Faster and Non-ergodic $O(1/K)$ Stochastic Alternating Direction Method of Multipliers

Cong Fang  Feng Cheng  Zhouchen Lin*
Key Laboratory of Machine Perception (MOE), School of EECS, Peking University, P. R. China
Cooperative Medianet Innovation Center, Shanghai Jiao Tong University, P. R. China
fangcong@pku.edu.cn  fengcheng@pku.edu.cn  zlin@pku.edu.cn

Abstract

We study stochastic convex optimization subjected to linear equality constraints. Traditional Stochastic Alternating Direction Method of Multipliers [1] and its Nesterov’s acceleration scheme [2] can only achieve ergodic $O(1/\sqrt{K})$ convergence rates, where $K$ is the number of iteration. By introducing Variance Reduction (VR) techniques, the convergence rates improve to ergodic $O(1/K)$ [3,4]. In this paper, we propose a new stochastic ADMM which elaborately integrates Nesterov’s extrapolation and VR techniques. With Nesterov’s extrapolation, our algorithm can achieve a non-ergodic $O(1/K)$ convergence rate which is optimal for separable linearly constrained non-smooth convex problems, while the convergence rates of VR based ADMM methods are actually tight $O(1/\sqrt{K})$ in non-ergodic sense. To the best of our knowledge, this is the first work that achieves a truly accelerated, stochastic convergence rate for constrained convex problems. The experimental results demonstrate that our algorithm is faster than the existing state-of-the-art stochastic ADMM methods.

1 Introduction

We consider the following general convex finite-sum problem with linear constraints:

$$
\min_{x_1, x_2} \quad h_1(x_1) + f_1(x_1) + h_2(x_2) + \frac{1}{n} \sum_{i=1}^{n} f_{2,i}(x_2),
$$

s.t. $A_1 x_1 + A_2 x_2 = b$,  \hspace{1cm} (1)

where $f_1(x_1)$ and $f_{2,i}(x_2)$ with $i \in \{1, 2, \ldots, n\}$ are convex and have Lipschitz continuous gradients, $h_1(x_1)$ and $h_2(x_2)$ are also convex, but can be non-smooth. We use the following notations: $L_1$ denotes the Lipschitz constant of $f_1(x_1)$, $L_2$ is the Lipschitz constant of $f_{2,i}(x_2)$ with $i \in \{1, 2, \ldots, n\}$, and $f_2(x) = \frac{1}{n} \sum_{i=1}^{n} f_{2,i}(x)$. And we use $\nabla f$ to denote the gradient of $f$.

Problem (1) is of great importance in machine learning. The finite-sum functions $f_2(x_2)$ are typically a loss over training samples, and the remaining functions control the structure or regularize the model to aid generalization [2]. The idea of using linear constraints to decouple the loss and regularization terms enables researchers to consider some more sophisticated regularized terms which might be very complicated to solve through proximity operators for Gradient Descent [5] methods. For example, for multitask learning problems [6,7], the regularization term is set as $\mu_1 \|x\|_2 + \mu_2 \|x\|_1$, for most graph-guided fused Lasso and overlapping group Lasso problem [8,4], the regularization term can be written as $\mu_1 \|Ax\|_1$, and for many multi-view learning tasks [9], the regularization terms always involve $\mu_1 \|x\|_{2,1} + \mu_2 \|x\|_1$.

*Corresponding author.
Alternating Direction Method of Multipliers (ADMM) is a very popular optimization method to solve convex finite-sum problems with linear constraints. By elaborately integrating Nesterov’s extrapolation inspired by \([10]\) who proposes an alternating optimization algorithm named Katyusha for convex nonsmooth convex problems is exactly \(O(1)\). The lower complexity bound of ADMM type methods for the separable linearly constrained stochastic gradient algorithm named Katyusha for convex nonsmooth convex problems is exactly \(O(1)\), which demonstrates that their algorithm is optimal. The convergence rates for different ADMM based methods are shown in Table 1.

