JMLR: Workshop and Conference Proceedings 29:1–16, 2013 ACML 2013

Linearized Alternating Direction Method with Parallel Splitting and Adaptive Penalty for Separable Convex Programs in Machine Learning

Risheng Liu RSLIU@DLUT.EDU.CN
Faculty of Electronic Information and Electrical Engineering, Dalian University of Technology

Zhouchen Lin* ZLIN@PKU.EDU.CN
Key Lab. of Machine Perception (MOE), School of EECS, Peking University

Zhixun Su ZXSU@DLUT.EDU.CN
School of Mathematical Sciences, Dalian University of Technology

Editor: Cheng Soon Ong and Tu Bao Ho

Abstract

Many problems in statistics and machine learning (e.g., probabilistic graphical model, feature extraction, clustering and classification, etc) can be (re)formulated as linearly constrained separable convex programs. The traditional alternating direction method (ADM) or its linearized version (LADM) is for the two-variable case and *cannot* be naively generalized to solve the multi-variable case. In this paper, we propose LADM with parallel splitting and adaptive penalty (LADMPSAP) to solve multi-variable separable convex programs efficiently. When all the component objective functions have bounded subgradients, we obtain convergence results that are stronger than those of ADM and LADM, e.g., allowing the penalty parameter to be unbounded and proving the *sufficient and necessary conditions* for global convergence. We further propose a simple optimality measure and reveal the convergence *rate* of LADMPSAP in an ergodic sense. For programs with extra convex set constraints, we devise a practical version of LADMPSAP for faster convergence. LADMPSAP is particularly suitable for sparse representation and low-rank recovery problems because its subproblems have closed form solutions and the sparsity and low-rankness of the iterates can be preserved during the iteration. It is also *highly parallelizable* and hence fits for parallel or distributed computing. Numerical experiments testify to the speed and accuracy advantages of LADMPSAP.

Keywords: Convex Programs, Linearized Alternating Direction Method, Parallel Splitting, Adaptive Penalty, Subspace Clustering, Matrix Completion.

1. Introduction

Recently, convex programs have become increasingly popular for solving a variety of statistics and machine learning problems, ranging from theoretical modeling, e.g., latent variable graphical model selection (Chandrasekaran et al., 2012), low-rank feature extraction (i.e., matrix decomposition (Candès et al., 2011) and matrix completion (Candès and Recht, 2009)), subspace clustering (Liu et al., 2012) and kernel discriminant analysis (Ye et al., 2008), to real-world applications, e.g., face recognition (Wright et al., 2009), saliency detec-

* Corresponding author

tion (Shen and Wu, 2012) and video denoising (Ji et al., 2010). Specifically, these problems can all be (re)formulated as the following linearly constrained separable convex program[1]:

$$\min_{\mathbf{x}_1,\cdots,\mathbf{x}_n} \sum_{i=1}^{n} f_i(\mathbf{x}_i), \quad s.t. \quad \sum_{i=1}^{n} \mathcal{A}_i(\mathbf{x}_i) = \mathbf{b}, \tag{1}$$

where $\mathbf{x}_i$ and $\mathbf{b}$ could be either vectors or matrices[2], f_i is a closed proper convex function (e.g., nuclear norm $\|\cdot\|_*$ (Fazel, 2002), defined as the sum of singular values; ℓ_1 norm $\|\cdot\|_1$ (Candès et al., 2011), defined as the sum of absolute values of all entries; and Frobenius norm $\|\cdot\|$, etc), and $\mathcal{A}_i : \mathbb{R}^{d_i} \to \mathbb{R}^m$ is a linear mapping (e.g., the subsampling operator in matrix completion (Candès and Recht, 2009)). Without loss of generality, we may assume that none of the $\mathcal{A}_i$'s is a zero mapping, the solution to $\sum_{i=1}^{n} \mathcal{A}_i(\mathbf{x}_i) = \mathbf{b}$ is non-unique, and the mapping $\mathcal{A}(\mathbf{x}_1, \cdots, \mathbf{x}_n) \equiv \sum_{i=1}^{n} \mathcal{A}_i(\mathbf{x}_i)$ is onto[3].

Although general theories on convex programs are fairly complete nowadays, e.g., most of them can be solved by the interior point method (Boyd and Vandenberghe, 2004), when facing with large scale problems, which are typical in machine learning, the general theory may not lead to efficient enough algorithms. For example, when using CVX[4], an interior point based toolbox, to solve nuclear norm minimization problems (i.e., one of the f_i's is the nuclear norm of a matrix), such as matrix completion (Candès and Recht, 2009), robust principal component analysis (Candès et al., 2011), and low-rank representation (Liu et al., 2010, 2012), the complexity of each iteration is $O(q^6)$, where $q \times q$ is the matrix size. Such a complexity is unbearable for large scale problems.

To address the scalability issue, first order methods are often preferred. The accelerated proximal gradient (APG) algorithm (Beck and Teboulle, 2009; Toh and Yun, 2010) is popular due to its guaranteed $O(K^{-2})$ convergence rate, where K is the iteration number. However, APG is basically for unconstrained optimization. For constrained optimization, the constraints have to be added to the objective function as penalties, resulting in approximated solutions only. The alternating direction method (ADM) (Fortin and Glowinski, 1983; Boyd et al., 2010; Lin et al., 2009a) has regained a lot of attention recently and is also widely used. It is especially suitable for separable convex programs like (1) because it fully utilizes the separable structure of the objective function. Unlike APG, ADM can solve (1) exactly. Another first order method is the split Bregman method (Goldstein and Osher, 2008), which is closely related to ADM (Esser, 2009) and is very influential in image processing.

1. If the objective function is not separable or there are extra convex set constraints, $\mathbf{x}_i \in X_i$, $i = 1, \cdots, n$, the program can be transformed into (1) by introducing auxiliary variables, c.f. (28)-(30).
2. Here we call $\mathbf{x}_i$ a "vector variable" or a "matrix variable" because each $\mathbf{x}_i$ may consist of multiple scalar variables. However, in the following for simplicity we still call each $\mathbf{x}_i$ as one variable because all scalar variables therein are processed simultaneously in the same manner. Such a convention has been implicitly used in the literature (He and Yuan, 2013; Tao, 2011; Lin et al., 2011). We will also use bold capital letters if a variable is known to be a matrix.
3. These two assumptions are equivalent to that the matrix $\mathbf{A} \equiv (\mathbf{A}_1 \ \cdots \ \mathbf{A}_n)$ is not full column rank but full row rank, where $\mathbf{A}_i$ is the matrix representation of $\mathcal{A}_i$.
4. Available at http://stanford.edu/∼boyd/cvx

An important reason that first order methods are popular for solving large scale convex programs in machine learning is that the convex functions f_i's are often matrix or vector norms or characteristic functions of convex sets, which enables the following subproblems

$$\min_{\mathbf{x}_i} f_i(\mathbf{x}_i) + \frac{\sigma}{2}\|\mathbf{x}_i - \mathbf{w}\|^2 \tag{2}$$

to have closed form solutions. For example, when f_i is the ℓ_1 norm, the optimal solution is $\mathbf{x}_i^* = \mathcal{T}_{\sigma^{-1}}(\mathbf{w})$, where $\mathcal{T}_\varepsilon(x) = \mathrm{sgn}(x)\max(|x| - \varepsilon, 0)$ is the soft-thresholding operator (Goldstein and Osher, 2008); when f_i is the nuclear norm, the optimal solution is: $\mathbf{X}_i^* = \mathbf{U}\mathcal{T}_{\sigma^{-1}}(\mathbf{\Sigma})\mathbf{V}^T$, where $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ is the singular value decomposition (SVD) of $\mathbf{W}$ (Cai et al., 2010); and when f_i is the characteristic function of the nonnegative cone, the optimal solution is $\mathbf{x}_i^* = \max(\mathbf{w}, 0)$. Since subproblems like (2) have to be solved *in each iteration* when using first order methods to solve separable convex programs, that they have closed form solutions greatly facilitates the optimization.

