Fast Proximal Linearized Alternating Direction Method of Multiplier with Parallel Splitting

Canyi Lu[1], **Huan Li**[2], **Zhouchen Lin**[2,3,*], **Shuicheng Yan**[1]
[1] Department of Electrical and Computer Engineering, National University of Singapore
[2] Key Laboratory of Machine Perception (MOE), School of EECS, Peking University
[3] Cooperative Medianet Innovation Center, Shanghai Jiaotong University
canyilu@gmail.com, lihuan_ss@126.com, zlin@pku.edu.cn, eleyans@nus.edu.sg

Abstract

The Augmented Lagragian Method (ALM) and Alternating Direction Method of Multiplier (ADMM) have been powerful optimization methods for general convex programming subject to linear constraint. We consider the convex problem whose objective consists of a smooth part and a nonsmooth but simple part. We propose the Fast Proximal Augmented Lagragian Method (Fast PALM) which achieves the convergence rate $O(1/K^2)$, compared with $O(1/K)$ by the traditional PALM. In order to further reduce the per-iteration complexity and handle the multi-blocks problem, we propose the Fast Proximal ADMM with Parallel Splitting (Fast PL-ADMM-PS) method. It also achieves the rate $O(1/K^2)$ for the smooth part of the objective function. Experimental results on both synthesized and real world data demonstrate that our fast methods significantly improve the previous PALM and ADMM.

Introduction

This work aims to solve the following linearly constrained separable convex problem with n blocks of variables

$$\begin{aligned} \min_{\mathbf{x}_1,\cdots,\mathbf{x}_n} f(\mathbf{x}) &= \sum_{i=1}^{n} f_i(\mathbf{x}_i) = \sum_{i=1}^{n} \left(g_i(\mathbf{x}_i) + h_i(\mathbf{x}_i)\right), \\ \text{s.t.} \quad \mathcal{A}(\mathbf{x}) &= \sum_{i=1}^{n} \mathcal{A}_i(\mathbf{x}_i) = \mathbf{b}, \end{aligned} \tag{1}$$

where $\mathbf{x}_i$'s and $\mathbf{b}$ can be vectors or matrices and both g_i and h_i are convex and lower semi-continuous. For g_i, we assume that ∇g_i is Lipschitz continuous with the Lipschitz constant $L_i > 0$, i.e, $\|\nabla g_i(\mathbf{x}_i) - \nabla g_i(\mathbf{y}_i)\| \leq L_i \|\mathbf{x}_i - \mathbf{y}_i\|, \forall \mathbf{x}_i, \mathbf{y}_i$. For h_i, we assume that it may be nonsmooth and it is simple, in the sense that the proximal operator problem $\min_{\mathbf{x}} h_i(\mathbf{x}) + \frac{\alpha}{2}||\mathbf{x} - \mathbf{a}||^2$ ($\alpha > 0$) can be cheaply solved. The bounded mappings $\mathcal{A}_i$'s are linear (e.g., linear transformation or the sub-sampling operator in matrix completion (Candès and Recht 2009)). For the simplicity of discussion, we denote $\mathbf{x} = [\mathbf{x}_1; \mathbf{x}_2; \cdots; \mathbf{x}_n]$, $\mathcal{A} = [\mathcal{A}_1, \mathcal{A}_2, \cdots, \mathcal{A}_n]$ and $\sum_{i=1}^{n} \mathcal{A}_i(\mathbf{x}_i) = \mathcal{A}(\mathbf{x})$, $f_i = g_i + h_i$. For any compact set X, let $D_X = \sup_{\mathbf{x}_1,\mathbf{x}_2 \in X} ||\mathbf{x}_1 - \mathbf{x}_2||$ be the diameter of X. We also denote $\mathbf{D}_{\mathbf{x}^*} = ||\mathbf{x}^0 - \mathbf{x}^*||$. We assume there exists a saddle point $(\mathbf{x}^*, \boldsymbol{\lambda}^*) \in X \times \Lambda$ to (1), i.e., $\mathcal{A}(\mathbf{x}^*) = \mathbf{b}$ and $-\mathcal{A}_i^T(\boldsymbol{\lambda}^*) \in \partial f_i(\mathbf{x}_i^*),\ i = 1, \cdots, n$, where $\mathcal{A}^T$ is the adjoint operator of $\mathcal{A}$, X and Λ are the feasible sets of the primal variables and dual variables, respectively.

By using different g_i's and h_i's, a variety of machine learning problems can be cast into (1), including Lasso (Tibshirani 1996) and its variants (Lu et al. 2013; Jacob, Obozinski, and Vert 2009), low rank matrix decomposition (Candès et al. 2011), completion (Candès and Recht 2009) and representation model (Lu et al. 2012; Liu and Yan 2011) and latent variable graphical model selection (Chandrasekaran, Parrilo, and Willsky 2012). Specifically, examples of g_i are: (i) the square loss $\frac{1}{2}||\mathbf{D}\mathbf{x} - \mathbf{y}||^2$, where $\mathbf{D}$ and $\mathbf{y}$ are of compatible dimensions. A more special case is the known Laplacian regularizer $\mathrm{Tr}(\mathbf{X}\mathbf{L}\mathbf{X}^T)$, where $\mathbf{L}$ is the Laplacian matrix which is positive semi-definite; (ii) Logistic loss $\sum_{i=1}^{m} \log(1 + \exp(-y_i \mathbf{d}_i^T \mathbf{x}))$, where $\mathbf{d}_i$'s and y_i's are the data points and the corresponding labels, respectively; (iii) smooth-zero-one loss $\sum_{i=1}^{m} \frac{1}{1+\exp(c y_i \mathbf{d}_i^T \mathbf{x})}$, $c > 0$. The possibly nonsmooth h_i can be many norms, e.g., ℓ_1-norm $||\cdot||_1$ (the sum of absolute values of all entries), ℓ_2-norm $||\cdot||$ or Frobenius norm $||\cdot||_F$ and nuclear norm $||\cdot||_*$ (the sum of the singular values of a matrix).

This paper focuses on the popular approaches which study problem (1) from the aspect of the augmented Lagrangian function $L(\mathbf{x}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \langle \boldsymbol{\lambda}, \mathcal{A}(\mathbf{x}) - \mathbf{b} \rangle + \frac{\beta}{2}||\mathcal{A}(\mathbf{x}) - \mathbf{b}||^2$, where $\boldsymbol{\lambda}$ is the Lagrangian multiplier or dual variable and $\beta > 0$. A basic idea to solve problem (1) based on $L(\mathbf{x}, \boldsymbol{\lambda})$ is the Augmented Lagrangian Method (ALM) (Hestenes 1969), which is a special case of the Douglas-Rachford splitting (Douglas and Rachford 1956).

