An Inexact Proximal Alternating Direction Method for Non-convex and Non-smooth Matrix Factorization and Beyond

Yiyang Wang
School of Mathematical Sciences
Dalian University of Technology
yywerica@gmail.com

Risheng Liu
School of Software Technology
Dalian University of Technology
rsliu@dlut.edu.cn

Xiaoliang Song
School of Mathematical Sciences
Dalian University of Technology
songxiaoliang@mail.dlut.edu.cn

Zhixun Su
School of Mathematical Sciences
Dalian University of Technology
zxsu@dlut.edu.cn

Abstract

Since Non-convex and Non-smooth Matrix Factorization (NNMF) problems have great realistic significance in applications, they attract extensive attention in the fields of image processing and machine learning. We in this paper propose an inexact proximal alternating direction (IPAD) method for solving various complex NNMF problems. Our IPAD method is not a single algorithm, but a general and flexible framework which can fuse various numerical methods into. With a special designed error condition, the convergence properties of IPAD are analyzed for a general formulation, and can be extended to a wider range of problems. Moreover, an implementation method for checking the inexactness criterion is theoretically analyzed, which is more valid than the previously naive criteria in practice. Our IPAD algorithm is applied to a widely-concerned sparse dictionary learning problem on both synthetic and real-world data. The experimental results with detailed analyses and discussions are given to verify the efficiency of IPAD method.

1. Introduction

Being as an imperative tool for multivariate data analysis, matrix factorization (MF) has been widely used in the fields of pattern recognition and machine learning, signal and image processing, etc. More specifically, such applications include gene expression classification [5], hyperspectral unmixing [19], image restoration [30], and so on.

Considering the ill-posedness of the classic MF [11, 22] which factorizes the matrices without additional constraints, not a few literatures concentrate on introducing auxiliary constraints on the decomposed matrices in order to incorporate prior knowledge and reflect the characteristics of the issues more comprehensively [9, 18, 36]. Before revisiting some well-known MF techniques, we provide the general formulation of MF as follows:

$$\min_{A \in S_1, B \in S_2} L(I - AB) + r_1(A) + r_2(B),$$

where the regularization functions $r_1$ and $r_2$ denote the additional constraints that respectively act on the to-be factorized vectors/matrices $A$ and $B$; at the same time, $A$ and $B$ are restricted to the constraint sets $S_1$ and $S_2$; the coupling part in the loss function $L$ is employed to describe the data recovery ability of $AB$ to the input data $I$.

This general formulation covers numerous MF techniques. One of the most well-known MF techniques, principal component analysis (PCA) [22], actually does not make any assumptions on the factorized components, however, it can also be written in the form of (1). On the other hand, the variants of sparse PCA (SPCA) incorporate sparse constraints, such as lasso constraint [29], $\ell_0$ penalty [21] and elastic-net regularization [37] to the classic PCA [22]. In nonnegative matrix factorization (NMF), both components are constrained to be non-negative in accordance with biological evidences [23, 33]. The two factorized components in sparse dictionary learning (SDL) are called “dictionary” and “codes”, respectively. Many literatures have posted the superiority on restricting the dictionary with normalized bases while at the same time constraining the sparsity of codes with various non-convex sparse penalties [30, 2]. Since SDL is a powerful tool to represent diverse kinds of data in image processing, machine learning and neural computing [16], its algorithms and applications have gained wider attentions and further discussions from researchers.

MF problems formulated as (1) with simple or without
constraints have been relatively well-studied [18]; their designed algorithms usually yield satisfactory results [23, 31]. However, the simple-constrained or unconstrained MF techniques are far less effective than the ones with complex constraints. The special-designed constraints, absorbing various prior knowledge and reflecting the properties of the issues in a more comprehensive way, have great realistic significance and play an important role in real-world applications. In SDL problem, for example, using sparse and non-convex regularizers enable much more flexibility to adapt the representation to the data under an over-complete dictionary [30]. An orthogonal constraint is added on NMF to minimize the redundancy between different bases [24] and then result in some sparseness in the factorized components. Those effective but complex constraints, e.g., the non-convex space regularization in SDL and the orthogonal constraint in NMF, mainly exist in the regularization part and are always non-smooth and even non-convex functions [21, 30]. Due to their complexities, few algorithms are designed for solving the non-convex and non-smooth MF (NNMF) problems with rigid convergence analyses. Considering the great realistic significance of NNMF problems in practice, we in this paper focus on proposing a general algorithm framework for effectively and efficiently solving these complex NNMF problems.

