \tag{5}$$

where $\mathcal{C}_\alpha$ and $\mathcal{C}_\mathbf{d}$ are defined as follows:

$$\mathcal{C}_{\boldsymbol{\alpha}} = \{\boldsymbol{\alpha} : (\mathbf{I} - \mathbf{P}_{\boldsymbol{\alpha}}\mathbf{P}_{\boldsymbol{\alpha}}^\intercal)\boldsymbol{\alpha} = \mathbf{0}\}, \tag{6}$$

$$\mathcal{C}_\mathbf{d} = \{\mathbf{u} : (\mathbf{I} - \mathbf{P}_\mathbf{d}\mathbf{P}_\mathbf{d}^\intercal)\mathbf{u} = \mathbf{0}\}, \tag{7}$$

and in accordance to the convolution theorem, $\mathbf{P}_{\boldsymbol{\alpha}}$ and $\mathbf{P}_\mathbf{d}$ respectively perform zero padding to coefficient maps and dictionary elements.

ISBN 978-90-827970-1-5

Most algorithms solving Equation (3) or (5) were built under the framework of alternate optimization, where different methods are deployed in either or both of the convolutional sparse coding stage or the convolutional dictionary updating stage [7], [8], [9], [12], [13], [14]. However, three major artifacts are usually incurred by these algorithms:

1) Non-sparse representations: Approximating non-convex constraints with convex ones may cause non-sparse representations of signals, which usually hinders the performance for applications [15].
2) Redundant iterations: When the dictionary and coefficient maps are initialized at random, redundant iterations are incurred by the alternate optimization and thus slower the speed to convergence.
3) Missing convergence property: Although the convergence can be obtained respectively for the first and second steps, the global convergence properties of these algorithms are still missing.

In the present paper, we try to resolve the above artifacts, and our main contributions are summarized as follows:

- Based on forward-backward splitting, we propose an algorithm, denoted by **FB**, to solve CDL involving non-convex constraints. Unlike the mainstream algorithms which adopt alternate optimization, our **FB** algorithm jointly update the dictionary elements and coefficient maps in each iteration.
- We demonstrate that **FB** has the convergence property.
- Numerical results demonstrate that **FB** outperforms the existing methods with regard to convergence speed and the final function value.

The reminder of this paper is organized as follows. In Section II, we demonstrate our **FB** algorithm, and show its convergence property. In Section III, we compare the proposed method with the existing CDL algorithms with regard to the convergence speed and the final function value. Section IV gives the concluding remarks.

II. Forward-Backward Splitting for Convolutional Dictionary Learning with Non-convex Constraints

We briefly review the non-convex forward-backward splitting framework, which is designed to solve problems of the following form:

$$\arg\min_{\mathbf{x}} f(\mathbf{x}) + g(\mathbf{x}), \tag{8}$$

where f is a non-convex C^1 function whose gradient, ∇f, is Lipschitz continuous, and g is a non-convex, non-smooth, proper closed function. Let L_f denote the global Lipschitz constant of the gradient ∇f and assume there is a set $\{\eta_t\}$ where $0 < \eta_t < \frac{1}{L_f}$ for all t. We consider a sequence $\{\mathbf{x}^t\}$ satisfying the following equations:

$$\begin{aligned} & g(\mathbf{x}^{t+1}) + \langle \mathbf{x}^{t+1} - \mathbf{x}^t, \nabla f(\mathbf{x}^t) \rangle + \frac{1}{2\eta_t} \|\mathbf{x}^{t+1} - \mathbf{x}^t\|^2 \\ & \leqslant g(\mathbf{x}^t), \end{aligned} \tag{9}$$

and

$$\exists \mathbf{z}^{t+1} \in \partial g(\mathbf{x}^{t+1}), \|\mathbf{z}^{t+1} + \nabla f(\mathbf{x}^t)\| \leqslant \eta_t \|\mathbf{x}^{t+1} - \mathbf{x}^t\|, \tag{10}$$

where $\partial g(\mathbf{x}^{t+1})$ denotes the *subdifferential* of f at $\mathbf{x}^{t+1}$. Then we take into account of the following conditions:

Condition II.1. *(Conditions of f and g).*

1) $f + g$ is a proper closed function bounded from below,
2) $f + g$ is a Kurdyka-Łojasiewicz (KL) function,
3) $f \in [0, \infty)$,
4) f is a C^1 function with the global Lipschitz constant L_f,
5) dom g is continuous.