An influential variant of ALM is the Alternating Direction Mehtod of Multiplier (ADMM) (Boyd et al. 2011), which solves problem (1) with $n = 2$ blocks of variables. However, the cost for solving the subproblems in ALM and ADMM in each iteration is usually high when f_i is not simple and $\mathcal{A}_i$ is non-unitary ($\mathcal{A}_i^T \mathcal{A}_i$ is not the identity mapping). To alleviate this issue, the Linearized ALM (LALM) (Yang and Yuan 2013) and Linearized ADMM (LADMM) (Lin, Liu, and Su 2011) were proposed by linearizing the aug-

*Corresponding author.

PALM	Fast PALM	PL-ADMM-PS	Fast PL-ADMM-PS
$O\left(\frac{D^2_{\mathbf{x}^*}+D^2_{\boldsymbol{\lambda}^*}}{K}\right)$	$O\left(\frac{D^2_{\mathbf{x}^*}+D^2_{\boldsymbol{\lambda}^*}}{K^2}\right)$	$O\left(\frac{D^2_{\mathbf{x}^*}}{K}+\frac{D^2_{\mathbf{x}^*}}{K}+\frac{D^2_{\boldsymbol{\lambda}^*}}{K}\right)$	$O\left(\frac{D^2_{\mathbf{x}^*}}{K^2}+\frac{D^2_{X}}{K}+\frac{D^2_{\Lambda}}{K}\right)$

Table 1: Comparison of the convergence rates of previous methods and our fast versions

mented term $\frac{\beta}{2}||\mathcal{A}(\mathbf{x})-\mathbf{b}||^2$ and thus the subproblems are easier to solve. For (1) with $n>2$ blocks of variables, the Proximal Jacobian ADMM (Tao 2014) and Linearized ADMM with Parallel Splitting (L-ADMM-PS) (Lin, Liu, and Li 2014) guaranteed to solve (1) when $g_i=0$ with convergence guarantee. To further exploit the Lipschitz continuous gradient property of g_i's in (1), the work (Lin, Liu, and Li 2014) proposed a Proximal Linearized ADMM with Parallel Splitting (PL-ADMM-PS) by further linearizing the smooth part g_i. PL-ADMM-PS requires lower per-iteration cost than L-ADMM-PS for solving the general problem (1).

Beyond the per-iteration cost, another important way to measure the speed of the algorithms is the convergence rate. Several previous work proved the convergence rates of the augmented Lagrangian function based methods (He and Yuan 2012; Tao 2014; Lin, Liu, and Li 2014). Though the convergence functions used to measure the convergence rate are different, the convergence rates of all the above discussed methods for (1) are all $O(1/K)$, where K is the number of iterations. However, the rate $O(1/K)$ may be suboptimal in some cases. Motivated by the seminal work (Nesterov 1983), several fast first-order methods with the optimal rate $O(1/K^2)$ have been developed for unconstrained problems (Beck and Teboulle 2009; Tseng 2008). More recently, by applying a similar accelerating technique, several fast ADMMs have been proposed to solve a special case of problem (1) with $n=2$ blocks of variables

$$\min_{\mathbf{x}_1,\mathbf{x}_2} g_1(\mathbf{x}_1)+h_2(\mathbf{x}_2),\ \text{ s.t. } \mathcal{A}_1(\mathbf{x}_1)+\mathcal{A}_2(\mathbf{x}_2)=\mathbf{b}. \quad (2)$$

A fast ADMM proposed in (Azadi and Sra 2014)[1] is able to solve (2) with the convergence rate $O\left(\frac{D^2_X}{K^2}+\frac{D^2_\Lambda}{K}\right)$. But their result is a bit weak since their used function to characterize the convergence can be negative. The work (Ouyang et al. 2015) proposed another fast ADMM with the rate $O\left(\frac{D^2_{\mathbf{x}^*}}{K^2}+\frac{D^2_{\mathbf{x}^*}}{K}\right)$ for primal residual and $O\left(\frac{D^2_{\mathbf{x}^*}}{K^{3/2}}+\frac{D_{\mathbf{x}^*}+D_{\boldsymbol{\lambda}^*}}{K}\right)$ for feasibility residual. However, their result requires that the number of iterations K should be predefined, which is not reasonable in practice. It is usually difficult in practice to determine the optimal K since we usually stop the algorithms when both the primal and feasibility residuals are sufficiently small (Lin, Liu, and Li 2014). The fast ALM proposed in (He and Yuan 2010) owns the convergence rate $O(1/K^2)$, but it requires the objective f to be differentiable. This limits its applications for nonsmooth optimization in most compressed sensing problems. Another work (Goldstein et al. 2014) proved a better convergence rate than $O(1/K)$ for ADMM. But their method requires much stronger assumptions, e.g., strongly convexity of f_i's, which are usually violated in practice. In this work, we only consider (1) whose objective is not necessarily strongly convex.

In this work, we aim to propose fast ALM type methods to solve the general problem (1) with optimal convergence rates. The contributions are summarized as follows:

- First, we consider (1) with $n=1$ (or one may regard all n blocks as a superblock) and propose the Fast Proximal Augmented Lagrangian Method (Fast PALM). We prove that Fast PALM converges with the rate $O\left(\frac{D^2_{\mathbf{x}^*}+D^2_{\boldsymbol{\lambda}^*}}{K^2}\right)$, which is a significant improvement of ALM/PALM[2] with rate $O\left(\frac{D^2_{\mathbf{x}^*}+D^2_{\boldsymbol{\lambda}^*}}{K}\right)$. To the best of our knowledge, Fast PALM is the first improved ALM/PALM which achieves the rate $O(1/K^2)$ for the nonsmooth problem (1).
- Second, we consider (1) with $n>2$ and propose the Fast Proximal Linearized ADMM with Parallel Splitting (Fast PL-ADMM-PS), which converges with rate $O\left(\frac{D^2_{\mathbf{x}^*}}{K^2}+\frac{D^2_X}{K}+\frac{D^2_\Lambda}{K}\right)$. As discussed in Section 1.3 of (Ouyang et al. 2015), such a rate is optimal and thus is better than PL-ADMM-PS with rate $O\left(\frac{D^2_{\mathbf{x}^*}}{K}+\frac{D^2_{\mathbf{x}^*}}{K}+\frac{D^2_{\boldsymbol{\lambda}^*}}{K}\right)$ (Lin, Liu, and Li 2014). To the best of our knowledge, Fast PL-ADMM-PS is the first fast Jacobian type (update the variables in parallel) method to solve (1) when $n>2$ with convergence guarantee.