From the observation that it is either impossible or extremely hard for calculating the exact solutions of the sub-problems in NNMF, but the subproblems can be easily solved by other efficient numerical methods. Thus, we employ an alternating method to update each variable in a cyclic order; and each of the two subproblems are allowed to be solved approximately. We name our algorithm framework as inexact proximal alternating direction (IPAD) method, which is originally designed for solving NNMF problems, but can be extended to solving a wider range of non-convex problems in applications. Different from previously used inexact strategies, IPAD method theoretically give rigid implementations of the stopping criteria for inner algorithms, which is more rigid and robust in practice. In the existing literature, there are a few papers on designing and analyzing inexact algorithms [28, 20] for convex optimization problems with linear constraints. However, no literature has shown the convergence analyses on using inexact skills under non-convex framework. As far as we know, we are the first to fuse various numerical methods into a general algorithm framework and at the same time give rigorous convergence analyses for non-convex and non-smooth optimization problems. Here, we list the main contributions in this paper as follows:

1. We propose an algorithm framework named IPAD for solving various complex NNMF problems (1). Our IPAD is not a single algorithm for solving NNMF problems, but a general algorithm framework that can fuse various efficient numerical methods into.

2. Our IPAD method is proved to achieve the best convergence result in general non-convex and non-smooth problems. Moreover, the stopping criterion of ours is theoretically analyzed and is more valid than the commonly used one. Since the convergence of IPAD is established on general optimization problem, thus it can be applied to more applications beyond NNMF.

3. We apply IPAD to a specific problem, i.e., SDL problem with \( \ell_0 \) penalty, on both synthetic and real-world data. The experimental results verify the efficiency of IPAD for solving the NNMF problem (1).

2. Inexact Proximal Alternating Direction

In this section, we present the detailed implementation of our proposed IPAD method. To simplify the subsequent derivations, we prefer proposing IPAD method and analyzing its properties on a general and concise formulation:

\[
\min_{x, y} \psi(z) := f(x) + g(y) + H(x, y),
\]

where the function \( L(I - AB) \) in (1) is a coupling case corresponding to the function \( H(x, y) \) in (2); the regularization \( r_1(A) \) and an indicator function\(^1\) \( \chi_S(A) \) compose \( f(x) \) in (2); similarly, \( g(Y) \) is equal to \( r_2(B) + \chi_S(B) \). Moreover, we give some assumptions on the objective function in (2):

1. \( f \) and \( g \) are proper, lower semi-continuous functions;
2. \( H \) is a \( C^1 \) function and its gradient is Lipschitz continuous on a bounded set;
3. \( \psi \) is a coercive, Kurdyka-Łojasiewicz (KL) function\(^2\).

\[\text{Remark 1} \quad \text{It should be mentioned that the most frequently used } L(A, B) = \|I - AB\|^2 \text{ is a KL function which also satisfies the second assumption. On the other hand, regularizers like } \ell_0 \text{ penalty, } \ell_p \text{ norm, SCAD [14], MCP [35] and indicator functions are all KL functions and at the same time satisfy the first assumption. Since the finite sums of KL functions are also KL functions [6], thus not a few models in NNMF problems satisfy these three assumptions.}\]

2.1. Algorithm Implementation

We have the observation that it is either impossible or extremely hard for calculating the exact solutions of the subproblems in NNMF, but they can be easily solved by other efficient numerical methods. This perception motivates us to propose an algorithm to solve the subproblems approximately and then obtain the inexact solutions \( x^{t+1} \) and \( y^{t+1} \):

\[\chi_S(x) = 1 \text{ if } x \in S; \text{ otherwise } \chi_S(x) = \infty.\]

\[\text{To be self-contained, the definition of KL function is given in the supplemental material due to space limit.}\]
by some numerical methods. Specifically, our IPAD method employs an alternative technique, which updates the variables $x^{t+1}$ and $y^{t+1}$ in a cyclic order:

$$
x^{t+1} \approx \arg \min_{x} f(x) + H(x, y^{t}) + \frac{\eta_1}{2} \|x - x^{t}\|^2,
$$

$$
y^{t+1} \approx \arg \min_{y} g(y) + H(x^{t+1}, y) + \frac{\eta_2}{2} \|y - y^{t}\|^2,
$$

where $\eta_1$ and $\eta_2$ are two proximal parameters added on each subproblem respectively.

Since in most cases, it is extremely hard to obtain the exact solutions of the two subproblems in NNF problems, thus various iterative methods can be employed to computing inexact solutions, described as the approximately equals in the Eq. (3). Let’s take an example here: when facing the well-known K-SVD technique [13, 26], i.e. a SDL problem with $\ell_1$ regularizer, both the two subproblems in Eq. (3) are convex. Then, various efficient numerical methods designed under convex framework like homotopy method [12], FISTA [4] and ADMM [7, 18] can be applied to solving these two subproblems.