However, when applying ADM to solve (1) with generic linear mappings (i.e., $\mathcal{A}_i$ is not the identity mapping), the resulting subproblems may not have closed form solutions (because $\|\mathbf{x}_i - \mathbf{w}\|^2$ in (2) becomes $\|\mathcal{A}_i(\mathbf{x}_i) - \mathbf{w}\|^2$), hence need to be solved iteratively, making the optimization process awkward. Some work (Yang and Yuan, 2013; Lin et al., 2011) has considered this issue by linearizing the quadratic term in the subproblems, hence such a variant of ADM is called the linearized ADM (LADM). Nonetheless, most of the existing theories on ADM and LADM are for the *two-variable* case, i.e., $n = 2$ in (1) (Fortin and Glowinski, 1983; Boyd et al., 2010; Lin et al., 2011). The number of variables is restricted to two because the proofs of convergence for the two-variable case are not applicable for the multi-variable case, i.e., $n > 2$ in (1). Actually, a naive generalization of ADM or LADM to the multi-variable case may diverge (see (11)). Unfortunately, in practice multi-variable convex programs often occur, e.g., robust principal component analysis with dense noise (Candès et al., 2011), latent low-rank representation (Liu and Yan, 2011), and when there are extra convex set constraints (see (28)-(29) and (37)). So it is desirable to design practical algorithms for the multi-variable case.

Only very recently He and Yuan (2013) and Tao (2011) considered the multi-variable LADM and ADM, respectively. To safeguard convergence, He and Yuan (2013) proposed LADM with Gaussian back substitution (LADMGB), which destroys the sparsity or low-rankness of the iterates during iterations when dealing with sparse representation and low-rank recovery problems, while Tao (2011) proposed ADM with parallel splitting, whose subproblems may not be easily solvable. Moreover, they all developed their theories with the penalty parameter being fixed, resulting in slow convergence and difficulty in tuning an optimal penalty parameter that fits for different data and data sizes.

To propose an algorithm that is more suitable for convex programs in machine learning, in this paper we aim at combining the advantages of He and Yuan (2013), Tao (2011) and Lin et al. (2011), i.e., combining LADM, parallel splitting, and adaptive penalty. Hence we call our method LADM with parallel splitting and adaptive penalty (LADMPSAP). With LADM, the subproblems will have forms like (2) and hence can have closed form solutions. With parallel splitting, the sparsity and low-rankness of iterates can be preserved during iteration when dealing with sparse representation and low-rank recovery problems, saving both the storage and the computation load. With adaptive penalty, the convergence can

be faster and it is unnecessary to tune an optimal penalty parameter. Parallel splitting also make the algorithm *highly parallelizable*, making LADMPSAP suitable for parallel or distributed computing, which is important for large scale machine learning. When all the component objective functions have bounded subgradients, we prove convergence results that are stronger than the existing theories on ADM and LADM. For example, the penalty parameter can be unbounded and the *sufficient and necessary* conditions of the global convergence of LADMPSAP can be obtained as well. We also propose a simple optimality measure and prove the convergence *rate* of LADMPSAP in an ergodic sense under this measure. Our proof is much simpler than those in (He and Yuan, 2011) and (Tao, 2011) which relied on a complex optimality measure. When a convex program has extra convex set constraints, we further devise a practical version of LADMPSAP that converges faster thanks to better parameter analysis. Experiments testify to the advantage of LADMPSAP in speed and accuracy.

Note that Goldfarb and Ma (2012) also proposed a multiple splitting algorithm for convex optimization. However, they only considered a *very special* case of our model problem (1), i.e., all the linear mappings $\mathcal{A}_i$'s are identity mappings[5]. With their simpler model problem, linearization is unnecessary and a faster convergence rate, $O(K^{-2})$, can be achieved. In contrast, in this paper we aim at proposing a practical algorithm for efficiently solving much more general problems like (1).

2. Review of LADMAP for the Two-Variable Case

We first review LADM with adaptive penalty (LADMAP) (Lin et al., 2011) for the two-variable case of (1). Specifically, this algorithm consists of four steps:

1. Update $\mathbf{x}_1$:
$$\mathbf{x}_1^{k+1} = \underset{\mathbf{x}_1}{\operatorname{argmin}} f_1(\mathbf{x}_1) + \frac{\sigma_1^{(k)}}{2}\left\|\mathbf{x}_1 - \mathbf{u}_1^k\right\|^2, \tag{3}$$

2. Update $\mathbf{x}_2$:
$$\mathbf{x}_2^{k+1} = \underset{\mathbf{x}_2}{\operatorname{argmin}} f_2(\mathbf{x}_2) + \frac{\sigma_2^{(k)}}{2}\left\|\mathbf{x}_2 - \mathbf{u}_2^k\right\|^2, \tag{4}$$

3. Update λ:
$$\lambda^{k+1} = \lambda^k + \beta_k\left(\sum_{i=1}^{2}\mathcal{A}_i(\mathbf{x}_i^{k+1}) - \mathbf{b}\right), \tag{5}$$

4. Update β:
$$\beta_{k+1} = \min(\beta_{\max}, \rho\beta_k), \tag{6}$$

where λ is the Lagrange multiplier, β_k is the penalty parameter, $\sigma_i^{(k)} = \eta_i\beta_k$ with $\eta_i > \|\mathcal{A}_i\|^2$,

$$\mathbf{u}_i^k = \mathbf{x}_i^k - \mathcal{A}_i^*(\tilde{\lambda}_i^k)/\sigma_i^{(k)}, \quad i = 1, 2, \tag{7}$$

5. The multi-variable problems introduced in (Boyd et al., 2010) also fall within this category.

in which $\mathcal{A}_i^*$ is the adjoint operator of $\mathcal{A}_i$,

$$\tilde{\lambda}_1^k = \lambda^k + \beta_k(\mathcal{A}_1(\mathbf{x}_1^k) + \mathcal{A}_2(\mathbf{x}_2^k) - \mathbf{b}), \tag{8}$$
$$\tilde{\lambda}_2^k = \lambda^k + \beta_k(\mathcal{A}_1(\mathbf{x}_1^{k+1}) + \mathcal{A}_2(\mathbf{x}_2^k) - \mathbf{b}), \tag{9}$$

and ρ is an adaptively updated parameter. Please refer to (Lin et al., 2011) for details.