Table 1 shows the comparison of the convergence rates of previous methods and our fast versions. Note that Fast PALM and Fast PL-ADMM-PS have the same pter-iteration cost as PALM and PL-ADMM-PS, respectively. But the per-iteration cost of PL-ADMM-PS and Fast PL-ADMM-PS may be much cheaper than PALM and Fast PALM.

Fast Proximal Augmented Lagrangian Method

In this section, we consider (1) with $n=1$ block of variable,

$$\min_{\mathbf{x}} f(\mathbf{x})=g(\mathbf{x})+h(\mathbf{x}),\quad \text{s.t.}\quad \mathcal{A}(\mathbf{x})=\mathbf{b}, \quad (3)$$

where g and h are convex and ∇g is Lipschitz continuous with the Lipschitz constant L. The above problem can be solved by the traditional ALM which updates $\mathbf{x}$ and $\boldsymbol{\lambda}$ by

$$\begin{cases} \mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} g(\mathbf{x})+h(\mathbf{x})+\langle\boldsymbol{\lambda}^k,\mathcal{A}(\mathbf{x})-\mathbf{b}\rangle \\ \qquad\qquad +\dfrac{\beta^{(k)}}{2}||\mathcal{A}(\mathbf{x})-\mathbf{b}||^2, \\ \boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k+\beta^{(k)}(\mathcal{A}(\mathbf{x}^{k+1})-\mathbf{b}), \end{cases} \quad (4)$$

[1]The method in (Azadi and Sra 2014) is a fast stochastic ADMM. It is easy to give the corresponding deterministic version by computing the gradient in each iteration exactly to solve (2).

[2]PALM is a variant of ALM proposed in this work.

Initialize: $\mathbf{x}^0$, $\mathbf{z}^0$, $\boldsymbol{\lambda}^0$, $\beta^{(0)} = \theta^{(0)} = 1$.
for $k = 0, 1, 2, \cdots$ **do**

$$\mathbf{y}^{k+1} = (1-\theta^{(k)})\mathbf{x}^k + \theta^{(k)}\mathbf{z}^k; \quad (6)$$

$$\mathbf{z}^{k+1} = \operatorname*{argmin}_{\mathbf{x}} \left\langle \nabla g(\mathbf{y}^{k+1}), \mathbf{x} \right\rangle + h(\mathbf{x}) + \left\langle \boldsymbol{\lambda}^k, \mathcal{A}(\mathbf{x}) \right\rangle + \frac{\beta^{(k)}}{2}\|\mathcal{A}(\mathbf{x}) - \mathbf{b}\|^2 + \frac{L\theta^{(k)}}{2}\|\mathbf{x} - \mathbf{z}^k\|^2; \quad (7)$$

$$\mathbf{x}^{k+1} = (1-\theta^{(k)})\mathbf{x}^k + \theta^{(k)}\mathbf{z}^{k+1}; \quad (8)$$

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \beta^{(k)}(\mathcal{A}(\mathbf{z}^{k+1}) - \mathbf{b}); \quad (9)$$

$$\theta^{(k+1)} = \frac{-(\theta^{(k)})^2 + \sqrt{(\theta^{(k)})^4 + 4(\theta^{(k)})^2}}{2}; \quad (10)$$

$$\beta^{(k+1)} = \frac{1}{\theta^{(k+1)}}. \quad (11)$$

end

Algorithm 1: Fast PALM Algorithm

where $\beta^{(k)} > 0$. Note that ∇g is Lipschitz continuous. We have (Nesterov 2004)

$$g(\mathbf{x}) \le g(\mathbf{x}^k) + \langle \nabla g(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + \frac{L}{2}||\mathbf{x} - \mathbf{x}^k||^2. \quad (5)$$

This motivates us to use the right hand side of (5) as a surrogate of g in (4). Thus we can update $\mathbf{x}$ by solving the following problem which is simpler than (5),

$$\mathbf{x}^{k+1} = \arg\min_{\mathbf{x}} g(\mathbf{x}^k) + \langle \nabla g(\mathbf{x}^k), \mathbf{x} - \mathbf{x}^k \rangle + h(\mathbf{x}) + \langle \boldsymbol{\lambda}^k, \mathcal{A}(\mathbf{x}) - \mathbf{b} \rangle + \frac{\beta^{(k)}}{2}||\mathcal{A}(\mathbf{x}) - \mathbf{b}||^2 + \frac{L}{2}||\mathbf{x} - \mathbf{x}^k||^2.$$

We call the method by using the above updating rule as Proximal Augmented Lagrangian Method (PALM). PALM can be regarded as a special case of Proximal Linearized Alternating Direction Method of Multiplier with Parallel Splitting in (Lin, Liu, and Li 2014) and it owns the convergence rate $O(1/K)$, which is the same as the traditional ALM and ADMM. However, such a rate is suboptimal. Motivated by the technique from the accelerated proximal gradient method (Tseng 2008), we propose the Fast PALM as shown in Algorithm 1. It uses the interpolatory sequences $\mathbf{y}^k$ and $\mathbf{z}^k$ as well as the stepsize $\theta^{(k)}$. Note that if we set $\theta^{(k)} = 1$ in each iteration, Algorithm 1 reduces to PALM. With careful choices of $\theta^{(k)}$ and $\beta^{(k)}$ in Algorithm 1, we can accelerate the convergence rate of PALM from $O(1/K)$ to $O(1/K^2)$.