On the other hand, to meet the demands of solving large-scale machine learning problems, stochastic algorithms \([13]\) have drawn a lot of interest in recent years. For stochastic ADMM (SADMM), the prior works are from STOC-ADMM \([11]\) and OPG-ADMM \([16]\). Due to the noise of gradient, both of the two algorithms can only achieve an ergodic \(O(1/\sqrt{K})\) convergence rate. There are two lines of research to accelerate SADMM. The first is to introduce the Variance Reduction (VR) \([19,20,21]\) techniques into SADMM. VR methods ensure the descent direction to have a bounded variance and so can achieve faster convergence rates. The existing VR based SADMM algorithms include SDCA-ADMM \([17]\), SAG-ADMM \([3]\) and SVRG-ADMM \([4]\). SAG-ADMM and SVRG-ADMM can provably achieve ergodic \(O(1/K)\) rates for Problem \(1\). The second way to accelerate SADMM is through the Nesterov’s acceleration \([22]\). This work is from \([2]\), in which the authors propose an ergodic \(O(n_2^2 + D_2 + L + \sqrt{L})\) stochastic algorithm (OPT-ADMM). The dependence on the smoothness constant of the convergence rate is \(O(1/K^2)\) and so each term in the convergence rate seems to have been improved to optimal. However, the worst convergence rate of it is still \(O(1/\sqrt{K})\).

In this paper, we propose Accelerated Stochastic ADMM (ACC-SADMM) for large scale general convex finite-sum problems with linear constraints. By elaborately integrating Nesterov’s extrapolation and VR techniques, ACC-SADMM provably achieves an ergodic \(O(1/K)\) convergence rate which is optimal for non-smooth problems. As in non-ergodic sense, the VR based SADMM methods (e.g. SVRG-ADMM, SAG-ADMM) converges in a tight \(O(1/\sqrt{K})\) (please see detailed discussions in Section 5.3), ACC-SADMM improve the convergence rates from \(O(1/\sqrt{K})\) to \((1/K)\) in the ergodic sense and fill the theoretical gap between the stochastic and batch (deterministic) ADMM. The original idea to design our ACC-SADMM is by explicitly considering the snapshot vector \(\tilde{x}\) (approximately the mean value of \(x\) in the last epoch) into the extrapolation terms. This is, to some degree, inspired by \([23]\) who proposes an \(O(1/K^2)\) stochastic gradient algorithm named Katyusha for convex
problems. However, there are many distinctions between the two algorithms (please see detailed discussions in Section 12). Our method is also very efficient in practice since we have sufficiently considered the noise of gradient into our acceleration scheme. For example, we adopt extrapolation as \(y_s = x_s^k + (1 - \theta_{1,s} - \theta_2)(x_s^k - x_s^{k-1})\) in the inner loop, where \(\theta_2\) is a constant and \(\theta_{1,s}\) decreases after every epoch, instead of directly adopting extrapolation as \(y^k = x^k + \frac{\theta^k(1 - \theta_{1,s}^k - 1)}{\theta_{1,s}^k}(x^k - x^{k-1})\) in the original Nesterov’s scheme and adding proximal term \(\frac{\|x^{k+1} - x^k\|^2}{\theta_{1,s}^k}\) as [2] does. There are also variants on updating of multiplier and the snapshot vector. We list the contributions of our work as follows:

- We propose ACC-SADMM for large scale convex finite-sum problems with linear constraints which integrates Nesterov’s extrapolation and VR techniques. We prove that our algorithm converges in non-ergodic \(O(1/K)\) which is optimal for separable linearly constrained non-smooth convex problems. To our best knowledge, this is the first work that achieves a truly accelerated, stochastic convergence rate for constrained convex problems.
- We do experiments on four bench-mark datasets to demonstrate the superiority of our algorithm. We also do experiments on the Multitask Learning [6] problem to demonstrate that our algorithm can be used on very large datasets.