3. LADMPSAP for Multi-Variable Case

In this section, we provide a new LADM-based algorithm for the general multi-variable separable convex program (1). The *sufficient and necessary conditions* for global convergence and the convergence *rate* of this algorithm are discussed.

3.1. LADM with Parallel Splitting and Adaptive Penalty

Contrary to our intuition, the multi-variable case is actually *fundamentally different* from the two-variable one. For the multi-variable case, it is very natural to generalize LADMAP for the two-variable case in a straightforward way, with

$$\tilde{\lambda}_i^k = \lambda^k + \beta_k \left(\sum_{j=1}^{i-1} \mathcal{A}_j(\mathbf{x}_j^{k+1}) + \sum_{j=i}^{n} \mathcal{A}_j(\mathbf{x}_j^k) - \mathbf{b} \right), \ i = 1, \cdots, n. \tag{10}$$

Unfortunately, we were unable to prove the convergence of such a naive LADMAP using the same proof for the two-variable case. This is because their Fejér monotone inequalities (see Remark 2) cannot be the same. That is why He et al. has to introduce an extra Gaussian back substitution (He et al., 2012; He and Yuan, 2013). Actually, the above naive generalization of LADMAP may be divergent (which is even worse than converging to a wrong solution), e.g., when applied to the following problem with $n \geq 5$:

$$\min_{\mathbf{x}_1, \cdots, \mathbf{x}_n} \sum_{i=1}^{n} \|\mathbf{x}_i\|_1, \quad s.t. \quad \sum_{i=1}^{n} \mathbf{A}_i \mathbf{x}_i = \mathbf{b}. \tag{11}$$

Fortunately, by modifying $\tilde{\lambda}_i^k$ slightly we are able to prove the convergence of the corresponding algorithm. More specifically, our algorithm for solving (1) consists of the following steps:

1. Update $\mathbf{x}_i$'s in parallel:

$$\mathbf{x}_i^{k+1} = \underset{\mathbf{x}_i}{\operatorname{argmin}} f_i(\mathbf{x}_i) + \frac{\sigma_i^{(k)}}{2} \left\| \mathbf{x}_i - \mathbf{u}_i^k \right\|^2, \quad i = 1, \cdots, n, \tag{12}$$

2. Update λ:

$$\lambda^{k+1} = \lambda^k + \beta_k \left(\sum_{i=1}^{n} \mathcal{A}_i(\mathbf{x}_i^{k+1}) - \mathbf{b} \right), \tag{13}$$

3. Update β:

$$\beta_{k+1} = \min(\beta_{\max}, \rho\beta_k), \tag{14}$$

where $\sigma_i^{(k)} = \eta_i\beta_k$,

$$\mathbf{u}_i^k = \mathbf{x}_i^k - \mathcal{A}_i^*(\hat{\lambda}^k)/\sigma_i^{(k)}, \tag{15}$$

in which $\mathcal{A}_i^*$ is the adjoint operator of $\mathcal{A}_i$,

$$\hat{\lambda}^k = \lambda^k + \beta_k\left(\sum_{i=1}^n \mathcal{A}_i(\mathbf{x}_i^k) - \mathbf{b}\right), \tag{16}$$

and

$$\rho = \begin{cases} \rho_0, & \text{if } \max\left(\left\{\sqrt{\beta_k\sigma_i^{(k)}}\left\|\mathbf{x}_i^{k+1} - \mathbf{x}_i^k\right\|, i = 1, \cdots, n\right\}\right)/\|\mathbf{b}\| < \varepsilon_2, \\ 1, & \text{otherwise,} \end{cases} \tag{17}$$

with $\rho_0 > 1$ being a constant and $0 < \varepsilon_2 \ll 1$ being a threshold. Indeed, we replace $\tilde{\lambda}_i^k$ with $\hat{\lambda}^k$ as (16), which is *independent* of i, and the rest procedures of the algorithm are all inherited, except that η_i's have to be made larger. As now $\mathbf{x}_i$'s are updated in parallel and β_k changes adaptively, we call the new algorithm LADM with *parallel splitting* and *adaptive penalty* (LADMPSAP).

3.2. Stopping Criteria

Some existing work (e.g., (Liu et al., 2010; Favaro et al., 2011)) proposed stopping criteria out of intuition only, which may not guarantee that the correct solution is approached. Recently, Lin et al. (2009a) and Boyd et al. (2010) suggested that the stopping criteria can be derived from the KKT conditions of a problem. Here we also adopt such a strategy. Specifically, the iteration terminates when the following two conditions are met:

$$\left\|\sum_{i=1}^n \mathcal{A}_i(\mathbf{x}_i^{k+1}) - \mathbf{b}\right\|/\|\mathbf{b}\| < \varepsilon_1, \tag{18}$$

$$\max\left(\left\{\sqrt{\beta_k\sigma_i^{(k)}}\left\|\mathbf{x}_i^{k+1} - \mathbf{x}_i^k\right\|, i = 1, \cdots, n\right\}\right)/\|\mathbf{b}\| < \varepsilon_2. \tag{19}$$

The above stopping criteria (18) and (19) are deduced from the KKT condition (i.e., the criteria (18) and (19) are for the feasibility and duality conditions, respectively). Indeed, the update rules (14) and (17) for β are hinted by the stopping criteria (18) and (19) such that the two errors are well balanced.

For better reference, we summarize the proposed LADMPSAP algorithm in Algorithm 1.

3.3. Global Convergence

To prove the global convergence of LADMPSAP, we first have the following propositions[6].

6. Due to space limitations, all proofs in this paper are omitted. For these details, please consult our Supplementary Materials.

Algorithm 1 LADMPSAP for Solving Problem (1)

Initialize: Set $\varepsilon_1 > 0$, $\varepsilon_2 > 0$, $\beta_{\max} \gg 1 \gg \beta_0 > 0$, $\eta_i > n\|\mathcal{A}_i\|^2$, $\mathbf{x}_i^0$, $i = 1, \cdots, n$, λ^0.
while (18) or (19) is not satisfied **do**
Step 1: Compute $\hat{\lambda}^k$ as (16).
Step 2: Update $\mathbf{x}_i$'s in parallel by solving

$$\mathbf{x}_i^{k+1} = \underset{\mathbf{x}_i}{\operatorname{argmin}} f_i(\mathbf{x}_i) + \frac{\eta_i \beta_k}{2}\|\mathbf{x}_i - \mathbf{x}_i^k + \mathcal{A}_i^*(\hat{\lambda}^k)/(\eta_i\beta_k)\|^2, \ i = 1, \cdots, n. \tag{20}$$