After calculating the inexact solutions $x^t$ and $y^t$ of (3) by some numerical methods, we first compute intermediate variables $\tilde{x}^t$ and $\tilde{y}^t$ as the solutions of two specific proximal mappings\(^3\), which are assumed to be easily-solved:

$$
\tilde{x}^t = \text{prox}_{\frac{\eta_1}{2}}(v^t_x), \quad \tilde{y}^t = \text{prox}_{\frac{\eta_2}{2}}(v^t_y),
$$

where variables $v^t_x$ and $v^t_y$ in Eq. (4) are obtained by computing the following equalities:

$$
v^t_x = x^t - \nabla x H(x^t, y^t) - \eta_1^{-1}(x^t - x^{t-1}),
$$

$$
v^t_y = y^t - \nabla y H(x^t, y^t) - \eta_2^{-1}(y^t - y^{t-1}).
$$

After getting the intermediate variables $\tilde{x}^t$ and $\tilde{y}^t$ from $x^t$ and $y^t$, we compute two “errors” $e^t_x$ and $e^t_y$ for measuring the differences between $x^t$ and $\tilde{x}^t$, $y^t$ and $\tilde{y}^t$, respectively. Specifically, the well-designed “errors”, $e^t_x$ and $e^t_y$ are calculated in a special way:

$$
e^t_x = (1 - \eta_1^{-1})(x^t - \tilde{x}^t) + \nabla x H(x^t, y^t) - \nabla x H(\tilde{x}^t, y^t),
$$

$$
e^t_y = (1 - \eta_2^{-1})(y^t - \tilde{y}^t) + \nabla y H(x^t, y^t) - \nabla y H(x^t, \tilde{y}^t).
$$

Then, these two “errors” are used for checking whether the stopping criteria (7) are satisfied.

**Criterion 2** The two “errors”, $e^t_x$ and $e^t_y$, calculated by Eq. (6) must satisfy the following conditions:

$$
\|e^t_x\| \leq C_x \|x^t - x^{t-1}\|, \quad \|e^t_y\| \leq C_y \|y^t - y^{t-1}\|,
$$

where parameters $C_x$ and $C_y$ are two positive integers defined before the iteration starts.

The two parameters $C_x$ and $C_y$ in Eq. (7) have restricted relationships with the proximal parameters $\eta_1$ and $\eta_2$, which will be claimed later in Assumption 11. Then, once the criteria (7) are satisfied, we re-assign $\tilde{x}^t$ to $x^t$ and $\tilde{y}^t$ to $y^t$. Then $\tilde{x}^t$ and $\tilde{y}^t$ become the final solutions of their corresponding subproblems at $t$-th step. For clarity, we give the main steps of IPAD in Alg. 1.

**Algorithm 1** IPAD for solving problem (2).

1. Setting parameters: $C_x, C_y, \{\eta^1_1\}_{t \in \mathbb{N}}, \{\eta^2_2\}_{t \in \mathbb{N}}$.
2. Initializing variables: $x^0, y^0$.
3. while Not Converged do
4. \hspace{1em} while $\|e^t_x\| > C_x \|x^t - x^{t-1}\|$ do
5. \hspace{2em} Compute inexact $x^t$ by some numerical methods.
6. \hspace{2em} Calculate intermediate variable $\tilde{x}^t$ by Eq. (4).
7. \hspace{2em} Compute error $e^t_x$ by Eq. (6).
8. \hspace{1em} end while
9. \hspace{1em} Re-assign $x^t = \tilde{x}^t$.
10. \hspace{1em} while $\|e^t_y\| > C_y \|y^t - y^{t-1}\|$ do
11. \hspace{2em} Compute inexact $y^t$ by some numerical methods.
12. \hspace{2em} Calculate intermediate variable $\tilde{y}^t$ by Eq. (4).
13. \hspace{2em} Compute error $e^t_y$ by Eq. (6).
14. \hspace{1em} end while
15. \hspace{1em} Re-assign $y^t = \tilde{y}^t$.
16. end while

From the descriptions of the algorithm implementation, our IPAD method seems to be an unusual and complicated algorithm. However, the essence of IPAD can be summed up in a few words. Firstly, IPAD is an alternating method which updates each variable in a cyclic order. Secondly, at each step, the inner algorithms employ iterative processes to compute inexact solutions. In addition, the inner algorithms stop when reaching the special stopping criteria (7).

2.2. Discussions: why inexactness is better?

Solving subproblems inexactly is a common use in practical literatures. For example, the authors in [32] solve a linear subproblem by 2-step Gauss-Seidel iterations and receive good-enough solutions under the suggestions of [8]. Moreover, the authors of a lately NNF-related paper [18] stop all the inner iterations at fixed steps and achieve acceptable approximations of the exact solutions.

The great success of using inexact strategies in practice have attracted our attentions on proposing an efficient inexact algorithm framework for solving complex NNF problems. However, the previously used implementations and the corresponding stopping criteria usually lack theoretical support. Almost all the stopping criteria are designed from experience, e.g., a 2-step Gauss-Seidel is “good-enough” in [32]. Different from the previously naive criteria used in applications, our implementation of the inexactness and the corresponding stopping criteria are designed in theory.
Proposition 3 The “errors”, \( e_x^t \) and \( e_y^t \) calculated under our suggestions exactly describe the inexactness of (3).

To make the discussions in this section be more readable, we give the detailed proof of the Proposition 3 at the end of this paper in Appendix A.