Proposition 1. *In Algorithm 1, for any* $\mathbf{x}$*, we have*

$$\frac{1-\theta^{(k+1)}}{(\theta^{(k+1)})^2}\left(f(\mathbf{x}^{k+1}) - f(\mathbf{x})\right) - \frac{1}{\theta^{(k)}}\left\langle \mathcal{A}^T(\boldsymbol{\lambda}^{k+1}), \mathbf{x} - \mathbf{z}^{k+1} \right\rangle \le \frac{1-\theta^{(k)}}{(\theta^{(k)})^2}\left(f(\mathbf{x}^k) - f(\mathbf{x})\right) + \frac{L}{2}\left(\|\mathbf{z}^k - \mathbf{x}\|^2 - \|\mathbf{z}^{k+1} - \mathbf{x}\|^2\right). \quad (12)$$

Theorem 1. *In Algorithm 1, for any* $K > 0$*, we have*

$$f(\mathbf{x}^{K+1}) - f(\mathbf{x}^*) + \left\langle \boldsymbol{\lambda}^*, \mathcal{A}(\mathbf{x}^{K+1}) - \mathbf{b} \right\rangle + \frac{1}{2}\|\mathcal{A}(\mathbf{x}^{K+1}) - \mathbf{b}\|^2 \le \frac{2}{(K+2)^2}\left(LD_{\mathbf{x}^*}^2 + D_{\boldsymbol{\lambda}^*}^2\right). \quad (13)$$

We use the convergence function, i.e., the left hand side of (13), in (Lin, Liu, and Li 2014) to measure the convergence rate of the algorithms in this work. Theorem 1 shows that our Fast PALM achieves the rate $O\left(\frac{LD_{\mathbf{x}^*}^2 + D_{\boldsymbol{\lambda}^*}^2}{K^2}\right)$, which is much better than $O\left(\frac{LD_{\mathbf{x}^*}^2 + \frac{1}{\beta}D_{\boldsymbol{\lambda}^*}^2}{K}\right)$ by PALM[3]. The improvement of Fast PALM over PALM is similar to the one of Fast ISTA over ISTA (Beck and Teboulle 2009; Tseng 2008). The difference is that Fast ISTA targets for unconstrained problem which is easier than our problem (1). Actually, if the constraint in (1) is dropped (i.e., $\mathcal{A} = \mathbf{0}$, $\mathbf{b} = \mathbf{0}$), our Fast PALM is similar as the Fast ISTA.

We would like to emphasize some key differences between our Fast PALM and previous fast ALM type methods (Azadi and Sra 2014; Ouyang et al. 2015; He and Yuan 2010). First, it is easy to apply the two blocks fast ADMM methods in (Azadi and Sra 2014; Ouyang et al. 2015) to solve problem (3). Following their choices of parameters and proofs, the convergence rates are still $O(1/K)$. The key improvement of our method comes from the different choices of $\theta^{(k)}$ and $\beta^{(k)}$ as shown in Theorem 1. The readers can refer to the detailed proofs in the supplementary material. Second, the fast ADMM in (Ouyang et al. 2015) requires predefining the total number of iterations, which is usually difficult in practice. However, our Fast PALM has no such a limitation. Third, the fast ALM in (He and Yuan 2010) also owns the rate $O(1/K^2)$. But it is restricted to differentiable objective minimization and thus is not applicable to our problem (1). Our method has no such a limitation.

A main limitation of PALM and Fast PALM is that their per-iteration cost may be high when h_i is nonsmooth and $\mathcal{A}_i$ is non-unitary. In this case, solving the subproblem (7) requires calling other iterative solver, e.g., Fast ISTA (Beck and Teboulle 2009), and thus the high per-iteration cost may limit the application of Fast PALM. In next section, we present a fast ADMM which has lower per-iteration cost.

Fast Proximal Linearized ADMM with Parallel Splitting

In this section, we consider problem (1) with $n > 2$ blocks of variables. The state-of-the-art solver for (1) is the Proximal Linearized ADMM with Parallel Splitting (PL-ADMM-PS) (Lin, Liu, and Li 2014) which updates each $\mathbf{x}_i$ in parallel

[3]It is easy to achieve this since PALM is a special case of Fast PALM by taking $\theta^{(k)} = 1$.

by

$$\mathbf{x}_i^{k+1} = \operatorname*{argmin}_{\mathbf{x}_i} g_i(\mathbf{x}_i^k) + \left\langle \nabla g_i(\mathbf{x}_i^k), \mathbf{x}_i - \mathbf{x}_i^k \right\rangle + h_i(\mathbf{x}_i) + \left\langle \boldsymbol{\lambda}^k, \mathcal{A}_i(\mathbf{x}_i) \right\rangle + \left\langle \beta^{(k)} \mathcal{A}_i^T \left(\mathcal{A}(\mathbf{x}^k) - \mathbf{b} \right), \mathbf{x}_i - \mathbf{x}_i^k \right\rangle + \frac{L_i + \beta^{(k)} \eta_i}{2} \|\mathbf{x}_i - \mathbf{x}_i^k\|^2, \quad (14)$$

where $\eta_i > n||\mathcal{A}_i||^2$ and $\beta^{(k)} > 0$. Note that the subproblem (14) is easy to solve when h_i is nonsmooth but simple. Thus PL-ADMM-PS has much lower per-iteration cost than PALM and Fast PALM. On the other hand, PL-ADMM-PS converges with the rate $O(1/K)$ (Lin, Liu, and Li 2014). However, such a rate is also suboptimal. Now we show that it can be further accelerated by a similar technique as that in Fast PALM. See Algorithm 2 for our Fast PL-ADMM-PS.

Proposition 2. *In Algorithm 2, for any* $\mathbf{x}_i$*, we have*

$$\begin{aligned} &\frac{1-\theta^{(k+1)}}{(\theta^{(k+1)})^2} \left(f_i(\mathbf{x}_i^{k+1}) - f_i(\mathbf{x}_i) \right) \\ &- \frac{1}{\theta^{(k)}} \left\langle \mathcal{A}_i^T(\hat{\boldsymbol{\lambda}}^{k+1}), \mathbf{x}_i - \mathbf{z}_i^{k+1} \right\rangle \\ \leq &\frac{1-\theta^{(k)}}{(\theta^{(k)})^2} \left(f_i(\mathbf{x}_i^k) - f_i(\mathbf{x}_i) \right) \quad (15) \\ &+ \frac{L_i}{2} \left(\|\mathbf{z}_i^k - \mathbf{x}_i\|^2 - \|\mathbf{z}_i^{k+1} - \mathbf{x}_i\|^2 \right) \\ &+ \frac{\beta^{(k)} \eta_i}{2\theta^{(k)}} \left(\|\mathbf{z}_i^k - \mathbf{x}_i\|^2 - \|\mathbf{z}_i^{k+1} - \mathbf{x}_i\|^2 - \|\mathbf{z}_i^{k+1} - \mathbf{z}_i^k\|^2 \right), \end{aligned}$$

where $\hat{\boldsymbol{\lambda}}^{k+1} = \boldsymbol{\lambda}^k + \beta^{(k)} \left(\mathcal{A}(\mathbf{z}^k) - \mathbf{b} \right)$.