### 2 Preliminary

Most SADMM methods alternate minimizes the following variant surrogate of the augmented Lagrangian:

\[
L'(x_1, x_2, \lambda, \beta) = \langle \nabla f_1(x_1), x_1 \rangle + \frac{L_1}{2} \|x_1 - x^k_1\|^2_{G_i} + h_2(x_2) + \frac{L_2}{2} \|x_2 - x^k_2\|^2_{G_2} + \frac{\beta}{2} \|A_1 x_1 + A_2 x_2 - b + \lambda\|^2_{\beta},
\]

where \(\nabla f_2(x_2)\) is an estimator of \(\nabla f_2(x_2)\) from one or a mini-batch of training samples. So the computation cost for each iteration reduces from \(O(n)\) to \(O(b)\) instead, where \(b\) is the mini-batch size. When \(f_i(x) = 0\) and \(G_i = 0\), with \(i = 1, 2\), Problem (1) is solved as exact ADMM. When there is no \(h_i(x_i)\), \(G_i\) is set as the identity matrix \(I\), with \(i = 1, 2\), the subproblem in \(x_i\) can be solved through matrix inversion. This scheme is advocated in many SADMM methods [1][4]. Another common approach is linearization (also called the inexact Uzawa method) [24][25], where \(G_i\) is set as \(\eta_i I - \frac{\beta}{\eta_i} A_i^T A_i\) with \(\eta_i \geq 1 + \frac{\beta}{L_i} \|A_i^T A_i\|\).

For STOC-ADMM [1], \(\nabla f_2(x_2)\) is simply set as:

\[
\nabla f_2(x_2) = \frac{1}{b} \sum_{i_k \in I_k} \nabla f_{2,i_k}(x_2),
\]

where \(I_k\) is the mini-batch of size \(b\) from \(\{1, 2, \cdots, n\}\). For SVRG-ADMM [4], the gradient estimator can be written as:

\[
\nabla f_2(x_2) = \frac{1}{b} \sum_{i_k \in I_k} (\nabla f_{2,i_k}(x_2) - \nabla f_{2,i_k}(\bar{x}_2)) + \nabla f_2(\bar{x}_2),
\]

where \(\bar{x}_2\) is a snapshot vector (mean value of last epoch).
Algorithm 1 Inner loop of ACC-SADMM

for $k = 0$ to $m-1$ do
    Update dual variable: $\lambda^k_s = \tilde{\lambda}^k_s + \frac{\theta_2}{\theta_1,s} \left( A_1 x^k_{s,1} + A_2 x^k_{s,2} - \tilde{b}_s \right)$.
    Update $x^k_{s,1}$ through Eq. (6).
    Update $x^k_{s,2}$ through Eq. (7).
    Update dual variable: $\tilde{\lambda}^{k+1}_s = \lambda^k_s + \beta \left( A_1 x^{k+1}_{s,1} + A_2 x^{k+1}_{s,2} - b \right)$.
    Update $y^{k+1}_s$ through Eq. (5).
end for

3 Our Algorithm

3.1 ACC-SADMM

To help readers easier understand our algorithm, we list the notations and the variables in Table 2. Our algorithm has double loops as we use SVRG [19], which also have two layers of nested loops to estimate the gradient. We denote subscript $s$ as the index of the outer loop and superscript $k$ as the index in the inner loops. For example, $x^k_{s,1}$ is the value of $x_1$ at the $k$-th step of the inner iteration and the $s$-th step of the outer iteration. And we use $x^k_s$ and $y^k_s$ to denote $(x^k_{s,1}, x^k_{s,2})$, and $(y^k_{s,1}, y^k_{s,2})$, respectively. In each inner loop, we update primal variables $x^k_{s,1}$ and $x^k_{s,2}$, extrapolation terms $y^k_{s,1}$, $y^k_{s,2}$ and dual variable $\lambda^k_s$, and $s$ remains unchanged. In the outer loop, we maintain snapshot vectors $\tilde{x}_{s+1,1}$, $\tilde{x}_{s+1,2}$ and $\tilde{b}_{s+1}$, and then assign the initial value to the extrapolation terms $y^0_{s+1,1}$ and $y^0_{s+1,2}$. We directly linearize both the smooth term $f_i(x_i)$ and the augmented term $\frac{\beta}{2} \lVert A_1 x_1 + A_2 x_2 - b + \lambda \rVert^2$. The whole algorithm is shown in Algorithm 1.