The above Proposition 3 implies the following equalities:

\[
\begin{align*}
e_x^t &= g_x^t + \nabla_x H(x^t, y^{t-1}) + \eta_1^{t-1}(x^t - x^{t-1}), \\
e_y^t &= g_y^t + \nabla_y H(x^t, y^t) + \eta_2^{t-1}(y^t - y^{t-1}).
\end{align*}
\]

(8)

where \( g_x^{t+1} \in \partial f(x^{t+1}) \) and \( g_y^{t+1} \in \partial g(y^{t+1}) \).4 The above equalities are exactly the first-order optimality conditions of Eq. (3) by regarding \( e_x^t \) and \( e_y^t \) as the inexactness. We want to mention that \( e_x^t \) and \( e_y^t \) can not be directly calculated by Eq. (8) since \( g_x^t \) and \( g_y^t \) are unattainable in practice. Thus, our special implementation is applicable and at the same time is more rigid than previously used inexact strategies.

On the other hand, except for IPAD, there are a few literatures on designing converged algorithms for non-convex problems. Another popular converged algorithm is proximal alternating linearized method (PALM) [6], which is established with the global convergence property, the whole sequence converges to a critical point of the problem, which as far as we know, is so far the best result for non-convex and non-smooth optimization problems. Since PALM ensures explicit solutions of the subproblems by linearizing the coupling part \( H(x, y) \), thus it attracts much more attentions in applications and can also be applied for solving NNMF:

\[
\begin{align*}
x_{x}^{t+1} &\in \arg\min_x f(x) + \frac{\gamma_1}{2} \|x - u_x^t\|^2, \\
y_{y}^{t+1} &\in \arg\min_y g(y) + \frac{\gamma_2}{2} \|y - u_y^t\|^2,
\end{align*}
\]

(9)

where \( \gamma_1 > L_1^x \), \( \gamma_2 > L_2^y \) with Lipschitz constants \( L_1^x \) and \( L_2^y \) of \( \nabla_x H(x, y) \) and \( \nabla_y H(x, y) \), respectively; \( u_x^t = x_t - \nabla_x H(x^t, y^t) / \gamma_1 \) and \( u_y^t = y_t - \nabla_y H(x^t, y^t) / \gamma_2 \). However, the pursuit of closed-form solutions of the sub-problems does not ensure the efficiency of the whole algorithm (see Section 4.2). At the same time, for NNMF problems, it requires computing the exact Lipschitz constants at every iteration, which is not an easy work in practice. An alternative way is to estimate as small upper bound [3] as possible to speed up the convergence [34], however, estimating a tight upper bound is sometimes extremely hard and time-consuming. Furthermore, we can see from the experiments (in Section 4) that our inexact algorithm framework IPAD is more flexible and efficient than the PALM.

The notation “\( \partial \)” represents the limiting-subdifferential for proper, lower semi-continuous functions [27].

To distinguish from IPAD method, we add subscripts “\( \dagger \)” for PALM.

2.3. Discussions: a more flexible framework

Though the main procedure of our IPAD method is the same with many previously inexact techniques [18], our IPAD method (3) does not restrict specific formulae for solving the two subproblems of \( x^t \) and \( y^t \). Which means, one can apply totally different numerical methods for solving different subproblems. This flexibility of our IPAD method is especially welcomed in practice: since the constraints added on the two factorized components are always quite different from each other, efficient algorithms for solving specific subproblems should be carefully chosen.

Furthermore, no matter what it is, arbitrarily applying two inner algorithms still ensure the global convergence property of the whole algorithm as long as the stopping criteria (7) and the parameter conditions (11) are satisfied. From this perspective, our IPAD method is not a single algorithm for solving NNMF; it is an algorithm framework that can fuse other efficient numerical methods into. In addition, through blending the algorithms into our IPAD framework, some previous inexact methods applied to NNMF applications [25, 18] can also be proved to achieve the global convergence property.

3. Convergence Analyses of IPAD

In this section, we provide the theoretical support for IPAD method6. The strategic point on analyzing the convergence properties of IPAD is regarding \( x_{x}^{t+1} \) and \( y_{y}^{t+1} \) as the exact solutions on solving the following subproblems:

\[
\begin{align*}
\min_x f(x) + H(x, y^t) + \frac{\eta_1}{2} \|x - x^t\|^2 - (e_x^{t+1})^T x, \\
\min_y g(y) + H(x^{t+1}, y) + \frac{\eta_2}{2} \|y - y^t\|^2 - (e_y^{t+1})^T y.
\end{align*}
\]

(10)

This equivalent conversion is rigid since the first-order optimality conditions of (10) are exactly the same with Eq. (8). However, it should be emphasized that \( x_{x}^{t+1} \) and \( y_{y}^{t+1} \) are not computed by directly minimizing (10); this equivalent conversion is nothing but assisting in theoretical analyses.

Before proposing the key lemma for the main theorem, we give the requirements on \( \{\eta_1^t\}_{t \in \mathbb{N}} \) and \( \{\eta_2^t\}_{t \in \mathbb{N}} \).

Assumption 4 The proximal parameters \( \{\eta_1^t\}_{t \in \mathbb{N}} \) and \( \{\eta_2^t\}_{t \in \mathbb{N}} \) should satisfy

\[
\eta_1^t > 2C_x, \quad \eta_2^t > 2C_y, \quad \text{for all } t \in \mathbb{N},
\]

(11)

to ensure the whole IPAD algorithm converges.