The results in [16] show that if f and g satisfy the above conditions, a sequence $\{\mathbf{x}^t\}$ satisfying Equations (9) and (10) has the following convergence property:

1) $\{\mathbf{x}^t\}$ converge to a critical point of $f + g$.
2) $\{\mathbf{x}^t\}$ has a finite length, i.e.

$$\sum_{t=1}^{\infty} \|\mathbf{x}^{t+1} - \mathbf{x}^t\| < \infty. \tag{11}$$

Furthermore, such kind of sequence can be generated using the following equation in an iterative manner:

$$\mathbf{x}^{t+1} \leftarrow \text{prox}_{\eta_t g}(\mathbf{x}^t - \eta_t \nabla f(\mathbf{x}^t)). \tag{12}$$

To formulate the objective function of CDL into the framework in Equation (8), we let $\mathbf{x} = (\{\mathbf{d}_m\}, \{\boldsymbol{\alpha}_{l,m}\})$ and therefore f and g can be respectively defined as follows:

$$f(\{\mathbf{d}_m\}, \{\boldsymbol{\alpha}_{l,m}\}) = \frac{1}{2} \sum_l \| \sum_m \mathbf{d}_m * \boldsymbol{\alpha}_{l,m} - \mathbf{y}_l \|^2, \tag{13}$$

and

$$\begin{aligned} g(\{\mathbf{d}_m\}, \{\boldsymbol{\alpha}_{l,m}\}) \quad = \quad & \sum_l \{ \sum_m \lambda_1 (\Omega + \Gamma_{\mathcal{C}_\alpha})(\boldsymbol{\alpha}_{l,m}) \} \\ & + \sum_m \{ \lambda_2 (\Gamma_{\mathcal{C}} + \Gamma_{\mathcal{C}_\mathbf{d}})(\mathbf{d}_m) \} \end{aligned} \tag{14}$$

Then we may generate the sequence $\{\{\mathbf{d}_m^t\}, \{\boldsymbol{\alpha}_{l,m}^t\}\}$ using Equation (12). We define the gradient of f as follows:

$$\nabla f := \{ \{ \nabla_{\mathbf{d}_m} f \}, \{ \nabla_{\boldsymbol{\alpha}_{l,m}} f \} \}, \tag{15}$$

where $\nabla_{\mathbf{d}_m} f$ and $\nabla_{\boldsymbol{\alpha}_{l,m}} f$ for each l and m can be calculated as follows:

$$\begin{aligned} & \nabla_{\mathbf{d}_a} f(\{\mathbf{d}_m\}, \{\boldsymbol{\alpha}_{l,m}\}) \\ = \quad & \sum_l ((\sum_m \mathbf{d}_m * \boldsymbol{\alpha}_{l,m} - \mathbf{y}_l)^\intercal (\mathbf{I} * \boldsymbol{\alpha}_{l,a})), \end{aligned} \tag{16}$$

and

$$\begin{aligned} & \nabla_{\boldsymbol{\alpha}_{a,b}} f(\{\mathbf{d}_m\}, \{\boldsymbol{\alpha}_{l,m}\}) \\ = \quad & (\sum_m \mathbf{d}_m * \boldsymbol{\alpha}_{a,m} - \mathbf{y}_a) * (\mathbf{I} * \mathbf{d}_b). \end{aligned} \tag{17}$$

These gradients can be calculated in the Fourier domain to improve the computational efficiency.