Theorem 2. *In Algorithm 2, for any* $K > 0$*, we have*

$$\begin{aligned} &f(\mathbf{x}^{K+1}) - f(\mathbf{x}^*) + \left\langle \boldsymbol{\lambda}^*, \mathcal{A}(\mathbf{x}^{K+1}) - \mathbf{b} \right\rangle \\ &+ \frac{\beta\alpha}{2} \left\| \mathcal{A}(\mathbf{x}^{K+1}) - \mathbf{b} \right\|^2 \quad (16) \\ \leq &\frac{2L_{\max} D_{\mathbf{x}^*}^2}{(K+2)^2} + \frac{2\beta\eta_{\max} D_X^2}{K+2} + \frac{2D_\Lambda^2}{\beta(K+2)}, \end{aligned}$$

where $\alpha = \min\left\{ \frac{1}{n+1}, \left\{ \frac{\eta_i - n\|\mathcal{A}_i\|^2}{2(n+1)\|\mathcal{A}_i\|^2}, i = 1, \cdots, n \right\} \right\}$, $L_{\max} = \max\{L_i, i = 1, \cdots, n\}$ *and* $\eta_{\max} = \max\{\eta_i, i = 1, \cdots, n\}$.

From Theorem 2, it can be seen that our Fast PL-ADMM-PS *partially* accelerates the convergence rate of PL-ADMM-PS from $O\left(\frac{L_{\max} D_{\mathbf{x}^*}^2}{K} + \frac{\beta\eta_{\max} D_{\mathbf{x}^*}^2}{K} + \frac{D_{\boldsymbol{\lambda}^*}^2}{\beta K} \right)$ to $O\left(\frac{L_{\max} D_{\mathbf{x}^*}^2}{K^2} + \frac{\beta\eta_{\max} D_X^2}{K} + \frac{D_\Lambda^2}{\beta K} \right)$. Although the improved rate is also $O(1/K)$, what makes it more attractive is that it allows very large Lipschitz constants L_i's. In particular, L_i can be as large as $O(K)$, without affecting the rate of convergence (up to a constant factor). The above improvement is the same as fast ADMMs (Ouyang et al. 2015) for problem (2) with only $n = 2$ blocks. But it is inferior to the Fast PALM over PALM. The key difference is that Fast PL-ADMM-PS further linearizes the augmented term $\frac{1}{2}||\mathcal{A}(\mathbf{x}) - \mathbf{b}||^2$. This improves the efficiency for solving the subproblem, but slows down the convergence. Actually, when linearizing the augmented term, we have a new term with the factor $\beta^{(k)}\eta_i/\theta^{(k)}$ in (15) (compared with (12) in Fast PALM). Thus (16) has a new term by comparing with that in (13). This makes the choice of $\beta^{(k)}$ in Fast PL-ADMM-PS different from the one in Fast PALM. Intuitively, it can be seen that a larger value of $\beta^{(k)}$ will increase the second terms of (16) and decrease the third term of (16). Thus $\beta^{(k)}$ should be fixed in order to guarantee the convergence. This is different from the choice of $\beta^{(k)}$ in Fast PALM which is adaptive to the choice of $\theta^{(k)}$.

Initialize: $\mathbf{x}^0$, $\mathbf{z}^0$, $\boldsymbol{\lambda}^0$, $\theta^{(0)} = 1$, fix $\beta^{(k)} = \beta$ for $k \geq 0$, $\eta_i > n\|\mathcal{A}_i\|^2$, $i = 1, \cdots, n$,

for $k = 0, 1, 2, \cdots$ **do**

// Update $\mathbf{y}_i$, $\mathbf{z}_i$, $\mathbf{x}_i$, $i = 1, \cdots, n$, in parallel by

$$\mathbf{y}_i^{k+1} = (1 - \theta^{(k)})\mathbf{x}_i^k + \theta^{(k)}\mathbf{z}_i^k; \quad (17)$$

$$\mathbf{z}_i^{k+1} = \operatorname*{argmin}_{\mathbf{x}_i} \left\langle \nabla g_i(\mathbf{y}_i^{k+1}), \mathbf{x}_i \right\rangle + h_i(\mathbf{x}_i) + \left\langle \boldsymbol{\lambda}^k, \mathcal{A}_i(\mathbf{x}_i) \right\rangle + \left\langle \beta^{(k)} \mathcal{A}_i^T \left(\mathcal{A}(\mathbf{z}^k) - \mathbf{b} \right), \mathbf{x}_i \right\rangle + \frac{L(g_i)\theta^{(k)} + \beta^{(k)}\eta_i}{2} \|\mathbf{x}_i - \mathbf{z}_i^k\|^2; \quad (18)$$

$$\mathbf{x}_i^{k+1} = (1 - \theta^{(k)})\mathbf{x}_i^k + \theta^{(k)}\mathbf{z}_i^{k+1}; \quad (19)$$

$$\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \beta^k \left(\mathcal{A}(\mathbf{z}^{k+1}) - \mathbf{b} \right); \quad (20)$$

$$\theta^{(k+1)} = \frac{-(\theta^{(k)})^2 + \sqrt{(\theta^{(k)})^4 + 4(\theta^{(k)})^2}}{2}. \quad (21)$$

end

Algorithm 2: Fast PL-ADMM-PS Algorithm

Compared with PL-ADMM-PS, our Fast PL-ADMM-PS achieves a better rate, but with the price on the boundedness of the feasible primal set X and the feasible dual set Λ. Note that many previous work, e.g., (He and Yuan 2012; Azadi and Sra 2014), also require such a boundedness assumption when proving the convergence of ADMMs. In the following, we give some conditions which guarantee such a boundedness assumption.