Remark 5 Together with the Criterion 2 and the Eq. (3), we can see from the Assumption 4 that: if \( C_x \) and \( C_y \) are set too large, than it relatively reduce the inner iterations

---

6 Due to space limit, all the related proofs in this section will be detailedly given in the supplemental material.
but at the same time converges with a tiny step size. On the other side, if $C_x$ and $C_y$ are set too small, than it converges with relatively large step size but will increase the inner iterations. Thus, these parameters are balances that should be chosen carefully in practice.

Then with the help of the above assumption (11), we can obtain the main theorem in Theorem 6: our proposed IPAD has the best convergence property, that is, the global convergence property in general non-convex problems.

**Theorem 6** Under the Assumption 11 and suppose the sequence \( \{(x^t, y^t)\}_{t \in \mathbb{N}} \) generated by IPAD is bounded. Then the following two assertions are claimed in Theorem 6.

1. Suppose that the sequence \( \{(x^t, y^t)\}_{t \in \mathbb{N}} \) is a Cauchy sequence that converges to a critical point \((x^*, y^*)\) of \( \Psi \).

2. The second assertion ensures that there is a subsequence \( (12) \) in Lemma 7) during iterations, which together with \( \Psi \) is sufficiently descent (Eq. (12) in Lemma 7) during iterations, which together with the second assertion ensures that there is a subsequence of \( \{z^t\}_{t \in \mathbb{N}} \) that converges to a critical point of the problem. By further combining the KL property, we have the global convergence property for our proposed IPAD method as claimed in Theorem 6.

**Lemma 7** Suppose that the sequence \( \{(x^t, y^t)\}_{t \in \mathbb{N}} \) generated by IPAD is bounded. Then the following two assertions hold under the Assumption 11:

\[
\Psi(x^t) - \Psi(z_0) \geq a(\|x^t - x^t-1\|_2 + \|y^t - y^t-1\|_2), \quad (12)
\]

\[
dist(0, \partial \Psi(z^t)) \leq b(\|x^t - x^t-1\|_2 + \|y^t - y^t-1\|_2), \quad (13)
\]

where constants \( a = \min_{t \in \mathbb{N}} \{\frac{\eta_1}{4} - (C_x)^2, \frac{\eta_2}{4} - (C_y)^2\} \) and \( b = \max_{t \in \mathbb{N}} \{\eta_1 - C_x, M + \frac{\eta_2}{4} + C_y\} \) with Lipschitz constant \( M \) of \( \nabla H \) on bounded set.

For the convergence rate, our IPAD method shares the same result with PALM when the desingularising function \( \phi(s) = -\frac{\mu}{2} s^2 \) : \([0, \mu) \rightarrow \mathbb{R}_+ \) of \( \Psi \) is satisfied with positive constant \( C \) and \( \theta \in [0, 1) \). Specifically, IPAD converges in a finite number of steps when \( \theta = 1 \). For \( \theta \in [0, \frac{1}{2}) \) and \( \theta \in (\frac{1}{2}, 1) \), IPAD converges with a sublinear rate and a linear rate respectively. Though the convergence rate of IPAD is the same with PALM in theory (the convergence rate is not affected by the algorithm but the objective function \( \Psi \) of the KŁ function in supplemental material.}

**Remark 8** Our IPAD method can be applied to a general multi-block problem:

\[
\min_{x_1, \ldots, x_K} \sum_{i=1}^K f_i(x_i) + H(x_1, \ldots, x_K). \quad (14)
\]

All the convergence properties conducted on problem 2 can be straightforwardly extended to this multi-block case.

Since not a few problems in signal and image processing, machine learning and pattern recognition can be formulated or reformulated as this general formulation, thus our IPAD method can be applied for solving a wider range of applications beyond NNMF.

### 4. Experiments

We consider a widely-concerned SDL problem with \( \ell_0 \) penalty, which is formulated as:

\[
\min_{D, W} \frac{1}{2} \|I - DW^T\|^2 + \lambda \|W\|_0 + \lambda_D(D) + \lambda_W(W), \quad (15)
\]

where \( \|\cdot\|_0 \) denotes the \( \ell_0 \) penalty that counts the number of non-zero elements of \( W \). The indicator function \( \lambda \) acts on set \( D = \{D = \{d_i\}_{i=1}^m \subset \mathbb{R}^{n \times m} : \|d_i\|_1 = 1, \forall i\} \) set \( W \) is empty for synthetic data whereas \( W = \{W = \{w_i\}_{i=1}^m \subset \mathbb{R}^{m \times n} : \|w_i\|_{\infty} \leq U_0, \forall i\} \) is defined for real-world data to enhance the stability of the model \( [2] \). Moreover, we denote \( S(W) = \lambda \|W\|_0 + \lambda_D(D) + \lambda_W(W) \) to simplify the deduction.

