Distributed Multiagent Optimization: Linear Convergence Rate of ADMM

Anonymous Author(s)
Affiliation
Address
email

Abstract

We propose a distributed algorithm based on Alternating Direction Method of Multipliers (ADMM) to minimize the sum of locally known convex functions. This optimization problem captures many applications in distributed machine learning and statistical estimation. We provide a novel analysis that shows if the functions are strongly convex and have Lipschitz gradients, then an ϵ-optimal solution can be computed with $O(\sqrt{\kappa_f}\log(1/\epsilon))$ iterations, where κ_f is the condition number of the problem. This is the first paper in the literature that establishes this rate of convergence for distributed ADMM, matching the best known iteration complexity for centralized ADMM. Our analysis also highlights the effect of network structure on the convergence rate.

1 Introduction

Many of today's optimization problems in data science (including statistics, machine learning, and data mining) use distributed computation. Modern applications have access to an abundance of data, which cannot be handled by a single processor alone. This necessitates distributing data among multiple processors and processing it in a decentralized manner based on the available local information. The applications in statistical learning [55, 14, 44, 1, 18] along with other applications in distributed data processing where information is inherently distributed among many processors (see e.g. distributed sensor networks [16, 42], coordination and flow control problems [34, 28]) have spearheaded a large literature on distributed multiagent optimization.

In this paper, we focus on a general multiagent optimization problems as follows

$$\min_{x\in\mathbb{R}^d} \sum_{i=1}^{n} f_i(x), \tag{1}$$

where $f_i : \mathbb{R}^d \to \mathbb{R}$ is a convex function that is known only to agent i referred to as local objective function.[1] Machines can communicate over a given network and their goal is to collectively solve this optimization. A prominent example where this general formulation emerges is *Empirical Risk Minimization* (EMR). Suppose that we have M data points $\{(x_i, y_i)\}_{i=1}^{M}$, where $x_i \in \mathbb{R}^d$ is a feature vector and $y_i \in \mathbb{R}$ is a target output. The empirical risk minimization is then given by

$$\min_{\theta\in\mathbb{R}^d} \frac{1}{M}\sum_{i=1}^{M} L(y_i, x_i, \theta) + p(\theta), \tag{2}$$

for some convex loss function $L : \mathbb{R}\times\mathbb{R}^d\times\mathbb{R}^d \to \mathbb{R}$ and some convex penalty function $p : \mathbb{R}^d \to \mathbb{R}$. This general formulation captures many statistical scenarios including:

[1] We use the terms machine, agent, and node interchangeably.

- Least-Absolute Shrinkage and Selection Operator (LASSO): $\min_{\theta\in\mathbb{R}^d} \frac{1}{M}\sum_{i=1}^{M}(y_i - \theta' x_i)^2 + \tau||\theta||_1$.

- Logistic Regression: $\min_{\theta\in\mathbb{R}^d} \frac{1}{M}\sum_{i=1}^{M}\log\left(1+\exp\left(-y_i(\theta' x_i)\right)\right)$.

- Support Vector Machine ([10]): $\min_{\theta\in\mathbb{R}^d} \frac{1}{M}\sum_{i=1}^{M}\max\{0, 1-y_i(\theta' x_i)\} + \tau||\theta||_2^2$.

Suppose our distributed computing system consists of n machines each with $k = M/n$ data points (without loss of generality suppose M is divisible by n, otherwise one of the machines have the remainder of data points). For all $i = 1, \ldots, n$, we define a function based on the available data to machine i as $f_i(\theta) = \frac{1}{k}\sum_{1+(i-1)k}^{ik} L(y_i, x_i, \theta) + p(\theta)$. Therefore, the empirical risk minimization (2) can be written as

$$\min_{\theta\in\mathbb{R}^d} \frac{1}{n}\sum_{i=1}^{n} f_i(\theta),$$

where function $f_i(\theta)$ is only available to machine i. Data is distributed across different machines either because it is collected by decentralized agents due to privacy and budget constraints [58, 59, 36] or because memory constraints prevent it from being stored in a single machine[18, 60, 61, 27]. The decentralized nature of data together with communication constraints necessitate distributed processing which has motivated a large literature in optimization and statistical learning on distributed algorithms (see e.g. [22, 32, 54, 2, 45]).

Much of this literature builds on the seminal works [50, 51], which proposed gradient methods that can parallelize computations across multiple processors. A number of recent papers proposed subgradient type methods [38, 43, 7, 29, 47, 43, 30] or dual averaging method [17] to design distributed optimization algorithms.

An alternative approach is to use Alternating Direction Method of Multipliers (ADMM) type methods which for separable problems exploits the separability leading to decoupled computations (see [6] and [19] for comprehensive tutorials on ADMM). ADMM has been studied extensively in the 80's [25, 23, 3]. More recently, it has found applications in a variety of distributed settings in machine learning such as model fitting, resource allocation, and classification (see e.g. [52, 46, 53, 56, 57]). Several recent papers have studied the convergence rate of ADMM in centralized and distributed settings. In the centralized setting, with convex functions, the recent paper [26] shows $O(\frac{1}{\epsilon})$ iteration complexity. In the distributed setting, $O(\frac{1}{\epsilon})$ iteration complexity is proved in [49, 35].