Step 3: Update λ by (13), β by (14) and (17).
end while

Proposition 1 *Let $\{(\mathbf{x}_1^*, \cdots, \mathbf{x}_n^*, \lambda^*)\}$ be the Kuhn-Karush-Tucker (KKT) point of problem (1) and $\langle \cdot, \cdot \rangle$ be the inner product. Then we have*

$$\begin{aligned}
& \beta_k \sum_{i=1}^n \sigma_i^{(k)} \|\mathbf{x}_i^{k+1} - \mathbf{x}_i^*\|^2 + \|\lambda^{k+1} - \lambda^*\|^2 && (21)\\
\leq \ & \beta_k \sum_{i=1}^n \sigma_i^{(k)} \|\mathbf{x}_i^{k} - \mathbf{x}_i^*\|^2 + \|\lambda^{k} - \lambda^*\|^2 && (22)\\
& -2\beta_k \sum_{i=1}^n \left\langle \mathbf{x}_i^{k+1} - \mathbf{x}_i^*, -\sigma_i^{(k)}(\mathbf{x}_i^{k+1} - \mathbf{u}_i^k) + \mathcal{A}_i^*(\lambda^*) \right\rangle && (23)\\
& -\beta_k \sum_{i=1}^n \left(\sigma_i^{(k)} - n\beta_k\|\mathcal{A}_i\|^2\right) \|\mathbf{x}_i^{k+1} - \mathbf{x}_i^k\|^2 - \|\lambda^k - \hat{\lambda}^k\|^2. && (24)
\end{aligned}$$

Remark 2 *Proposition 1 shows that the sequence $\{(\mathbf{x}_1^k, \cdots, \mathbf{x}_n^k, \lambda^k)\}$ is Fejér monotone. Proposition 1 is different from Lemma 1 in Supplementary Material of Lin et al. (2011) because for $n > 2$ we cannot obtain an (in)equality that is similar to Lemma 1 in Supplementary Material of Lin et al. (2011) such that each term with minus sign could be made non-positive. Such Fejér monotone (in)equalities are the corner stones for proving the convergence of Lagrange multiplier based optimization algorithms. As a result, we cannot prove the convergence of the naively generalized LADM for the multi-variable case.*

Then we have the following proposition.

Proposition 3 *Let $\sigma_i^{(k)} = \eta_i\beta_k$, $i = 1, \cdots, n$. If $\{\beta_k\}$ is non-decreasing, $\eta_i > n\|\mathcal{A}_i\|^2$, $i = 1, \cdots, n$, and $(\mathbf{x}_1^*, \cdots, \mathbf{x}_n^*, \lambda^*)$ is any KKT point of problem (1), then:*

1) $\left\{\sum_{i=1}^n \eta_i\|\mathbf{x}_i^k - \mathbf{x}_i^*\|^2 + \beta_k^{-2}\|\lambda^k - \lambda^*\|^2\right\}$ *is nonnegative and non-increasing.*

2) $\|\mathbf{x}_i^{k+1} - \mathbf{x}_i^k\| \to 0$, $i = 1, \cdots, n$, $\beta_k^{-1}\|\lambda^k - \hat{\lambda}^k\| \to 0$.

3) $\sum_{k=1}^{+\infty} \beta_k^{-1} \left\langle \mathbf{x}_i^{k+1} - \mathbf{x}_i^*, -\sigma_i^{(k)}(\mathbf{x}_i^{k+1} - \mathbf{u}_i^k) + \mathcal{A}_i^*(\lambda^*) \right\rangle < +\infty$, $i = 1, \cdots, n$.

Now we can prove the global convergence of Algorithm 1, as stated in the following theorem, where we denote $\{\mathbf{x}_i^k\} = \{\mathbf{x}_1^k, \cdots, \mathbf{x}_n^k\}$ for simplicity.

Theorem 4 *If $\{\beta_k\}$ is non-decreasing and upper bounded, $\eta_i > n\|\mathcal{A}_i\|^2$, $i = 1, \cdots, n$, then $\{(\{\mathbf{x}_i^k\}, \lambda^k)\}$ generated by LADMPSAP converges to a KKT point of problem (1).*

3.4. Enhanced Convergence Results

Theorem 4 is a convergence result for general convex programs (1), where f_i's are general convex functions and hence $\{\beta_k\}$ need to be bounded. Actually, almost all the existing theories on ADM and LADM even assumed a fixed β. For adaptive β_k, it will be more convenient if a user need not specify an upper bound on $\{\beta_k\}$ because imposing a large upper bound essentially equals to allowing $\{\beta_k\}$ to be unbounded. Since many machine learning problems choose f_i's as matrix/vector norms, which result in bounded subgradients, we find that the boundedness assumption can be removed. Moreover, we can further prove the *sufficient and necessary* condition for global convergence.

We first have the following proposition.

Proposition 5 *If $\{\beta_k\}$ is non-decreasing and unbounded, $\eta_i > n\|\mathcal{A}_i\|^2$, $\partial f_i(\mathbf{x})$ is bounded, $i = 1, \cdots, n$, then Proposition 3 holds and*

$$\beta_k^{-1}\lambda^k \to 0. \tag{25}$$

Based on Proposition 5, we have the following enhanced theorems.

Theorem 6 *If $\{\beta_k\}$ is non-decreasing and $\sum_{k=1}^{+\infty} \beta_k^{-1} = +\infty$, $\eta_i > n\|\mathcal{A}_i\|^2$, $\partial f_i(\mathbf{x})$ is bounded, $i = 1, \cdots, n$, then the sequence $\{\mathbf{x}_i^k\}$ generated by LADMPSAP converges to an optimal solution to (1).*

Theorem 7 *If $\{\beta_k\}$ is non-decreasing, $\eta_i > n\|\mathcal{A}_i\|^2$, $\partial f_i(\mathbf{x})$ is bounded, $i = 1, \cdots, n$, then $\sum_{k=1}^{+\infty} \beta_k^{-1} = +\infty$ is also the necessary condition for the global convergence of $\{\mathbf{x}_i^k\}$ generated by LADMPSAP to an optimal solution to (1).*

With the above analysis, when all the subgradients of the component objective functions are bounded we can remove $\beta_{\max}$ in Algorithm 1.

3.5. Convergence Rate

The convergence rate of ADM and LADM in the traditional sense is an open problem (Goldfarb and Ma, 2012). Recently, He et al. (He and Yuan, 2011) and Tao (Tao, 2011) proved an $O(1/K)$ convergence rate of ADM and ADM with parallel splitting in an ergodic sense, respectively. Namely $\frac{1}{K}\sum_{k=1}^{K} \mathbf{x}_i$ violates an optimality measure in $O(1/K)$. Their proof is rather lengthy and is for fixed penalty parameter only. In this subsection, based on a simple optimality measure we give a simple proof for the convergence rate of LADMPSAP. For simplicity, we define $\mathbf{x} = (\mathbf{x}_1^T, \cdots, \mathbf{x}_n^T)^T$, $\mathbf{x}^* = ((\mathbf{x}_1^*)^T, \cdots, (\mathbf{x}_2^*)^T)^T$ and $f(\mathbf{x}) = \sum_{i=1}^{n} f_i(\mathbf{x}_i)$, where $(\mathbf{x}_1^*, \cdots, \mathbf{x}_2^*, \lambda^*)$ is a KKT point of (1). We first have the following proposition.