It is observed that problem (15) is a special case of problem (2). Thus, PALM can be applied to solve (15) by computing Lipschitz constants at every iteration. Moreover, since it is extremely hard to get exact solutions of the subproblems of (15), our IPAD inexactly solves the following problems to get \( D^{t+1} \) and \( W^{t+1} \):

\[
\min_{W} S(W) + \frac{1}{2} \|I - D^tW^T\|^2 + \frac{\eta_1^t}{2} \|W - W^t\|^2, \quad (16)
\]

\[
\min_{D} \lambda_D(D) + \frac{1}{2} \|I - D(W^{t+1})^T\|^2 + \frac{\eta_2^t}{2} \|D - D^t\|^2. \quad (17)
\]

As far as we know, few efficient numerical methods are designed for solving the non-convex subproblem (16). We apply a proximal iterative hard-thresholding (PITH) algorithm \([1, 17]\) to solve this subproblem. On the other hand, for subproblem (17), we apply ADMM \([7]\) for solving it. All the algorithms compared in this paper are implemented by Matlab R2013b and are tested on a PC with 8 GB of RAM and Intel Core i5-4200M CPU.

#### 4.1. Synthetic Data

We generate synthetic data with different sizes to help analyze the property of IPAD (see Tab. 1). All the algorithms for the synthetic data stop when satisfying:

\[
\max\{\|D^{t+1} - D^t\|_2, \|W^{t+1} - W^t\|_2, \|\Psi^{t+1} - \Psi^t\|_2\} < 1e^{-4}, \quad (18)
\]

where \( \Psi^t \) is the objective value at step \( t \).
Table 1. The number of outer iterations and the iteration time (s) of PALM \[6\], INV \[3\], IPAD-PITH (PITH for short in this table), IPAD-ADMM (ADMM for short) and IPAD-P2A (P2A for short) for SDL problem with $l_0$ penalty on synthetic data. The convergence results in the first row belongs to $n = 64$; the second row belongs to $n = 144$ and the last row belongs to $n = 256$.

| Data   | $n = 64, m = 600, p = 4000$ | $n = 144, m = 900, p = 10000$ | $n = 256, m = 1600, p = 16000$ |
|--------|-----------------------------|-----------------------------|-----------------------------|
| Alg.   | PALM | INV | PITH | ADMM | P2A | PALM | INV | ADMM | P2A | PALM | INV | ADMM | P2A |
| Out-iter | 104 | 23 | 51 | 22 | 14 | 56 | 33 | 22 | 15 | 31 | 38 | 18 | 12 |
| Time(s) | 52.82 | **7.81** | 253.56 | 8.72 | 8.31 | 96.08 | 45.93 | **35.58** | 38.94 | 319.97 | 253.17 | **150.25** | 158.42 |

4.1.1 Efficiency of Inexact Strategy

To show the respective effects of using inexact strategies on different subproblems, we propose IPAD-PITH which obtains $W^{t+1}$ by PITH but keeps $D$-subproblem the same as PALM\textsuperscript{8}. We also design IPAD-ADMM that computes $D^{t+1}$ by ADMM but remains $W$-subproblem the same as PALM.

The comparisons in Tab. 1 among PALM, IPAD-PITH and IPAD-ADMM show that inexact strategies help reduce the iteration steps: both the IPAD-PITH and IPAD-ADMM converges with less iterations than PALM. However, the performances of IPAD-PITH and IPAD-ADMM are quite different in inner iterations. We can see from Fig. 1(d) that IPAD-ADMM uses few inner steps during iterations. However, IPAD-PITH reaches the maximum inner steps (set as 20) at almost every iteration. This from one side shows that ADMM is suitable for solving (17) but PITH is less efficient for solving (15). On the other side it is caused by the

\textsuperscript{8}Due to the space limit, detailed implementations of all the algorithms mentioned in Section 4 are given in supplemental material.
| Image / σ / λ  | PALM PSNR Iter Time | mPALM PSNR Iter Time | INV PSNR Iter Time | IPAD-ADMM PSNR Iter Time |
|----------------|---------------------|----------------------|-------------------|-------------------------|
| Peppers512 / 30 / 5500 | 28.64 4 4.11 | 30.11 10 325.33 | 30.14 61 48.83 | 30.21 25 27.29 |
| Lena512 / 25 / 4500 | 28.85 4 3.98 | 31.04 14 459.10 | 31.11 62 50.20 | 31.13 31 37.78 |
| Barbara512 / 20 / 3500 | 28.73 4 4.13 | 30.06 11 399.19 | 30.09 58 45.09 | 30.22 18 16.54 |
| Hill512 / 15 / 2500 | 29.84 4 4.03 | 31.20 26 85.34 | 31.31 84 69.73 | 31.32 19 17.81 |

Table 2. The PSNR values of the recovered images, number of outer iterations and the whole iteration time (s) of PALM [6], mPALM [2], INV [3] and IPAD-ADMM for real-world data.

By comparing the algorithms, all the inexact strategies of our IPAD method perform better than PALM and are verified to be practicable, converged and efficient. However, a less efficient numerical method for solving subproblem indeed reduce the efficiency of the whole algorithm. Therefore, we should carefully choose effective numerical methods for solving subproblems.