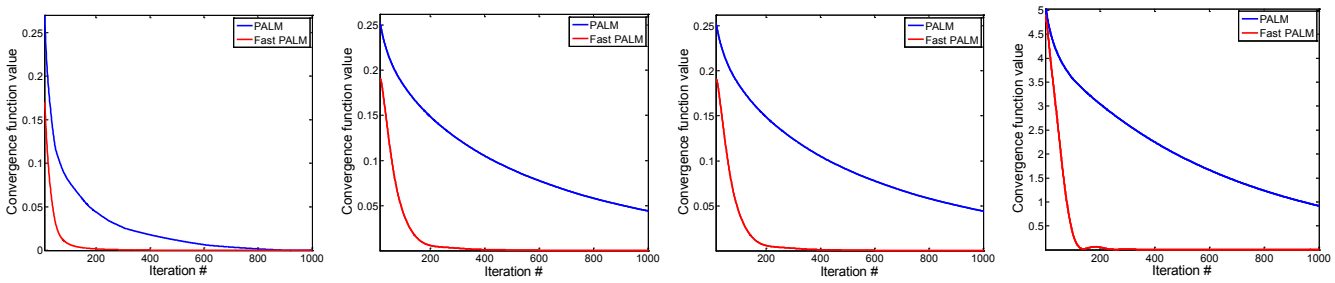
Theorem 3. *Assume the mapping* $\mathcal{A}(\mathbf{x}_1, \cdots, \mathbf{x}_n) = \sum_{i=1}^n \mathcal{A}_i(\mathbf{x}_i)$ *is onto*[4]*, the sequence* $\{\mathbf{z}^k\}$ *is bounded,* $\partial h(\mathbf{x})$ *and* $\nabla g(\mathbf{x})$ *are bounded if* $\mathbf{x}$ *is bounded, then* $\{\mathbf{x}^k\}$*,* $\{\mathbf{y}^k\}$ *and* $\{\boldsymbol{\lambda}^k\}$ *are bounded.*

Many convex functions, e.g., the ℓ_1-norm, in compressed sensing own the bounded subgradient.

Experiments

In this section, we report some numerical results to demonstrate the effectiveness of our fast PALM and PL-ADMM-PS. We first compare our Fast PALM which owns the optimal convergence rate $O(1/K^2)$ with the basic PALM on a

[4]This assumption is equivalent to that the matrix $A \equiv (A_1, \cdots, A_n)$ is of full row rank, where A_i is the matrix representation of $\mathcal{A}_i$.



(a) $m = 100, n = 300$ (b) $m = 300, n = 500$ (c) $m = 500, n = 800$ (d) $m = 800, n = 1000$

Figure 1: Plots of the convergence function values of (13) in each iterations by using PALM and Fast PALM for (22) with different sizes of $\mathbf{A} \in \mathbb{R}^{m\times n}$.

problem with only one block of variable. Then we conduct two experiments to compare our Fast PL-ADMM-PS with PL-ADMM-PS on two multi-blocks problems. The first one is tested on the synthesized data, while the second one is for subspace clustering tested on the real-world data. We examine the convergence behaviors of the compared methods based on the convergence functions shown in (13) and (16). All the numerical experiments are run on a PC with 8 GB of RAM and Intel Core 2 Quad CPU Q9550.

Comparison of PALM and Fast PALM

We consider the following problem

$$\min_{\mathbf{x}} ||\mathbf{x}||_1 + \frac{\alpha}{2}||\mathbf{A}\mathbf{x} - \mathbf{b}||_2^2, \ \text{s.t.} \ \mathbf{1}^T\mathbf{x} = 1, \tag{22}$$

where $\alpha > 0$, $\mathbf{A} \in \mathbb{R}^{m\times n}$, $\mathbf{b} \in \mathbb{R}^m$, and $\mathbf{1} \in \mathbb{R}^n$ is the all one vector. There may have many fast solvers for problem (22). In this experiment, we focus on the performance comparison of PALM and Fast PALM for (22). Note that the per-iteration cost of these two methods are the same. Both of them requires solving an ℓ_1-minimization problem in each iteration. In this work, we use the SPAMS package (Mairal et al. 2010) to solve it which is very fast.

The data matrix $\mathbf{A} \in \mathbb{R}^{m\times n}$, and $\mathbf{b} \in \mathbb{R}^m$ are generated by the Matlab command `randn`. We conduct four experiments on different sizes of $\mathbf{A}$ and $\mathbf{b}$. We use the left hand side of (13) as the convergence function to evaluate the convergence behaviors of PALM and Fast PALM. For the saddle point $(\mathbf{x}^*, \boldsymbol{\lambda}^*)$ in (13), we run the Fast PALM with 10,000 iterations and use the obtained solution as the saddle point. Figure 1 plots the convergence functions value within 1,000 iterations. It can be seen that our Fast PALM converges much faster than PALM. Such a result verifies our theoretical improvement of Fast PALM with optimal rate $O(1/K^2)$ over PALM with the rate $O(1/K)$.

Comparison of PL-ADMM-PS and Fast PL-ADMM-PS

In this subsection, we conduct a problem with three blocks of variables as follows

$$\begin{aligned} \min_{\mathbf{X}_1,\mathbf{X}_2,\mathbf{X}_3} \ & \sum_{i=1}^{3}\left(||\mathbf{X}_i||_{\ell_i} + \frac{\alpha_i}{2}||\mathbf{C}_i\mathbf{X}_i - \mathbf{D}_i||_F^2\right), \\ \text{s.t.} \ & \sum_{i=1}^{3}\mathbf{A}_i\mathbf{X}_i = \mathbf{B}, \end{aligned} \tag{23}$$

where $||\cdot||_{\ell_1} = ||\cdot||_1$ is the ℓ_1-norm, $||\cdot||_{\ell_2} = ||\cdot||_*$ is the nuclear norm, and $||\cdot||_{\ell_3} = ||\cdot||_{2,1}$ is the $\ell_{2,1}$-norm defined as the sum of the ℓ_2-norm of each column of a matrix. We simply consider all the matrices with the same size $\mathbf{A}_i, \mathbf{C}_i, \mathbf{D}_i, \mathbf{B}, \mathbf{X}_i \in \mathbb{R}^{m\times m}$. The matrices $\mathbf{A}_i, \mathbf{C}_i, \mathbf{D}_i, i = 1, 2, 3$, and $\mathbf{B}$ are generated by the Matlab command `randn`. We set the parameters $\alpha_1 = \alpha_2 = \alpha_3 = 0.1$. Problem (23) can be solved by PL-ADMM-PS and Fast PL-ADMM-PS, which have the same and cheap per-iteration cost. The experiments are conducted on three different values of m =100, 300 and 500. Figure 2 plots the convergence function values of PL-ADMM-PS and Fast PL-ADMM-PS in (16). It can be seen that Fast PL-ADMM-PS converges much faster than PL-ADMM-PS. Though Fast PL-ADMM-PS only accelerates PL-ADMM-PS for the smooth parts g_i's, the improvement of Fast PL-ADMM-PS over PL-ADMM-PS is similar to that in Fast PALM over PALM. The reason behind this is that the Lipschitz constants L_i's are not very small (around 400, 1200, and 2000 for the cases $m = 100$, $m = 300$, and $m = 500$, respectively). And thus reducing the first term of (16) faster by our method is important.