ISBN 978-90-827970-1-5 © EURASIP 2018

We define the intermediate variables for each l and m as follows:

$$\mathbf{d}_m^{t+\frac{1}{2}} := \mathbf{d}_m^t - \eta_t \nabla_{\mathbf{d}_m} f(\{\mathbf{d}_m\}, \{\boldsymbol{\alpha}_{l,m}\}), \quad (18)$$

$$\boldsymbol{\alpha}_{l,m}^{t+\frac{1}{2}} := \boldsymbol{\alpha}_{l,m}^t - \eta_t \nabla_{\boldsymbol{\alpha}_{l,m}} f(\{\mathbf{d}_m\}, \{\boldsymbol{\alpha}_{l,m}\}). \quad (19)$$

Then the variables used in the following iteration are calculated as follows:

$$(\{\mathbf{d}_m^{t+1}\}, \{\boldsymbol{\alpha}_{l,m}^{t+1}\}) = \text{prox}_{\eta_t g}(\{\mathbf{d}_m^{t+\frac{1}{2}}\}, \{\boldsymbol{\alpha}_{l,m}^{t+\frac{1}{2}}\}), \quad (20)$$

where $\text{prox}_{\eta_t g}$ can be calculated using the derivation in Appendix. Then we may formulate our **FB** algorithm in Algorithm 1.

Algorithm 1 FB for the CDL problem.

Require: initial dictionary, $\{\mathbf{d}_m^0\}$; initial coefficient maps, $\{\boldsymbol{\alpha}_{l,m}^0\}$; training signals, $\{\mathbf{y}_l\}$; and descent parameter, η;
Ensure: learned dictionary, $\{\mathbf{d}_m^t\}$; learned coefficients, $\{\boldsymbol{\alpha}_{l,m}^t\}$;

1: $t \leftarrow 0$.
2: **repeat**
3: $\eta_t \leftarrow \eta$
4: Compute $(\{\mathbf{d}_m^{t+\frac{1}{2}}\}, \{\boldsymbol{\alpha}_{l,m}^{t+\frac{1}{2}}\})$ from $(\{\mathbf{d}_m^t\}, \{\boldsymbol{\alpha}_{l,m}^t\})$ using Equations (18) and (19).
5: Compute $(\{\mathbf{d}_m^{t+1}\}, \{\boldsymbol{\alpha}_{l,m}^{t+1}\})$ from $(\{\mathbf{d}_m^{t+\frac{1}{2}}\}, \{\boldsymbol{\alpha}_{l,m}^{t+\frac{1}{2}}\})$ using Equation (21).
6: $t \leftarrow t + 1$.
7: **until** convergence.
8: **return** $\{\mathbf{d}_m^t\}$ and $\{\boldsymbol{\alpha}_{l,m}^t\}$.

Theorem II.2 (Convergence of Algorithm 1)**.** *When the objective function of CDL is split into f and g as the definitions in Equations (13) and (14), and η is set at a small enough value, the sequence generated by Algorithm 1 converges to a critical point and has a finite length.*

The above theorem can be proved by showing that f and g satisfy the properties listed in Condition II.1. Nevertheless, the details are omitted due to the limitation of the paper length.

III. Performance Evaluation

In this section, we compare the following dictionary learning algorithms for convolutional sparse representation:

- **ADMM-ℓ_1** denotes the method proposed in [12], where the sparse inducing function Ω is the ℓ_1 norm.
- **ADMM-ℓ_0** denotes the method proposed in [12], where the sparse inducing function Ω is the ℓ_0 norm.
- **AO** denotes the method proposed in [7], where the sparse inducing function Ω is the ℓ_0 norm.
- **FB** denotes the proposed method where the sparse inducing function Ω is the ℓ_0 norm.

The performances of these algorithms are evaluated with regard to the speed to achieve convergence and the final function value.

A. Configuration of Learning Algorithms

ADMM-ℓ_1, **ADMM-ℓ_0**, and **AO** adopt the two-step alternate optimization scheme. **ADMM-ℓ_1** and **ADMM-ℓ_0** use ADMM in both of the steps, and the ADMM parameter is set at 0.01. In **AO**, COD is used in dictionary updating while the convolutional FIESTA is used in coefficient maps updating. The parameter is set at 0.1 in the optimization procedures of both steps. For the proposed **FB**, we set η at 0.01.

For all algorithms in the experiment, each entry of the initial dictionary is generated using an i.i.d uniformly distributed random variable. The entries of initial coefficient maps are all set at 0.

For **ADMM-ℓ_1** and **ADMM-ℓ_0**, total 200 iterations are performed, and in each iteration, the coefficient maps and the dictionary are respectively updated only once. For **AO**, only 33 iterations are performed because each iteration, where the first 15 internal iterations are performed in using FIESTA to update the coefficient maps, and the second 15 internal iterations are performed in using COD to update the dictionary, requires much more time than the other methods. For our **FB** algorithm, total 200 iterations are performed.