3.2 Inner Loop

The inner loop of ACC-SADMM is straightforward, shown as Algorithm 1. In each iteration, we do extrapolation, and then update the primal and dual variables. There are two critical steps which ensures us to obtain a non-ergodic results. The first is extrapolation. We do extrapolation as:

$$
\tilde{y}^{k+1}_s = y^k_s + (1 - \theta_1,s - \theta_2) (y^k_s - x^k_s),
$$

We can find that $1 - \theta_1,s - \theta_2 \leq 1 - \theta_1,s$. So comparing with original Nesterov’s scheme, our way is more “mild” to tackle the noise of gradient. The second step is on the updating primal variables:

$$
x^k_{s,1} = \text{argmin}_{x_1} h_1(x_1) + \left( \nabla f_1(y^k_{s,1}), x_1 \right)
+ \frac{\beta}{\theta_1,s} \left( A_1 y^k_{s,1} + A_2 y^k_{s,2} - b \right) + \lambda^k_s, A_1 x_1 \right) + \left( \frac{L_1}{2} + \frac{\beta \| A_1^T A_1 \|}{2 \theta_1,s} \right) \| x_1 - y^k_{s,1} \|^2.
$$

And then update $x_2$ with the latest information of $x_1$, which can be written as:

$$
x^k_{s,2} = \text{argmin}_{x_2} h_2(x_2) + \left( \nabla f_2(y^k_{s,2}), x_2 \right) + \left( \frac{\beta}{\theta_1,s} \left( A_1 x^k_{s,1} + A_2 y^k_{s,2} - b \right) \right)
+ \lambda^k_s, A_2 x_2 \right) + \left( \frac{1}{2} + \frac{\beta \| A_2^T A_2 \|}{2 \theta_1,s} \right) \| x_2 - y^k_{s,2} \|^2,
$$

where $\tilde{\nabla} f_2(y^k_{s,2})$ is obtained by the technique of SVRG [19] with the form:

$$
\tilde{\nabla} f_2(y^k_{s,2}) = \frac{1}{b} \sum_{i_k,s \in I_{k,s}} \left( \nabla f_2, i_{k,s} (y^k_{s,2}) - \nabla f_2, i_{k,s} (\tilde{x}_{s,2}) + \nabla f_2 (\tilde{x}_{s,2}) \right).
$$

Comparing with unaccelerated SADMM methods, which alternately minimize Eq. (2), our method is distincted in two ways. The first is that the gradient estimator are computed on the $y^k_{s,2}$. The second is that we have chosen a slower increasing penalty factor $\frac{\beta}{\theta_1,s}$, instead of a fixed one.
Algorithm 2 ACC-SADMM

Input: epoch length $m > 2$, $\beta$, $\tau = 2$, $c = 2$, $x^0_0 = 0$, $\lambda^0_0 = 0$, $\bar{x}^0 = x^0_0$, $y^0 = x^0_0$,
$\theta_1,s = \frac{1}{c+s}$, $\theta_2 = \frac{m-\tau}{\tau(m-1)}$.

for $s = 0$ to $S - 1$ do
  Do inner loop, as stated in Algorithm 1
  Set primal variables: $x^0_{s+1} = x^m_s$.
  Update snapshot vectors $\bar{x}_{s+1}$ through Eq. (8).
  Update dual variable: $\lambda^0_{s+1} = \lambda^m_{s+1} + \beta(1-\tau)(A_1 x^m_{s+1} + A_2 x^m_{s+2} - b)$.
  Update dual snapshot variable: $b_{s+1} = A_1 \bar{x}_{s+1,1} + A_2 \bar{x}_{s+1,2}$.
  Update extrapolation terms $y^0_{s+1}$ through Eq. (9).
end for $s$.