Application to Subspace Clustering

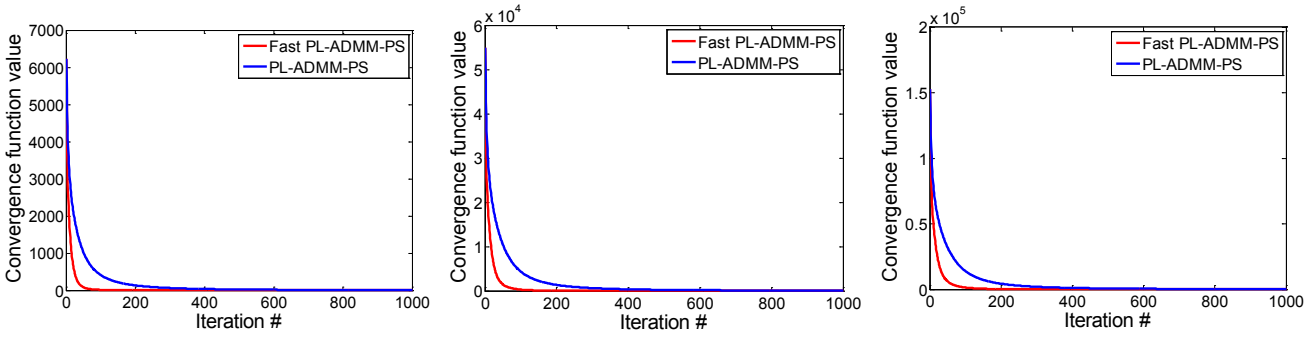
In this subsection, we consider the following low rank and sparse representation problem for subspace clustering

$$\begin{aligned} \min_{\mathbf{Z}} \ & \alpha_1||\mathbf{Z}||_* + \alpha_2||\mathbf{Z}||_1 + \frac{1}{2}||\mathbf{X}\mathbf{Z} - \mathbf{X}||^2, \\ \text{s.t.} \ & \mathbf{1}^T\mathbf{Z} = \mathbf{1}^T, \end{aligned} \tag{24}$$

where $\mathbf{X}$ is the given data matrix. The above model is motivated by (Zhuang et al. 2012). However, we further consider the affine constraint $\mathbf{1}^T\mathbf{Z} = \mathbf{1}^T$ for affine subspace clustering (Elhamifar and Vidal 2013). Problem (24) can be reformulated as a special case of problem (1) by introducing auxiliary variables. Then it can be solved by PL-ADMM-PS and Fast PL-ADMM-PS.

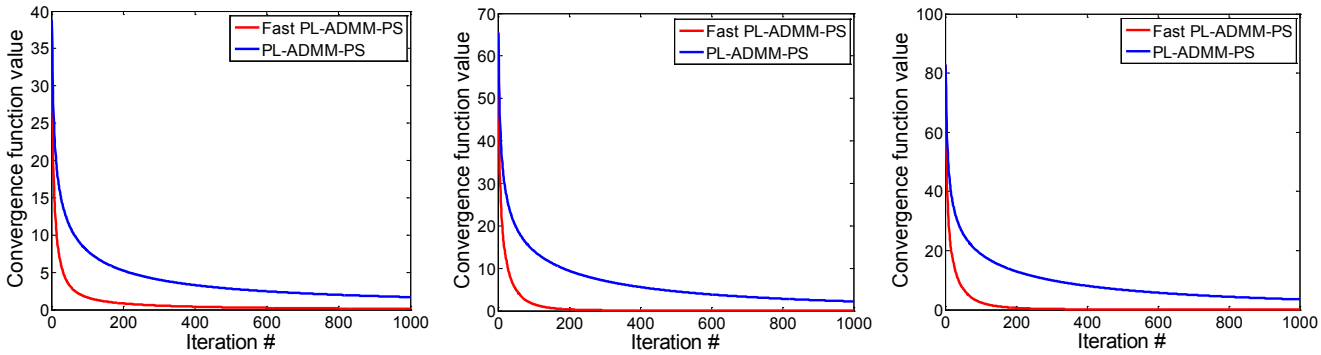
Given a data matrix $\mathbf{X}$ with each column as a sample, we solve (24) to get the optimal solution $\mathbf{Z}^*$. Then the affinity matrix $\mathbf{W}$ is defined as $\mathbf{W} = (|\mathbf{Z}| + |\mathbf{Z}^T|)/2$. The normalized cut algorithm (Shi and Malik 2000) is then performed on $\mathbf{W}$ to get the clustering results of the data matrix $\mathbf{X}$. The whole clustering algorithm is the same as (Elhamifar and Vidal 2013), but using our defined affinity matrix $\mathbf{W}$ above.

We conduct experiments on the Extended Yale B database (Georghiades, Belhumeur, and Kriegman 2001), which is



(a) $m = 100$ (b) $m = 300$ (c) $m = 500$

Figure 2: Plots of the convergence function values of (16) in each iterations by using PL-ADMM-PS and Fast PL-ADMM-PS for (23) with different sizes of $\mathbf{X} \in \mathbb{R}^{m\times m}$.



(a) 5 subjects (b) 8 subjects (c) 10 subjects

Figure 3: Plots of the convergence function values of (16) in each iterations by using PL-ADMM-PS and Fast PL-ADMM-PS for (24) with different sizes of data $\mathbf{X}$ for subspace clustering.

Table 2: Comparision of subspace clustering accuracies (%) on the Extended Yale B database.

Methods	5 subjects	8 subjects	10 subjects
PL-ADMM-PS	94.06	85.94	75.31
Fast PL-ADMM-PS	96.88	90.82	75.47

challenging for clustering. It consists of 2,414 frontal face images of 38 subjects under various lighting, poses and illumination conditions. Each subject has 64 faces. We construct three matrices $\mathbf{X}$ based on the first 5, 8 and 10 subjects. The data matrices $\mathbf{X}$ are first projected into a 5×6, 8×6, and 10×6-dimensional subspace by PCA, respectively. Then we run PL-ADMM-PS and Fast PL-ADMM-PS for 1000 iterations, and use the solutions $\mathbf{Z}$ to define the affinity matrix $\mathbf{W} = (|\mathbf{Z}| + |\mathbf{Z}^T|)/2$. Finally, we can obtain the clustering results by normalized cuts. The accuracy, calculated by the best matching rate of the predicted label and the ground truth of data, is reported to measure the performance. Table 2 shows the clustering accuracies based on the solutions to problem (24) obtained by PL-ADMM-PS and Fast PL-ADMM-PS. It can be seen that Fast PL-ADMM-PS usually outperfoms PL-ADMM-PS since it achieves a better solution than PL-ADMM-PS within 1000 iterations. This can be verified in Figure 3 which shows the convergence function values in (16) of PL-ADMM-PS and Fast PL-ADMM-PS in each iteration. It can be seen that our Fast PL-ADMM-PS converges much faster than PL-ADMM-PS.