B. Data Preprocessing

The elements of $\{\mathbf{y}_l\}$ are obtained from natural images. As the method used in [17], each image is decomposed into a high frequency component and a low frequency component using a low pass filter, and only the high frequency part is used to learn the convolutional dictionary. After the decomposition, each high frequency component is normalized using the Frobenius norm, and therefore the entry values range from 0 to 1.

In our experiment, $\{\mathbf{y}_l\}$ contains 10 normalized high frequency components extracted from 10 grey scale images of 256×256 pixels. The dictionary $\{\mathbf{d}_m\}$ contains 32 or 64 elements, and the size of each element can be 8×8 or 16×16.

C. Experimental Results

In the experiment, we compare the performances of **ADMM-ℓ_1**, **ADMM-ℓ_0**, **AO**, and **FB** with regard to computing time and final function value. In calculating the function value, Ω is the ℓ_0 norm.

Table I demonstrates the final function values and the computing times of the comparable methods using different settings of dictionary size and Lagrangian parameter. Note that for each setting, 100 trails are executed with different random initial dictionaries, and the average of these trails are shown in the table. The results indicate that compared to the other methods, the proposed **FB** achieves the point with the smallest final function value and in doing so takes less than half computing time.

In Figure 1, we demonstrate the value of $\log(f + g)$ in each iteration during the dictionary learning procedure. The proposed **FB** approach is very stable and getting close to the convergence point after performing only few iterations. **FB** also has the smallest function values in all iterations.

The **AO** approach also generates a stable sequence including decreasing function values. However, the speed of decrement

ISBN 978-90-827970-1-5 © EURASIP 2018

TABLE I
COMPARISON OF FINAL FUNCTIONAL VALUES & COMPUTING TIMES.

(a) $\lambda_1 = 0.005$ (left), and $\lambda_1 = 0.01$ (right)

$32, 8 \times 8$	$f+g$	Time (s)	$32, 8 \times 8$	$f+g$	Time (s)
ADMM-ℓ_1	4814	719	**ADMM**-ℓ_1	8202	723
ADMM-ℓ_0	5391	889	**ADMM**-ℓ_0	7649	885
AO	8101	728	**AO**	9557	721
FB	4445	331	**FB**	4683	323
$64, 8 \times 8$	$f+g$	Time (s)	$64, 8 \times 8$	$f+g$	Time (s)
ADMM-ℓ_1	5359	1443	**ADMM**-ℓ_1	9222	1450
ADMM-ℓ_0	6245	1737	**ADMM**-ℓ_0	8167	1795
AO	7616	1417	**AO**	7957	1428
FB	4343	638	**FB**	4393	680
$32, 16 \times 16$	$f+g$	Time (s)	$32, 16 \times 16$	$f+g$	Time (s)
ADMM-ℓ_1	4178	763	**ADMM**-ℓ_1	7005	722
ADMM-ℓ_0	3327	938	**ADMM**-ℓ_0	4994	865
AO	5266	771	**AO**	6173	709
FB	2660	327	**FB**	2740	323
$64, 16 \times 16$	$f+g$	Time (s)	$64, 16 \times 16$	$f+g$	Time (s)
ADMM-ℓ_1	4279	1443	**ADMM**-ℓ_1	7394	1445
ADMM-ℓ_0	5146	1748	**ADMM**-ℓ_0	6578	1736
AO	4762	1409	**AO**	5079	1409
FB	2167	638	**FB**	2843	652

(b) $\lambda_1 = 0.05$ (left), and $\lambda_1 = 0.1$ (right)