4.1.2 Other Comparisons

At last, we compare INV that solves \( W^{t+1} \) in the same way as PALM but treats \( D^{t+1} \) as the solution of a linear system first and then project the solution on set \( D \). Though this strategy seems to be efficient in practice [3], it lacks theoretical guarantee. Firstly, \( D^{t+1} \) calculated by INV is not an exact solution of \( (17) \). Secondly, it is computed without measure the inexactness. So applying INV sometimes creates oscillations during iterations (Fig. 1(f)) and the performances of INV are unstable especially in real-world applications (see the experimental results in Section 4.2). Thus we do not recommend using it in practice.

In the end, we compare the computational cost of PALM, IPAD-ADMM and INV at every iteration. Since these three algorithms share the same updates of \( W^{t+1} \), the only difference of the computational cost lies in the update of \( D^{t+1} \). We list the computational cost of updating \( D^{t+1} \) for only once. The number of the dominant operations of PALM is \( O(m^3 + mn + mp) \), Calculating \( D^{t+1} \) by IPAD-ADMM will cost \( O(m^3n + m^2p + mn + mp) \). The last algorithm, INV has \( O(m^3n + m^2p + mn + mp) \) calculations.

Though INV seems to take more operations than PALM, the experimental results in Tab. 1 do not match this expectation: INV converges with more steps but uses less time. We should mention that the calculations \( O(m^3n) \) comes from using Gauss-Elimination to obtain an inverse of matrix. However, the inversion process in Matlab may employ more efficient methods. This may be the reason for causing the contradiction between theory and practice.

4.2 Real-world Data

We apply IPAD to real-world data on image denoising problem [13, 10]. We only compare PALM, INV and IPAD-ADMM in this section and all the algorithms terminates when reaching \( ||D^{t+1} - D^t||_2/||D^t|| < 10^{-2} \). The patches in each image, of size \( 8 \times 8 \), are regularly sampled in an overlapping manner. The noisy images are obtained by adding Gaussian randomly noises with level \( \sigma = 15, 20, 25, 30 \), see Tab. 2. We apply IPAD to 7 images and select some experimental results in Fig. 2-4.

---

9See supplemental material for detailed discussions.

10See the proof in supplemental material.

11See supplemental material for detailed proof.

12See supplemental material for detailed analyses.

13See supplemental material for more results.
code for [2] is not provided so that we cannot tell the reasons for causing this inconsistency. But we admit that the computational time of all the compared algorithms will be reduced by optimizing them carefully. Though the iterative time of mPALM does not match with the results in [2], we can see from the Tab. 2 that our IPAD-ADMM uses comparative time with the results posted in [2] but at the same time better recovers the noisy images.

On the other hand, the vibration of INV causes more iteration steps than IPAD-ADMM for converging. Thus our carefully designed algorithm, IPAD-ADMM performs more stable than the commonly used INV and is efficient for the real-world applications.

5. Conclusion and Future Work

In this paper, we propose a general and flexible algorithm framework named IPAD for solving NNMF problems, which achieves the best convergence result for non-convex and non-smooth problems. Different from previously used inexact strategies, IPAD method gives rigid implementations of the stopping criteria, which is more robust than the previously used ones in practice. Moreover, we verify from the experimental results that IPAD is efficient and effective for solving a SDL problem with $\ell_0$ penalty.

Since our IPAD method can be straightforwardly extended to a general multi-block non-convex and non-smooth problem, it can be applied to a wider range of applications beyond NNMF, like discriminative learning problems, optimizations on manifolds and tensor decompositions. Moreover, the iterative process of IPAD can be more improved: a random version of IPAD is welcomed to avoid local optimal; a hybrid form of IPAD is more flexible in practice just as claimed in the last section. At the moment, considering the generality and flexibility of IPAD method, we are focusing on discovering its connections with some frameworks in neural network; and then try to optimize those network structures to obtain more robust results.

As shown in Tab. 2, PALM seems to converge quickly but get bad converged results. However, the truth is: the large upper bound $\| (D^{t+1})^\top D^{t+1} \|_2$ of the Lipschitz constant emphasizes the function of the proximal term. So it causes tiny differences between $D^{t+1}$ and $D^t$. Thus, PALM does not converge when reaching the stopping criterion; on the contrary, it converges quite slow.

For the failure of using PALM, we adopt the strategy used in [2], which regards the problem (15) as a $m + 1$-block problem: solving $\{d_i\}_{i=1}^m$ separately by PALM. We name this algorithm as mPALM and show the results in Tab. 2. This time, mPALM does converge when reaching the stopping criterion. However, mPALM seems to use more time for one iteration (see Tab. 2). We must emphasize that all the algorithms are implemented by Matlab without optimization. We code mPALM in a normal way but the iteration time seems to be inconsistent to [2]. Since the detailed
The above deductions can be similarly extended to the case of $y^t$. Together with Eq. (5), we have
\[ x^t = \text{prox}_{\mathcal{H}}(v^t_x + e^t_x), \quad y^t = \text{prox}_{\mathcal{G}}(v^t_y + e^t_y). \] (20)
From the definition of prox mapping, Eq. (20) is equal to
\[ e^t_x = g^t_x + \nabla_x H(x^t, y^{t-1}) + \eta_{1}^{-1}(x^t - x^{t-1}), \]
\[ e^t_y = g^t_y + \nabla_y H(x^t, y^t) + \eta_{2}^{-1}(y^t - y^{t-1}). \] (21)
where $g^{t+1}_x \in \partial f(x^{t+1})$ and $g^{t+1}_y \in \partial g(y^{t+1})$. The above equalities are exactly the first-order optimality conditions of (3) by regarding $e^t_x$ and $e^t_y$ as the inexactness. 