Proposition 8 *$\tilde{\mathbf{x}}$ is an optimal solution to (1) if and only if there exists $\alpha > 0$, such that*

$$f(\tilde{\mathbf{x}}) - f(\mathbf{x}^*) + \sum_{i=1}^{n} \langle \mathcal{A}_i^*(\lambda^*), \tilde{\mathbf{x}}_i - \mathbf{x}_i^* \rangle + \alpha \left\| \sum_{i=1}^{n} \mathcal{A}_i(\tilde{\mathbf{x}}_i) - \mathbf{b} \right\|^2 = 0. \tag{26}$$

By the above proposition, we may use the magnitude of the left hand side of (26), which is nonnegative, to measure the optimality of a point $\tilde{\mathbf{x}}$. Note that in the unconstrained case, as in APG, one may simply use $f(\tilde{\mathbf{x}}) - f(\mathbf{x}^*)$ to measure the optimality. But here we have to deal with the constraints.

Remark 9 *Our criterion for checking the optimality of a solution only requires comparing with the optimal solution. It is much simpler than that in (He and Yuan, 2011; Tao, 2011), which has to compare with all $(\mathbf{x}_1, \cdots, \mathbf{x}_n, \lambda) \in \mathbb{R}^{d_1} \times \cdots \times \mathbb{R}^{d_n} \times \mathbb{R}^m$.*

Then we have the following convergence rate theorem for LADMPSAP in an ergodic sense.

Theorem 10 *Define $\bar{\mathbf{x}}^K = \sum_{k=0}^{K} \gamma_k \mathbf{x}^{k+1}$, where $\gamma_k = \beta_k^{-1} / \sum_{j=0}^{K} \beta_j^{-1}$. Then*

$$f(\bar{\mathbf{x}}^K) - f(\mathbf{x}^*) + \sum_{i=1}^{n} \langle \mathcal{A}_i^*(\lambda^*), \bar{\mathbf{x}}_i^K - \mathbf{x}_i^* \rangle + \frac{\alpha\beta_0}{2} \left\| \sum_{i=1}^{n} \mathcal{A}_i(\bar{\mathbf{x}}_i^K) - \mathbf{b} \right\|^2 \leq C_0 / \left(2 \sum_{k=0}^{K} \beta_k^{-1} \right), \tag{27}$$

where $\alpha^{-1} = (n+1) \max\left(1, \left\{ \frac{\|\mathcal{A}_i\|^2}{\eta_i - n\|\mathcal{A}_i\|^2}, i = 1, \cdots, n \right\}\right)$ and $C_0 = \sum_{i=1}^{n} \eta_i \left\|\mathbf{x}_i^0 - \mathbf{x}_i^*\right\|^2 + \beta_0^{-2} \left\|\lambda^0 - \lambda^*\right\|^2$.

Theorem 10 means that $\bar{\mathbf{x}}^K$ is by $O\left(1/\sum_{k=0}^{K} \beta_k^{-1}\right)$ from being an optimal solution. This theorem holds for both bounded and unbounded $\{\beta_k\}$. In the bounded case, $O\left(1/\sum_{k=0}^{K} \beta_k^{-1}\right)$ is simply $O(1/K)$.

4. Practical LADMPSAP for Even More General Convex Programs

In real applications, we are often faced with convex programs with convex set constraints:

$$\min_{\mathbf{x}_1, \cdots, \mathbf{x}_n} \sum_{i=1}^{n} f_i(\mathbf{x}_i), \ \ s.t. \ \sum_{i=1}^{n} \mathcal{A}_i(\mathbf{x}_i) = \mathbf{b}, \ \mathbf{x}_i \in X_i, i = 1, \cdots, n, \tag{28}$$

where $X_i \subseteq \mathbb{R}^{d_i}$ is a closed convex set. In this section, we assume that the projections onto X_i's are all easily computable. For many convex sets used in machine learning, such an assumption is valid, e.g., X_i's are nonnegative cones or positive semi-definite cones. In the following, we discuss how to solve (28) efficiently. For simplicity, we assume $X_i \neq \mathbb{R}^{d_i}$, $\forall i$.

We introduce auxiliary variables $\mathbf{x}_{n+i}$ to convert $\mathbf{x}_i \in X_i$ into $\mathbf{x}_i = \mathbf{x}_{n+i}$ and $\mathbf{x}_{n+i} \in X_i$, $i = 1, \cdots, n$. Then (28) can be reformulated as:

$$\min_{\mathbf{x}_1,\cdots,\mathbf{x}_{2n}} \sum_{i=1}^{2n} f_i(\mathbf{x}_i), \quad s.t. \quad \sum_{i=1}^{2n} \hat{\mathcal{A}}_i(\mathbf{x}_i) = \hat{\mathbf{b}}, \tag{29}$$

where $f_{n+i}(\mathbf{x}) \equiv \chi_{X_i}(\mathbf{x}) = \begin{cases} 0, & \text{if } \mathbf{x} \in X_i, \\ +\infty, & \text{otherwise}, \end{cases}$ is the characteristic function of X_i and

$$\hat{\mathcal{A}}_i(\mathbf{x}_i) = (\mathcal{A}_i(\mathbf{x}_i), 0, \cdots, \mathbf{x}_i, \cdots, 0)^T, \hat{\mathcal{A}}_{n+i}(\mathbf{x}_{n+i}) = (0, 0, \cdots, -\mathbf{x}_{n+i}, \cdots, 0)^T, i = 1, \cdots, n,$$
$$\hat{\mathbf{b}} = (\mathbf{b}, 0, \cdots, 0, \cdots, 0)^T. \tag{30}$$

The adjoint operator $\hat{\mathcal{A}}_i^*$ is

$$\hat{\mathcal{A}}_i^*(\mathbf{y}) = \mathcal{A}_i^*(\mathbf{y}_1) + \mathbf{y}_{i+1}, \ \hat{\mathcal{A}}_{n+i}^*(\mathbf{y}) = -\mathbf{y}_{i+1}, \ i = 1, \cdots, n, \tag{31}$$

where $\mathbf{y}_i$ is the i-th sub-vector of $\mathbf{y}$, partitioned according to the sizes of $\mathbf{b}$ and $\mathbf{x}_i$, $i = 1, \cdots, n$.