$32, 8 \times 8$	$f+g$	Time (s)	$32, 8 \times 8$	$f+g$	Time (s)
ADMM-ℓ_1	28024	725	**ADMM**-ℓ_1	37283	723
ADMM-ℓ_0	6554	866	**ADMM**-ℓ_0	6826	871
AO	13762	706	**AO**	12787	708
FB	5582	315	**FB**	6160	315
$64, 8 \times 8$	$f+g$	Time (s)	$64, 8 \times 8$	$f+g$	Time (s)
ADMM-ℓ_1	28726	1465	**ADMM**-ℓ_1	39154	1446
ADMM-ℓ_0	7288	1738	**ADMM**-ℓ_0	6900	1741
AO	12564	1407	**AO**	12414	1411
FB	6119	631	**FB**	6378	649
$32, 16 \times 16$	$f+g$	Time (s)	$32, 16 \times 16$	$f+g$	Time (s)
ADMM-ℓ_1	19808	720	**ADMM**-ℓ_1	29046	718
ADMM-ℓ_0	6269	864	**ADMM**-ℓ_0	6773	865
AO	7209	706	**AO**	7391	704
FB	5252	315	**FB**	5981	317
$64, 16 \times 16$	$f+g$	Time (s)	$64, 16 \times 16$	$f+g$	Time (s)
ADMM-ℓ_1	22013	1433	**ADMM**-ℓ_1	27903	1441
ADMM-ℓ_0	6968	1727	**ADMM**-ℓ_0	6888	1734
AO	6061	1408	**AO**	6683	1410
FB	5963	631	**FB**	6062	627

is slow owing to the redundant iterations. Compared to the other methods, using **ADMM**-ℓ_1 in the dictionary learning procedure causes much larger function values because the ℓ_1 function, which is used as the approximation of the ℓ_0 norm, usually causes non-sparse representations of real-world signals. In contrast, although **ADMM**-ℓ_0 deals with the ℓ_0 constraint directly, it is unable to generate a sequence of decreasing function values. In fact, the sequence of function values generated using **ADMM**-ℓ_0 is very unstable, and even more iterations are performed, it can result in a worse dictionary.

IV. CONCLUDING REMARKS

The mainstream algorithms adopted the two-step alternate optimization scheme to solve the dictionary learning problem for convolutional sparse representation. However, such kind of scheme may cause redundant internal iterations and therefore reduced the speeds of these algorithms to achieve convergence. Moreover, these algorithms dealt with the problem where non-convex constraints were approximated with convex ones. This meant that the ℓ_0 constraint imposed on the coefficients was approximated by the ℓ_1 function, and the unit-norm sphere constraint imposed on the length of dictionary element was approximated by the unit-norm ball. When handling real-world signals, non-sparse representations were usually caused by these algorithms, and thus hindered the performances of signal processing applications. In this paper, we proposed an approach called **FB** to directly solve the convolutional dictionary learning problem involving the non-convex constraints. The proposed **FB** adopted the forward-backward splitting framework because we found that the objective function of the convolutional dictionary learning problem met the requirements of the framework to obtain the convergence property. In the experiments, we compared the performances of the existing methods with **FB** with regard to the final function value and the computing time, and the results indicated that the proposed **FB** outperformed the other methods in all cases. In our future work, we will extend **FB** using parameter adaption to further improve the speed of the algorithm to achieve convergence.

APPENDIX

The definition of the proximal mapping (denoted by prox) can be found in [16], and thereby $\text{prox}_{\eta_t g}$ can be further split as follows:

$$\text{prox}_{\eta_t g}(\{\mathbf{d}_m^{t+\frac{1}{2}}\}, \{\boldsymbol{\alpha}_{l,m}^{t+\frac{1}{2}}\}) = (\{\text{prox}_{\eta_t \lambda_2(\Gamma_{\mathcal{C}} + \Gamma_{\mathcal{C}_\mathbf{d}})}(\mathbf{d}_m^{t+\frac{1}{2}})\}, \{\text{prox}_{\eta_t \lambda_1(\|\cdot\|_0 + \Gamma_{\mathcal{C}_{\boldsymbol{\alpha}}})}(\boldsymbol{\alpha}_{l,m}^{t+\frac{1}{2}})\}),$$

and if we consider the properties of the constraint functions composing g, their proximal mappings can be further decoupled as follows:

$$\text{prox}_{\eta_t \lambda_2(\Gamma_{\mathcal{C}} + \Gamma_{\mathcal{C}_\mathbf{d}})} = \text{prox}_{\Gamma_{\mathcal{C}}} \circ \text{prox}_{\Gamma_{\mathcal{C}_\mathbf{d}}}, \quad (21)$$

$$\text{prox}_{\eta_t \lambda_1(\|\cdot\|_0 + \Gamma_{\mathcal{C}_{\boldsymbol{\alpha}}})} = \text{prox}_{\eta_t \lambda_1 \|\cdot\|_0} \circ \text{prox}_{\Gamma_{\mathcal{C}_{\boldsymbol{\alpha}}}}. \quad (22)$$