Output: $\bar{x}_S = \frac{1}{(m-1)(\theta_{1,s} + \theta_2) + 1} x^m_s + \frac{\theta_{1,s} + \theta_2}{(m-1)(\theta_{1,s} + \theta_2) + 1} \sum_{k=1}^{m-1} x^k_S$.

3.3 Outer Loop

The outer loop of our algorithm is a little complex, in which we preserve snapshot vectors, and then resets the initial value. The main variants we adopt is on the snapshot vector $\bar{x}_{s+1}$ and the extrapolation term $y^0_{s+1}$. For the snapshot vector $\bar{x}_{s+1}$, we update it as:

$$\bar{x}_{s+1} = \frac{1}{m} \left[ 1 - \frac{(\tau - 1)\theta_{1,s} + 1}{\theta_2} \right] x^m_s + \left[ 1 + \frac{(\tau - 1)\theta_{1,s} + 1}{(m-1)(\theta_{1,s} + \theta_2)} \sum_{k=1}^{m-1} x^k_s \right]. \tag{8}$$

$\bar{x}_{s+1}$ is not the average of $\{x^k_s\}$ different from most SVRG-based methods [19] [4]. The way of generating $\bar{x}$ guarantees a faster convergence rate for the constraints. Then we reset $y^0_{s+1}$ as:

$$y^0_{s+1} = (1 - \theta_2) x^m_s + \theta_2 \bar{x}_{s+1} + \frac{\theta_{1,s} + 1}{\theta_{1,s} + \theta_2} \left[ (1 - \theta_1,s) x^m_s - (1 - \theta_1,s - \theta_2) x^{m-1} - \theta_2 \bar{x}_s \right]. \tag{9}$$

4 Convergence Analysis

In this section, we give the convergence results of ACC-SADMM. The proof and a outline can be found in Supplementary Material. As we have mentioned in Section 3.2, the main strategy that enable us to obtain a non-ergodic results is that we adopt extrapolation as Eq. (5). We first analyze each inner iteration, shown in Lemma 1. We ignore subscript $s$ as $s$ is unchanged in the inner iteration.

Lemma 1 Assume that $f_i(x_1)$ and $f_{2,i}(x_2)$ with $i \in \{1, 2, \cdots, n\}$ are convex and have Lipschitz continuous gradients. $L_1$ is the Lipschitz constant of $f_1(x_1)$. $L_2$ is the Lipschitz constant of $f_{2,i}(x_2)$ with $i \in \{1, 2, \cdots, n\}$. $h_1(x_1)$ and $h_2(x_2)$ is also convex. For Algorithm 2 in any epoch, we have

$$\mathbb{E}_{i,k} \left[ L(x_{k+1}^i, x_{k+1}^2, \lambda^*) - h_2(\bar{x}_1) - (1 - \theta_2 - \theta_1)L(x_1^k, x_2^k, \lambda^*) \right] \leq \frac{\theta_1}{\beta} \left( \left\| \lambda^k - \lambda^* \right\|^2 - \mathbb{E}_{i,k} \left[ \left\| \lambda^k - \lambda^* \right\|^2 \right] \right) + \frac{1}{2} \left\| y_1^k - (1 - \theta_1 - \theta_2) x_1^k \right\|_{\mathbb{G}_3}^2 + \frac{1}{2} \left\| y_2^k - (1 - \theta_1 - \theta_2) x_2^k \right\|_{\mathbb{G}_4}^2.$$

where $\mathbb{E}_{i,k}$ denotes that the expectation is taken over the random samples in the minibatch $\mathcal{I}_{k,s}$, $L(x_1, x_2, \lambda) = f_1(x_1) + f_{2,i}(x_2) + \langle \lambda, A_1 x_1 + A_2 x_2 - b \rangle$ and $\lambda^k = \lambda^k + \frac{\beta(1-\theta_1)}{\theta_1} (A x^k - b)$, $G_3 = \left( L_1 + \frac{\beta \| A^2_1 + A^2_2 \|}{\theta_1} \right) - \frac{\beta \| A^2_1 \|}{\theta_1}$, and $G_4 = \left( (1 + \frac{1}{\theta_2}) L_2 + \frac{\beta \| A^2_2 \|}{\theta_1} \right) I$. Then Theorem 1 analyses ACC-SADMM in the whole iteration, which is the key convergence result of the paper.
Then there exist convex functions $F$ and $\lambda^*$

We discuss some properties of ACC-SADMM and make further comparisons with some related methods.