Then LADMPSAP can be applied to solve problem (29). The Lagrange multiplier λ and the auxiliary multiplier $\hat{\lambda}$ are updated as

$$\lambda_1^{k+1} = \lambda_1^k + \beta_k \left(\sum_{i=1}^{n} \mathcal{A}_i(\mathbf{x}_i^{k+1}) - \mathbf{b} \right), \ \lambda_{i+1}^{k+1} = \lambda_{i+1}^k + \beta_k(\mathbf{x}_i^{k+1} - \mathbf{x}_{n+i}^{k+1}), \ i = 1, \cdots, n, \tag{32}$$

$$\hat{\lambda}_1^k = \lambda_1^k + \beta_k \left(\sum_{i=1}^{n} \mathcal{A}_i(\mathbf{x}_i^k) - \mathbf{b} \right), \ \hat{\lambda}_{i+1}^k = \lambda_{i+1}^k + \beta_k(\mathbf{x}_i^k - \mathbf{x}_{n+i}^k), \ i = 1, \cdots, n, \tag{33}$$

respectively, and $\mathbf{x}_i$ is updated as (see (12))

$$\mathbf{x}_i^{k+1} = \operatorname*{argmin}_{\mathbf{x}} f_i(\mathbf{x}) + \frac{\eta_i \beta_k}{2} \left\| \mathbf{x} - \mathbf{x}_i^k + [\mathcal{A}_i^*(\hat{\lambda}_1^k) + \hat{\lambda}_{i+1}^k]/(\eta_i \beta_k) \right\|^2, \ i = 1, \cdots, n, \tag{34}$$

$$\begin{aligned} \mathbf{x}_{n+i}^{k+1} &= \operatorname*{argmin}_{\mathbf{x} \in X_i} \frac{\eta_{n+i}\beta_k}{2} \|\mathbf{x} - \mathbf{x}_{n+i}^k - \hat{\lambda}_{i+1}^k/(\eta_{n+i}\beta_k)\|^2 \\ &= \pi_{X_i}(\mathbf{x}_{n+i}^k + \hat{\lambda}_{i+1}^k/(\eta_{n+i}\beta_k)), \ i = 1, \cdots, n, \end{aligned} \tag{35}$$

where π_{X_i} is the projection onto X_i.

Finally, we summarize LADMPSAP for problem (29) in Algorithm 2, which is indeed the practical algorithm to solve (28).

As for the choice of η_i's, although we can simply apply Theorem 4 to assign their values $\eta_i > 2n(\|\mathcal{A}_i\|^2 + 1)$ and $\eta_{n+i} > 2n$, $i = 1, \cdots, n$, such choices are too pessimistic. As η_i's are related to the magnitudes of the differences in $\mathbf{x}_i^{k+1}$ from $\mathbf{x}_i^k$, we had better provide tighter estimate on η_i's in order to achieve faster convergence. Actually, we have the following better result.

Theorem 11 *For problem (29), if $\{\beta_k\}$ is non-decreasing and upper bounded and η_i's are chosen as $\eta_i > n\|\mathcal{A}_i\|^2 + 2$ and $\eta_{n+i} > 2$, $i = 1, \cdots, n$, then the sequence $\{(\{\mathbf{x}_i^k\}, \lambda^k)\}$ generated by LADMPSAP converges to a KKT point of problem (29).*

Remark 12 *Analogs of Theorems 6 and 7 are also true for Algorithm 2 although ∂f_{n+i}'s are unbounded. Consequently, $\beta_{\max}$ can also be removed if all ∂f_i, $i = 1, \cdots, n$, are bounded.*

Algorithm 2 LADMPSAP for Problem (29), also the Practical Algorithm for (28).

Initialize: Set $\varepsilon_1 > 0$, $\varepsilon_2 > 0$, $\beta_{\max} \gg 1 \gg \beta_0 > 0$, $\eta_i > n\|\mathcal{A}_i\|^2 + 2$, $\eta_{n+i} > 2$, $\mathbf{x}_i^0$, $\mathbf{x}_{n+i}^0 = \mathbf{x}_i^0$, $i = 1, \cdots, n$, $\lambda^0 = ((\lambda_1^0)^T, \cdots, (\lambda_{n+1}^0)^T)^T$.
while (18) or (19) is not satisfied **do**
  Step 1: Compute $\hat{\lambda}^k$ as (33).
  Step 2: Update $\mathbf{x}_i$, $i = 1, \cdots, 2n$, in parallel as (34)-(35).
  Step 3: Update λ by (33) and β by (14) and (17).
end while
(Note that in (17), (18), and (19), n and $\mathcal{A}_i$ should be replaced by $2n$ and $\hat{\mathcal{A}}_i$, respectively.)

5. Numerical Results

In this section, we test the performance of LADMPSAP on two examples of problem (1) in machine learning[7].

5.1. Solving Latent LRR

LRR (Liu et al., 2010, 2012) is a recently proposed technique for robust subspace clustering and has been applied to many machine learning and computer vision problems. However, LRR works well only when the number of samples is more than the dimension of the samples, which may not be satisfied when the data dimension is high. So Liu et al. proposed latent LRR (Liu and Yan, 2011) to overcome this difficulty. The mathematical model of latent LRR is as follows:

$$\min_{\mathbf{Z},\mathbf{L},\mathbf{E}} \|\mathbf{Z}\|_* + \|\mathbf{L}\|_* + \mu\|\mathbf{E}\|_1, \quad s.t. \quad \mathbf{X} = \mathbf{XZ} + \mathbf{LX} + \mathbf{E}. \tag{36}$$

In order to test LADMPSAP and related algorithms with data whose characteristics are controllable, we follow (Liu et al., 2010) to generate synthetic data, which are parameterized as $(s, p, d, \tilde{r})$, where s, p, d, and $\tilde{r}$ are the number of independent subspaces, points in each subspace, and ambient and intrinsic dimensions, respectively. The number of scale variables and constraints is $(sp) \times d$.

As first order algorithms have been proved more efficient than standard solvers for convex programs in modern machine learning society (Boyd et al., 2010), here we compare LADMPSAP with several conceivable first order algorithms, including APG, naive ADM, naive LADM, LADMGB, and LADMPS. Naive ADM and naive LADM are generalizations of ADM and LADM, respectively, which are straightforwardly generalized from two variables to multiple variables. Naive ADM is applied to solve (36) after rewriting the constraint of (36) as $\mathbf{X} = \mathbf{XP} + \mathbf{QX} + \mathbf{E}, \mathbf{P} = \mathbf{Z}, \mathbf{Q} = \mathbf{L}$. For LADMPS, β_k is fixed in order to show the effectiveness of adaptive penalty. The parameters of APG and ADM are the same as those in (Lin et al., 2009b) and (Liu and Yan, 2011), respectively. For LADM, we follow the suggestions in (Yang and Yuan, 2013) to fix its penalty parameter β at $2.5/\min(d, sp)$, where $d \times sp$ is the size of $\mathbf{X}$. For LADMGB, as there is no suggestion in He and Yuan (2013) on how

7. To perform fair comparison for these algorithms, we utilize PROPACK (Larsen, 1998) to solve SVD in each iterations for all the compared algorithms. More details on the experiments, including experimental settings and descriptions of other algorithms for comparison, can be found in Supplementary Materials.