ISBN 978-90-827970-1-5

The proximal mapping of an indicator function with regard to a closed set is the projection to that set, and therefore $\text{prox}_{\Gamma_{\mathcal{C}_{\mathbf{d}}}}$, $\text{prox}_{\Gamma_{\mathcal{C}_{\boldsymbol{\alpha}}}}$, and $\text{prox}_{\Gamma_{\mathcal{C}}}$ can be calculated as follows:

$$\text{prox}_{\Gamma_{\mathcal{C}_{\boldsymbol{\pi}_{\mathbf{d}}}}}(\mathbf{x}) = \mathbf{P_d}\mathbf{P_d^\intercal}(\mathbf{x}), \quad (23)$$

$$\text{prox}_{\Gamma_{\mathcal{C}_{\boldsymbol{\pi}_{\boldsymbol{\alpha}}}}}(\mathbf{x}) = \mathbf{P}_{\boldsymbol{\alpha}}\mathbf{P}_{\boldsymbol{\alpha}}^\intercal(\mathbf{x}), \quad (24)$$

$$\text{prox}_{\Gamma_{\mathcal{C}}}(\mathbf{x}) = \frac{\mathbf{x}}{\|\mathbf{x}\|^2}, \text{ if } \mathbf{x} \neq \mathbf{0}. \quad (25)$$

Note that $\mathbf{P_d}\mathbf{P_d^\intercal}(\mathbf{x})$ and $\mathbf{P}_{\boldsymbol{\alpha}}\mathbf{P}_{\boldsymbol{\alpha}}^\intercal(\mathbf{x})$ can be obtained by enforcing the values of padded positions to be zeros. Let $\mathbf{x} = (x_1, x_2, \cdots, x_N) \in \mathbb{R}^N$, and then $\text{prox}_{\eta_t\lambda_1\|\cdot\|_0}(\mathbf{x})$ can be further decoupled as follows:

$$\text{prox}_{\eta_t\lambda_1\|\cdot\|_0}(\mathbf{x}) = (\text{prox}_{\eta_t\lambda_1|\cdot|_0}(x_1), \cdots, \text{prox}_{\eta_t\lambda_1\|\cdot\|_0}(x_N)), \quad (26)$$

where for the scalar $x \in \mathbb{R}$, $\text{prox}_{\eta_t\lambda_1|\cdot|_0}(x)$ is calculated as follows:

$$\text{prox}_{\eta_t\lambda_1\|\cdot\|_0}(x) = \begin{cases} x & \text{if } |x| > \sqrt{2\eta_t\lambda_1} \\ \{x, 0\} & \text{if } |x| = \sqrt{2\eta_t\lambda_1} \\ 0 & \text{otherwise.} \end{cases}$$

REFERENCES

[1] A. Szlam, K. Kavukcuoglu, and Y. LeCun, "Convolutional matching pursuit and dictionary training," *Computer Research Repository (arXiv)*, 2010.

[2] Q. Barthélemy, A. Larue, A. Mayoue, D. Mercier, and J. I. Mars, "Shift & 2d rotation invariant sparse coding for multivariate signals," *IEEE Transactions on Signal Processing*, vol. 60, no. 4, pp. 1597–1611, 2012.

[3] D. L. Donoho and X. Huo, "Uncertainty principles and ideal atomic decomposition," *IEEE Transactions on Information Theory*, vol. 47, no. 7, pp. 2845–2862, 2001.

[4] V. Papyan, J. Sulam, and M. Elad, "Working locally thinking globally: Theoretical guarantees for convolutional sparse coding," *IEEE Transactions on Signal Processing*, vol. 65, no. 21, pp. 5687–5701, Nov 2017.

[5] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.