Theorem 2 is Theorem 15 in [15]. More details can be found in it. Problem (1) as we can set each $F_{2,i}(x_2) = F(x_2)$ with $i = 1, \cdots, n$ or set $n = 1$. So there is no better ADMM type algorithm which converges faster than $O(1/K)$ for Problem (1).

5 Discussions

We discuss some properties of ACC-SADMM and make further comparisons with some related methods.
5.1 Comparison with Katyusha

As we have mentioned in Introduction, some intuitions of our algorithm are inspired by Katyusha [23], which obtains an $O(1/K^2)$ algorithm for convex finite-sum problems. However, Katyusha cannot solve the problem with linear constraints. Besides, Katyusha uses the Nesterov’s second scheme to accelerate the algorithm while our method conducts acceleration through Nesterov’s extrapolation (Nesterov’s first scheme). And our proof uses the technique of [26], which is different from [23]. Our algorithm can be easily extended to unconstrained convex finite-sum and can also obtain a $O(1/K^2)$ rate but belongs to the Nesterov’s first scheme[3].

5.2 The Growth of Penalty Factor $\frac{\beta}{\theta_{1,s}}$

The penalty factor $\frac{\beta}{\theta_{1,s}}$ increases linearly with the iteration. One might deem that this make our algorithm impractical because after dozens of epochs, the large value of penalty factor might slow down the decrement of function value. However, we have not found any bad influence. There may be two reasons 1. In our algorithm, $\theta_{1,s}$ decreases after each epoch ($m$ iterations), which is much slower than LADM-NE [15]. So the growth of penalty factor works as a continuation technique [23], which may help to decrease the function value. 2. From Theorem [1], our algorithm converges in $O(1/S)$ whenever $\theta_{1,s}$ is large. So from the theoretical viewpoint, a large $\theta_{1,s}$ cannot slow down our algorithm. We find that OPT-ADMM [2] also needs to decrease the step size with the iteration. However, its step size decreasing rate is $O(k^2)$ and is faster than ours.

5.3 The Importance of Non-ergodic $O(1/K)$

SAG-ADMM [3] and SVRG-ADMM [4] accelerate SADMM to ergodic $O(1/K)$. In Theorem 9 of [15], the authors generate a class of functions showing that the original ADMM has a tight non-ergodic $O(1/\sqrt{K})$ convergence rate. When $n = 1$, SAG-ADMM and SVRG-ADMM are the same as batch ADMM, so their convergence rates are no better than $O(1/\sqrt{K})$. So in non-ergodic sense, our algorithm does have a faster convergence rate than VR based SADMM methods.

Then we are to highlight the importance of our non-ergodic result. As we have mentioned in the Introduction, in practice, the output of ADMM methods is the non-ergodic result $x^K$, not the mean of $x^1$ to $x^K$. For deterministic ADMM, the proof of ergodic $O(1/K)$ rate is proposed in [11], after ADMM had become a prevailing method of solving machine learning problems [29]; for stochastic ADMM, e.g. SVRG-ADMM [4], the authors give an ergodic $O(1/K)$ proof, but in experiment, what they emphasize to use is the mean value of the last epoch as the result. As the non-ergodic results are more close to reality, our algorithm is much faster than VR based SADMM methods, even when its rate is seemingly the same. Actually, though VR based SADMM methods have provably faster rates than STOC-ADMM, the improvement in practice is evident only after numbers of iterations, when point are close to the convergence point, rather than at the early stage. In both [3] and [4], the authors claim that SAG-ADMM and SVRG-ADMM are sensitive to initial points. We also find that if the step sizes are set based on the their theoretical guidances, sometimes they are even slower than STOC-ADMM (see Fig. 1) as the early stage lasts longer when the step size is small. Our algorithm is faster than the two algorithms which demonstrates that Nesterov’s extrapolation has truly accelerated the speed and the integration of extrapolation and VR techniques is harmonious and complementary.