to choose a fixed β, we simply set it the same as that in LADM. The rest of the parameters are the same as suggested in (He et al., 2012). We fix $\beta = \sigma_{\max}(\mathbf{X}) \min(d, sp)\varepsilon_2$ in LADMPS and set $\beta_0 = \sigma_{\max}(\mathbf{X}) \min(d, sp)\varepsilon_2$ and $\rho_0 = 10$ in LADMPSAP. For LADMPSAP, we also set $\eta_Z = \eta_L = 1.02 \times 3\sigma^2_{\max}(\mathbf{X})$, where η_Z and η_L are the parameters η_i's in Algorithm 1 for $\mathbf{Z}$ and $\mathbf{L}$, respectively. For the stopping criteria, $\|\mathbf{X}\mathbf{Z}^k + \mathbf{L}^k\mathbf{X} + \mathbf{E}^k - \mathbf{X}\|/\|\mathbf{X}\| \le \varepsilon_1$ and $\max(\|\mathbf{Z}^k - \mathbf{Z}^{k-1}\|, \|\mathbf{L}^k - \mathbf{L}^{k-1}\|, \|\mathbf{E}^k - \mathbf{E}^{k-1}\|)/\|\mathbf{X}\| \le \varepsilon_2$, with $\varepsilon_1 = 10^{-3}$ and $\varepsilon_2 = 10^{-4}$ are used for all the algorithms. For the parameter μ in (36), we empirically set it as $\mu = 0.01$. To measure the relative errors in the solutions we run LADMPSAP 2000 iterations with $\rho_0 = 1.01$ to obtain the estimated ground truth solution $(\mathbf{Z}^*, \mathbf{L}^*, \mathbf{E}^*)$. The experiments are run and timed on a notebook computer with an Intel Core i7 2.00 GHz CPU and 6GB memory, running Windows 7 and Matlab 7.13.

Table 1 shows the results of related algorithms. We can see that LADMPS and LADMPSAP is much faster and much more accurate than LADMGB, and LADMPSAP is even faster than LADMPS thanks to the adaptive penalty. Moreover, naive ADM and naive LADM have relatively poorer numerical accuracy, possibly due to converging to wrong solutions. The numerical accuracy of APG is also worse than those of LADMPS and LADMPSAP because it only solves an approximate problem by adding the constraint to the objective function as penalty. Note that although we do not require $\{\beta_k\}$ to be bounded, this does not imply that β_k will grow infinitely. As a matter of fact, when LADMPSAP terminates the final values of β_k are 21.1567, 42.2655, and 81.4227 for the three data settings, respectively.

5.2. Solving Nonnegative Matrix Completion

This subsection evaluates the performance of the practical LADMPSAP proposed in Section 4 for solving nonnegative matrix completion (NMC) (Xu et al., 2011), which is a specific case of problem (28). Specifically, NMC problem can be formulated as:

$$\min_{\mathbf{X},\mathbf{e}} \|\mathbf{X}\|_* + \frac{1}{2\mu}\|\mathbf{e}\|^2, \quad s.t. \quad \mathbf{b} = \mathcal{P}_\Omega(\mathbf{X}) + \mathbf{e},\ \mathbf{X} \ge 0, \tag{37}$$

where Ω is an index set and $\mathcal{P}_\Omega$ is a linear mapping that selects those elements whose indices are in Ω.

We first evaluate numerical performance on synthetic data to demonstrate the superiority of practical LADMPSAP with conventional LADM. The nonnegative low-rank matrix $\mathbf{X}_0$ is generated by truncating the singular values of a randomly generated matrix. As LADM cannot handle the nonnegativity constraint, it actually solve the standard matrix completion problem, i.e., (37) without the nonnegativity constraint. For LADMPSAP, we follow the conditions in Theorem 11 to set η_i's and set the rest of the parameters the same as those in Section 5.1. The stopping tolerances are set as $\varepsilon_1 = \varepsilon_2 = 10^{-5}$. The numerical comparison is shown in Table 2, where the relative nonnegative feasibility (FA) is defined as (Xu et al., 2011):

$$\text{FA} := \|\min(\hat{\mathbf{X}}, 0)\|/\|\mathbf{X}_0\|,$$

in which $\mathbf{X}_0$ is the ground truth image and $\hat{\mathbf{X}}$ is the recovered image. It can be seen that the numerical performance of LADMPSAP is much better than that of LADM, thus again verify the efficiency of our proposed parallel splitting and adaptive penalty for enhancing ADM/LADM type algorithms.

Table 1: Comparison of APG, naive ADM (nADM), naive LADM (nLADM), LADMGB, LADMPS and LADMPSAP on the latent LRR problem (36). The quantities include computing time (in seconds), number of iterations, relative errors, and clustering accuracy (in percentage). They are averaged over 10 runs.

$(s,\ p,\ d,\ \tilde{r})$	Method	Time(s)	#Iter.	$\frac{\|\hat{\mathbf{Z}}-\mathbf{Z}^*\|}{\|\mathbf{Z}^*\|}$	$\frac{\|\hat{\mathbf{L}}-\mathbf{L}^*\|}{\|\mathbf{L}^*\|}$	$\frac{\|\hat{\mathbf{E}}-\mathbf{E}^*\|}{\|\mathbf{E}^*\|}$	Acc.
(5, 50, 250, 5)	APG	18.20	236	0.3389	0.3167	0.4500	95.6
	nADM	16.32	172	0.3993	0.3928	0.5592	95.6
	nLADM	21.34	288	0.4553	0.4408	0.5607	95.6
	LADMGB	24.10	290	0.4520	0.4355	0.5610	95.6
	LADMPS	17.15	232	0.0163	0.0139	**0.0446**	95.6
	LADMPSAP	**8.04**	**109**	**0.0089**	**0.0083**	0.0464	95.6
(10, 50, 500, 5)	APG	85.03	234	0.1020	0.0844	0.7161	95.8
	nADM	78.27	170	0.0928	0.1026	0.6636	95.8
	nLADM	181.42	550	0.2077	0.2056	0.6623	95.8
	LADMGB	214.94	550	0.1877	0.1848	0.6621	95.8
	LADMPS	64.65	200	0.0167	0.0089	0.1059	95.8
	LADMPSAP	**37.85**	**117**	**0.0122**	**0.0055**	**0.0780**	95.8
(20, 50, 1000, 5)	APG	544.13	233	0.0319	0.0152	0.2126	95.2
	nADM	466.78	166	0.0501	0.0433	0.2676	95.2
	nLADM	1888.44	897	0.1783	0.1746	0.2433	95.2
	LADMGB	2201.37	897	0.1774	0.1736	0.2434	95.2
	LADMPS	367.68	177	0.0151	0.0105	0.0872	95.2
	LADMPSAP	**260.22**	**125**	**0.0106**	**0.0041**	**0.0671**	95.2

Table 2: Numerical comparison on the NMC problem (37) with synthetic data, average of 10 runs. q, t and d_r denote, respectively, sample ratio, the number of measurements $t = q(mn)$ and the "degree of freedom" defined by $d_r = r(m+n-r)$ for a matrix with rank r and q. Here we set $m = n$ and fix $r = 10$ in all the tests.