Conclusions

This paper presented two fast solvers for the linearly constrained convex problem (1). In particular, we proposed the Fast Proximal Augmented Lagragian Method (Fast PALM) which achieves the convergence rate $O(1/K^2)$. Note that such a rate is theoretically optimal by comparing with the rate $O(1/K)$ by traditional ALM/PALM. Our fast version does not require additional assumptions (e.g. boundedness of X and Λ, or a predefined number of iterations) as in the previous works (Azadi and Sra 2014; Ouyang et al. 2015). In order to further reduce the per-iteration complexity and handle the multi-blocks problems ($n > 2$), we proposed the Fast Proximal Linearized ADMM with Parallel Splitting (Fast PL-ADMM-PS). It also achieves the optimal $O(1/K^2)$ rate for the smooth part of the objective. Compared with PL-ADMM-PS, though Fast PL-ADMM-PS requires additional assumptions on the boundedness of X and Λ in theory, our experimental results show that significant improvements are obtained especially when the Lipschitz constant of the smooth part is relatively large.

Acknowledgements

This research is supported by the Singapore National Research Foundation under its International Research Centre @Singapore Funding Initiative and administered by the IDM Programme Office. Zhouchen Lin is supported by National Basic Research Program of China (973 Program) (grant no. 2015CB352502), National Natural Science Foundation (NSF) of China (grant nos. 61272341 and 61231002), and Microsoft Research Asia Collaborative Research Program.

References

Azadi, S., and Sra, S. 2014. Towards an optimal stochastic alternating direction method of multipliers. In *ICML*, 620–628.

Beck, A., and Teboulle, M. 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*.

Boyd, S.; Parikh, N.; Chu, E.; Peleato, B.; and Eckstein, J. 2011. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine Learning* 3(1):1–122.

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

Candès, E. J.; Li, X. D.; Ma, Y.; and Wright, J. 2011. Robust principal component analysis? *Journal of the ACM* 58(3).

Chandrasekaran, V.; Parrilo, P. A.; and Willsky, A. S. 2012. Latent variable graphical model selection via convex optimization. 40(4):1935–1967.

Douglas, J., and Rachford, H. H. 1956. On the numerical solution of heat conduction problems in two and three space variables. *Transactions of the American mathematical Society* 421–439.

Elhamifar, E., and Vidal, R. 2013. Sparse subspace clustering: Algorithm, theory, and applications. *TPAMI* 35(11):2765–2781.

Georghiades, A. S.; Belhumeur, P. N.; and Kriegman, D. 2001. From few to many: Illumination cone models for face recognition under variable lighting and pose. *PAMI* 23(6):643–660.

Goldstein, T.; O'Donoghue, B.; Setzer, S.; and Baraniuk, R. 2014. Fast alternating direction optimization methods. *SIAM Journal on Imaging Sciences* 7(3):1588–1623.

He, B., and Yuan, X. 2010. On the acceleration of augmented Lagrangian method for linearly constrained optimization. *optimization online www. optimization-online. org/DB FILE/2010/10/2760. pdf*.

He, B., and Yuan, X. 2012. On the $O(1/n)$ convergence rate of the Douglas-Rachford alternating direction method. *SIAM Journal on Numerical Analysis* 50(2):700–709.

Hestenes, M. R. 1969. Multiplier and gradient methods. *Journal of optimization theory and applications* 4(5):303–320.

Jacob, L.; Obozinski, G.; and Vert, J.-P. 2009. Group Lasso with overlap and graph Lasso. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 433–440. ACM.

Lin, Z.; Liu, R.; and Li, H. 2014. Linearized alternating direction method with parallel splitting and adaptive penalty for separable convex programs in machine learning. In *Machine Learning*.

Lin, Z.; Liu, R.; and Su, Z. 2011. Linearized alternating direction method with adaptive penalty for low-rank representation. In *NIPS*, volume 2, 6.

Liu, G., and Yan, S. 2011. Latent low-rank representation for subspace segmentation and feature extraction. In *ICCV*, 1615–1622. IEEE.

Lu, C.-Y.; Min, H.; Zhao, Z.-Q.; Zhu, L.; Huang, D.-S.; and Yan, S. 2012. Robust and efficient subspace segmentation via least squares regression. In *ECCV*, 347–360. Springer.

Lu, C.; Feng, J.; Lin, Z.; and Yan, S. 2013. Correlation adaptive subspace segmentation by trace Lasso. In *CVPR*. IEEE.

Mairal, J.; Bach, F.; Ponce, J.; and Sapiro, G. 2010. Online learning for matrix factorization and sparse coding. *JMLR* 11:19–60.

Nesterov, Y. 1983. A method for unconstrained convex minimization problem with the rate of convergence $O(1/k^2)$. In *Doklady AN SSSR*, volume 269, 543–547.

Nesterov, Y. 2004. *Introductory lectures on convex optimization: A basic course*, volume 87. Springer.

Ouyang, Y.; Chen, Y.; Lan, G.; and Pasiliao Jr, E. 2015. An accelerated linearized alternating direction method of multipliers. *SIAM Journal on Imaging Sciences* 8(1):644–681.

Shi, J., and Malik, J. 2000. Normalized cuts and image segmentation. *PAMI* 22(8):888–905.

Tao, M. 2014. Some parallel splitting methods for separable convex program-ming with $O(1/t)$ convergence rate. *Pacific Journal of Optimization* 10:359–384.

Tibshirani, R. 1996. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)* 267–288.

Tseng, P. 2008. On accelerated proximal gradient methods for convex-concave optimization. *in submission*.

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

Zhuang, L.; Gao, H.; Lin, Z.; Ma, Y.; Zhang, X.; and Yu, N. 2012. Non-negative low rank and sparse graph for semi-supervised learning. In *CVPR*, 2328–2335. IEEE.