---

3 We follow [26] to name the extrapolation scheme as Nesterov’s first scheme and the three-step scheme [27] as the Nesterov’s second scheme.
6 Experiments

We conduct experiments to show the effectiveness of our method\(^1\). We compare our method with the following state-of-the-art SADMM algorithms: (1) STOC-ADMM \(^2\), (2) SVRG-ADMM \(^3\), (3) OPT-SADMM \(^4\), (4) SAG-ADMM \(^5\). We ignore SDCA-ADMM \(^6\) in our comparison since it gives no analysis on general convex problems and it is also not faster than SVRG-ADMM \(^4\).

Experiments are performed on Intel(R) CPU i7-4770 @ 3.40GHz machine with 16 GB memory. Our experiments focus on two typical problems: (a) Lasso Problem and Multitask Learning. Due to space limited, the experiment of Multitask Learning is shown in Supplementary Materials. For the Lasso problems, we perform experiments under the following typical variations. The first is the original Lasso problem; and the second is Graph-Guided Fused Lasso model: \(\min_{\mathbf{x}} \frac{1}{n} \sum_{i=1}^{n} l_i(\mathbf{x}), \) where \(l_i(\mathbf{x})\) is the logistic loss on sample \(i\), and \(\mathbf{A} = [\mathbf{G}; \mathbf{I}]\) is a matrix encoding the feature sparsity pattern. \(\mathbf{G}\) is the sparsity pattern of the graph obtained by sparse inverse covariance estimation \(^{[50]}\). The experiments are performed on four benchmark data sets: a9a, covertype, mnist and dna.\(^\dagger\) The details of the dataset and the mini-batch size that we use in all SADMM are shown in Table 3. And like \(^3\) and \(^4\), we fix \(\mu = 10^{-5}\) and report the performance based on \((\mathbf{x}_t, \mathbf{A}\mathbf{x}_t)\) to satisfy the constraints of ADMM. Results are averaged over five repetitions. And we set \(m = \frac{2n}{b}\) for all the algorithms. For original Lasso problem, the step sizes are set through theoretical guidelines for each algorithm. For the Graph-Guided Lasso, the best step sizes are obtained through searches on parameters which give best convergence progress. Except ACC-SADMM, we use the continuation technique \(^{[28]}\) to accelerate algorithms. SAG-ADMM is performed on the first three datasets due to its large memory requirement.

The experimental results are shown in Fig. 1. We can find that our algorithm consistently outperforms other compared methods in all these datasets for both the two problems, which verifies our theoretical analysis. The details about parameter setting, experimental results where we set a larger fixed step size for the group guided Lasso problem, curves of the test error, the memory costs of all algorithms, and Multitask learning experiment are shown in Supplementary Materials.

\(^\dagger\)The code will be available at \url{http://www.cis.pku.edu.cn/faculty/vision/zlin/zlin.htm}
\(^\dagger\)a9a, covertype and dna are from \url{http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/}
and mnist is from: \url{http://yann.lecun.com/exdb/mnist/}
7 Conclusion

We propose ACC-SADMM for the general convex finite-sum problems. ACC-SADMM integrates Nesterov’s extrapolation and VR techniques and achieves a non-ergodic $O(1/K)$ convergence rate, which shows theoretical and practical importance. We do experiments to demonstrate that our algorithm is faster than other SADMM methods.

Acknowledgment

Zhouchen Lin is supported by National Basic Research Program of China (973 Program) (grant no. 2015CB352502) and National Natural Science Foundation (NSF) of China (grant no.s 61625301, 61731018, and 61231002).

References