References

[1] F. Bach, R. Jenatton, J. Mairal, and G. Obozinski. Optimization with sparsity-inducing penalties. Foundations & Trends® in Machine Learning, 4(1):1-106, 2011. 5

[2] C. Bao, H. Ji, Y. Quan, and Z. Shen. $\ell_0$ norm based dictionary learning by proximal methods with global convergence. In CVPR, 2014. 1, 5, 7, 8

[3] C. Bao, H. Ji, Y. Quan, and Z. Shen. Dictionary learning for sparse coding: Algorithms and convergence analysis. IEEE TPAMI, 38(7):1356-1369, 2016. 4, 6, 7, 8

[4] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. Siam Journal on Imaging Sciences, 2(1):183-202, 2009. 3

[5] E. Benetos and C. Kotropoulos. Non-negative tensor factorization applied to music genre classification. IEEE TASLP, 18(8):1955-1967, 2010. 1

[6] P. Frankel, G. Garrigos, and J. Peyrò. Splittings methods with variable metric for kurdyka–lojasiewicz functions and general convergence rates. Journal of Optimization Theory & Applications, 165(3):874–900, 2015. 5

[7] J. Gregor and Y. LeCun. Learning fast approximations of sparse coding. In Proc. International Conference on Machine Learning, 2010. 1

[8] K. Herrity, A. C. Gilbert, and J. A. Tropp. Sparse approximation via iterative thresholding. In ICASSP, 2006. 5

[9] K. Huang, N. D. Sidiropoulos, and A. P. Liavas. A flexible and efficient algorithmic framework for constrained matrix and tensor factorization. IEEE TSP, 64(19):5052–5065, 2015. 1, 2, 3, 4

[10] S. Jia and Y. Qian. Constrained nonnegative matrix factorization for hyperspectral unmixing. IEEE TGRS, 47(1):161–173, 2009. 1

[11] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. Psychometrika, 1(3):211–218, 1936. 1

[12] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. Mathematics, 32(2):2004, 2004. 3

[13] M. Elad and M. Aharon. Image denoising via sparse and redundant representations over learned dictionaries. IEEE TIP, 15(12):3736–3745, 2006. 3, 7

[14] J. Fan and R. Li. Variable selection via nonconcave penalized likelihood and its oracle properties. Journal of the American statistical Association, 96(456):1348–1360, 2001. 2

[15] P. Frankel, G. Garrigos, and J. Peyrò. Splittings methods with variable metric for kurdyka–lojasiewicz functions and general convergence rates. Journal of Optimization Theory & Applications, 165(3):874–900, 2015. 5
[26] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11(1):19–60, 2010. 3

[27] B. S. Mordukhovich. *Variational analysis and generalized differentiation I: Basic theory*, volume 330. Springer Science & Business Media, 2006. 4

[28] M. K. Ng, F. Wang, and X. Yuan. Inexact alternating direction methods for image recovery. *SIAM Journal on Scientific Computing*, 33(4):1643–1668, 2011. 2

[29] H. Shen and J. Z. Huang. Sparse principal component analysis via regularized low rank matrix approximation. *Journal of Multivariate Analysis*, 99(6):1015–1034, 2008. 1

[30] J. Shi, X. Ren, G. Dai, and J. Wang. A non-convex relaxation approach to sparse dictionary learning. In *CVPR*, 2011. 1, 2

[31] G. Tomasi and R. Bro. A comparison of algorithms for fitting the parafac model. *Computational Statistics & Data Analysis*, 50(7):1700–1734, 2006. 2

[32] Y. Wang, R. Liu, X. Song, and Z. Su. Saliency detection via nonlocal $\ell_0$ minimization. In *ACCV*, 2014. 3

[33] Y. X. Wang and Y. J. Zhang. Nonnegative matrix factorization: A comprehensive review. *IEEE TKDE*, 25(6):1336–1353, 2013. 1

[34] Y. Xu and W. Yin. A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion. *SIAM Journal on Imaging Sciences*, 6(3):1758–1789, 2013. 4

[35] C. Zhang. Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*, pages 894–942, 2010. 2

[36] W. S. Zheng, S. Z. Li, J. H. Lai, and S. Liao. On constrained sparse matrix factorization. In *ICCV*, 2007. 1

[37] H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of Computational & Graphical Statistics*, 2007:1–30, 2012. 1