$\mathbf{X}$			LADM				LADMPSAP			
n	q	t/d_r	Iter.	Time(s)	RelErr	FA	Iter.	Time(s)	RelErr	FA
1000	20%	10.05	375	177.92	1.35E-5	6.21E-4	58	24.94	9.67E-6	0
	10%	5.03	1000	459.70	4.60E-5	6.50E-4	109	42.68	1.72E-5	0
5000	20%	50.05	229	1613.68	1.08E-5	1.93E-4	49	369.96	9.05E-6	0
	10%	25.03	539	2028.14	1.20E-5	7.70E-5	89	365.26	9.76E-6	0
10000	10%	50.03	463	6679.59	1.11E-5	4.18E-5	89	1584.39	1.03E-5	0

We then consider the image inpainting problem, which is to fill in the missing pixel values of a corrupted image. As the pixel values are nonnegative, the image inpainting problem

Table 3: Numerical comparison on the image inpainting problem.

Method	#Iter.	Time(s)	PSNR	FA
FPCA	179	228.99	27.77dB	9.41E-4
LADM	228	207.95	26.98dB	2.92E-3
LADMPSAP	143	134.89	31.39dB	0

can be formulated as NMC problem. To prepare a low-rank image, we also truncate the singular values of the 1024×1024 grayscale image "man"[8] to obtain an image of rank 40, shown in Fig. 1 (a)-(b). The corrupted image is generated from original image by sampling 20% of the pixels uniformly at random and adding Gaussian noise with mean zero and standard deviation 0.1.

Besides LADM, here we also consider another recently proposed fixed point continuation with approximate SVD (FPCA) on this problem. Similar to LADM, the code of FPCA can only solve the standard matrix completion problem without the nonnegativity constraint. This time we set $\varepsilon_1 = 10^{-3}$ and $\varepsilon_2 = 10^{-1}$ as the stopping criteria. The recovered images are shown in Fig. 1 (c)-(e) and the quantitative results are in Table 3. One can see that on our test image both the qualitative and the quantitative results of LADMPSAP are better than those of FPCA and LADM. Note that LADMPSAP is much faster than FPCA and LADM even though they do not handle the nonnegativity constraint.

(a) Original (b) Corrupted (c) FPCA (d) LADM (e) LADMPSAP

Figure 1: Image inpainting by FPCA, LADM and LADMPSAP.

6. Conclusions

In this paper, we propose linearized alternating direction method with parallel splitting and adaptive penalty for efficiently solving linearly constrained multi-variable separable convex programs, which are abundant in machine learning. LADMPSAP fully utilizes the properties that the prox-function of the component objective functions and the projections onto convex sets are easily solvable, which are usually satisfied by machine learning problems, making each of its iterations cheap. It is also highly parallel, making it appealing for parallel or distributed computing. Numerical experiments testify to the advantages of LADMPSAP over other possible first order methods.

8. Available at http://sipi.usc.edu/database/.

Acknowledgments

R. Liu is supported by NSFC (No. 61300086), the China Postdoctoral Science Foundation (No. 2013M530917) and the Fundamental Research Funds for the Central Universities (No. DUT12RC(3)67). Z. Lin is supported by NSFC (Nos. 61272341, 61231002, 61121002). Z. Su is supported by NSFC (Nos. 61173103 and U0935004).

References

A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sciences*, 2(1):183–202, 2009.

S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. In M. Jordan, editor, *Foundations and Trends in Machine Learning*, 2010.

J. Cai, E. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM J. Optimization*, 20(4):1956–1982, 2010.

E. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

E. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis? *J. ACM*, 58(3):No.11, 2011.

V. Chandrasekaran, P. Parrilo, and A. Willsky. Latent variable graphical model selection via convex optimization. *The Annals of Statistics*, 40(4):1935–1967, 2012.

E. Esser. Applications of Lagrangian-based alternating direction methods and connections to split Bregman. *CAM Report 09-31, UCLA*, 2009.

P. Favaro, R. Vidal, and A. Ravichandran. A closed form solution to robust subspace estimation and clustering. In *CVPR*, 2011.

M. Fazel. Matrix rank minimization with applications. PhD thesis, 2002.

M. Fortin and R. Glowinski. *Augmented Lagrangian methods*. North-Holland, 1983.

D. Goldfarb and S. Ma. Fast multiple splitting algorithms for convex optimization. *SIAM J. Optimization*, 22(2):533–556, 2012.

T. Goldstein and S. Osher. The split Bregman method for l_1 regularized problems. *SIAM J. Imaging Sciences*, 2(2):323–343, 2008.

B. S. He and X. Yuan. On the $O(1/t)$ convergence rate of alternating direction method. *Preprint*, 2011.

B. S. He and X. Yuan. Linearized alternating direction method with Gaussian back substitution for separable convex programming. *Numerical Algebra, Control and Optimization*, 3(2):247–260, 2013.

B. S. He, M. Tao, and X. Yuan. Alternating direction method with Gaussian back substitution for separable convex programming. *SIAM J. Optimization*, 22(2):313–340, 2012.

H. Ji, C. Liu, Z. Shen, and Y. Xu. Robust video denoising using low rank matrix completion. In *CVPR*, 2010.

R. Larsen. Lanczos bidiagonalization with partial reorthogonalization. *Department of Computer Science, Aarhus University, Technical report, DAIMI PB-357*, 1998.

Z. Lin, M. Chen, and Y. Ma. The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices. *UIUC Technical Report UILU-ENG-09-2215*, 2009a.

Z. Lin, A. Ganesh, J. Wright, L. Wu, M. Chen, and Y. Ma. Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix. *UIUC Technical Report UILU-ENG-09-2214*, 2009b.

Z. Lin, R. Liu, and Z. Su. Linearized alternating direction method with adaptive penalty for low-rank representation. In *NIPS*, 2011.

G. Liu and S. Yan. Latent low-rank representation for subspace segmentation and feature extraction. In *ICCV*, 2011.

G. Liu, Z. Lin, and Y. Yu. Robust subspace segmentation by low-rank representation. In *ICML*, 2010.

G. Liu, Z. Lin, S. Yan, J. Sun, Y. Yu, and Y. Ma. Robust recovery of subspace structures by low-rank representation. *IEEE Trans. on PAMI*, 35(1):171–184, 2012.

X. Shen and Y. Wu. A unified approach to salient object detection via low rank matrix recovery. In *CVPR*, 2012.

M. Tao. Some parallel splitting methods for separable convex programming with $O(1/t)$ convergence rate. *Preprint*, 2011.

K. Toh and S. Yun. An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. *Pacific J. Optimization*, 6(15):615–640, 2010.

J. Wright, A. Yang, A. Ganesh, S. Sastry, and Y. Ma. Robust face recognition via sparse representation. *IEEE Trans. on PAMI*, 31(2):210–227, 2009.

Y. Xu, W. Yin, and Z. Wen. An alternating direction algorithm for matrix completion with nonnegative factors. *CAAM Technical Report TR11-03*, 2011.

J. Yang and X. Yuan. Linearized augmented Lagrangian and alternating direction methods for nuclear norm minimization. *Mathematics of Computation*, 82(281):301–329, 2013.

J. Ye, S. Ji, and J. Chen. Multi-class discriminant kernel learning via convex programming. *JMLR*, 9:719–758, 2008.