[6] H. Lee, A. Battle, R. Raina, and A. Y. Ng, "Efficient sparse coding algorithms," in *Proceedings of the 19th International Conference on Neural Information Processing Systems*, ser. NIPS'06. Cambridge, MA, USA: MIT Press, 2006, pp. 801–808.

[7] R. Chalasani, J. C. Principe, and N. Ramakrishnan, "A fast proximal method for convolutional sparse coding," in *Neural Networks (IJCNN), The 2013 International Joint Conference on*. IEEE, 2013, pp. 1–5.

[8] H. Bristow, A. Eriksson, and S. Lucey, "Fast convolutional sparse coding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 391–398.

[9] F. Heide, W. Heidrich, and G. Wetzstein, "Fast and flexible convolutional sparse coding," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 5135–5143.

[10] S. Mallat, *A wavelet tour of signal processing*. Academic Press, 1998.

[11] E. Candes, L. Demanet, D. Donoho, and L. Ying, "Fast discrete curvelet transforms," *Multiscale Modeling & Simulation*, vol. 5, no. 3, pp. 861–899, 2006.

[12] B. Wohlberg, "Efficient algorithms for convolutional sparse representations," *IEEE Transactions on Image Processing*, vol. 25, no. 1, pp. 301–315, 2016.

[13] H. Bristow and S. Lucey, "Optimization methods for convolutional sparse coding," *arXiv preprint arXiv:1406.2407*, 2014.

[14] I. Y. Chun and J. A. Fessler, "Convolutional dictionary learning: Acceleration and convergence," *IEEE Transactions on Image Processing*, vol. 27, no. 4, pp. 1697–1712, April 2018.

[15] T. Zhang, "Analysis of multi-stage convex relaxation for sparse regularization," *Journal of Machine Learning Research*, vol. 11, no. Mar, pp. 1081–1107, 2010.

[16] H. Attouch, J. Bolte, and B. F. Svaiter, "Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized gauss–seidel methods," *Mathematical Programming*, vol. 137, no. 1-2, pp. 91–129, 2013.

[17] H. Zhang and V. M. Patel, "Convolutional sparse and low-rank coding-based rain streak removal," in *Applications of Computer Vision (WACV), 2017 IEEE Winter Conference on*. IEEE, 2017, pp. 1259–1267.

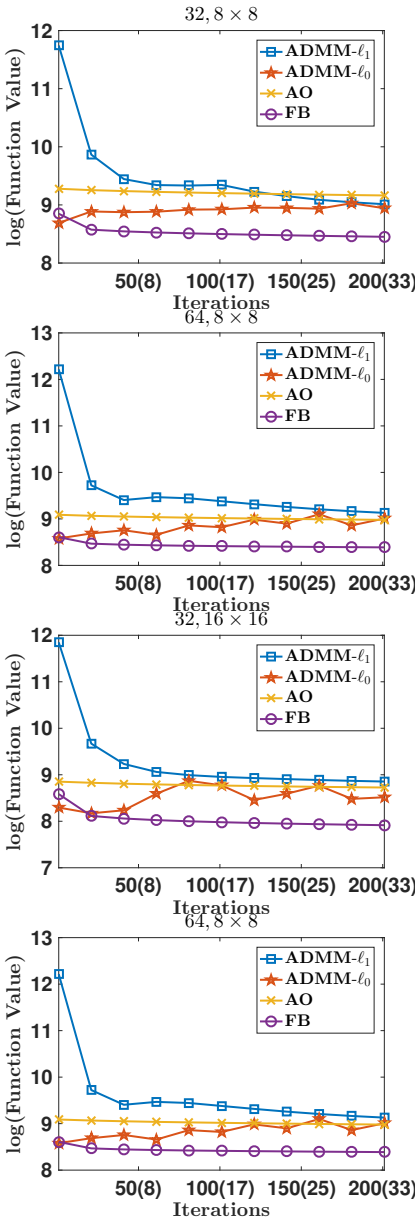

Fig. 1. Comparison of $log(f + g)$ and iteration number in the dictionary learning procedure. λ_1 is set at 0.01. 200 iterations are performed for **ADMM**-ℓ_1, **ADMM**-ℓ_0, and **FB**, while only 33 iterations is performed for **AO**.