For strongly convex functions with Lipschitz continuous gradients, [15] shows $O(\sqrt{\kappa_f}\log(\frac{1}{\epsilon}))$ iteration complexity in the centralized setting, where κ_f is the condition number of problem defined as L/ν, where L is the maximum Lipschitz gradient parameter and ν is the minimum strong convexity constant of the local objective functions. More generally, in the centralized setting, there are a number of results for operator splitting schemes, such as Douglas-Rachford splitting and relaxed Peaceman-Rachford [33, 20, 40, 41, 12, 13, 24]. The recent paper [39] provides a new proof of the linear convergence of ADMM when the objective functions are strongly convex, which is based on a framework introduced in [31]. In distributed setting, with strong convexity and Lipschitz continuous gradient assumptions, reference [48] shows $O(\kappa_f \log(\frac{1}{\epsilon}))$ iteration complexity for ADMM.

In this paper, we first propose a novel reformulation of distributed multiagent optimization that enables the implementation of ADMM in a distributed way. We then propose a novel analysis to establish $O(\sqrt{\kappa_f}\log(\frac{1}{\epsilon}))$ iteration complexity for distributed ADMM that matches with the best known rate of convergence for centralized ADMM.

The organization of paper is as follows. In section 2 we give the problem formulation and propose a novel distributed ADMM algorithm. In section 3, we show the linear convergence rate of our algorithm and prove $O(\sqrt{\kappa_f}\log(\frac{1}{\epsilon}))$ iteration complexity. Finally, in section 4 we provide numerical results that illustrate the performance of our algorithm. All the omitted proofs are included in the supplementary materials.

2 Framework

2.1 Problem Formulation

Consider a network represented by the connected graph $G = (V, E)$ where $V = \{1, \ldots, n\}$ is the set of agents and E is the set of edges. For any i, let $N(i)$ denote the set of neighbors of agent i including agent i itself, i.e., $N(i) = \{j \mid (i, j) \in E\} \cup \{i\}$, and let d_i denote the degree of agent i, i.e., $|N(i)| = d_i + 1$.

The goal of the agents is to collectively solve the following optimization problem

$$\min_{x \in \mathbb{R}^d} \sum_{i=1}^{n} f_i(x), \tag{3}$$

where $f_i : \mathbb{R}^d \to \mathbb{R}$ is a convex function. We assume that information is distributed across agents, i.e., the function f_i is only known to agent i. In order to solve optimization problem (3), we introduce a variable x_i for each i and write the objective function of problem (3) as $\sum_{i=1}^{n} f_i(x_i)$, so that the objective function is decoupled across the agents. The constraint that all the x_i's are equal can be imposed using the following matrix.

Definition 1. Let P be a $n \times n$ matrix whose entries satisfy the following: for any $i = 1, \ldots, n$, $P_{ij} = 0$ for $j \notin N(i)$, $P_{ij} \leq 0$ for $j \in N(i) \setminus \{i\}$, and $P_{ii} = -\sum_{j \in N(i) \setminus \{i\}} P_{ij}$. We refer to P as the *communication matrix*.

Assumption 1. The communication matrix P satisfies null$(P) =$ span$\{\mathbf{1}\}$, where $\mathbf{1}$ is a $n \times 1$ vector with all entries equal to one.

For instance, if $P_{ij} < 0$ for all $j \in N(i) \setminus \{i\}$ and the graph is connected, then Assumption 1 holds. As a particular case, the Laplacian matrix of the graph given by $P_{ij} = -\mathbf{1}\{j \in N(i) \setminus \{i\}\}$ and $P_{ii} = d_i$ is a communication matrix.

We next show that the constraint that all x_i's are equal can be enforced by the linear constraint $A\mathbf{x} = 0$, where $\mathbf{x} = (x_1, \ldots, x_n)$ with each x_i is a sub-vector of dimension d and A is a $dn \times dn$ matrix defined as the Kronecker product between communication matrix P and I_d, i.e., $A = P \otimes I_d$. Note that the sum of each row of A is zero.

Lemma 1. *Under Assumption 1, the constraint $A\mathbf{x} = 0$ guarantees that $x_i = x_j$ for all $i, j \in V$.*

Using Lemma 1, under Assumption 1, we can reformulate optimization problem (3) as

$$\begin{aligned} &\min_{\mathbf{x} \in \mathbb{R}^{nd}} \sum_{i=1}^{n} f_i(x_i) \\ &\text{s.t. } A\mathbf{x} = 0. \end{aligned} \tag{4}$$

Assumption 2. The optimal solution set of problem (4) is non-empty. We let $\mathbf{x}^*$ denote an optimal solution of problem (4).

2.2 Multiagent Distributed ADMM

In this section, we propose a distributed ADMM algorithm to solve problem (4). We first use a reformulation technique used in [5], which allows us to separate each constraint associated with a node into multiple constraints that involve only the variable corresponding with one of the neighboring nodes. We expand the constraint $A\mathbf{x} = 0$ so that for each node i, we have $\sum_{j \in N(i)} A_{ij} x_j = 0$, where $A_{ij} = P_{ij} \otimes I_d$ is a $d \times d$ matrix. We let $A_{ij} x_j = z_{ij} \in \mathbb{R}^d$ to obtain the following reformulation:

$$\min_{x_i,\, z_{ij}} \sum_{i=1}^{n} f_i(x_i) \tag{5}$$

$$\begin{aligned} &\text{s.t. } A_{ij} x_j = z_{ij}, \quad \text{for } i = 1, \ldots, n,\ j \in N(i), \\ &\sum_{j \in N(i)} z_{ij} = 0, \quad \text{for } i = 1, \ldots, n. \end{aligned} \tag{6}$$

For each equality constraint in (6), we let $\lambda_{ij} \in \mathbb{R}^d$ be the corresponding Lagrange multiplier and form the augmented Lagrangian function by adding a quadratic penalty with penalty parameter $c > 0$ for feasibility violation to the Lagrangian function as

$$L_c(\mathbf{x}, \{z_{ij}\}, \{\lambda_{ij}\}) = \sum_{i=1}^{n} f_i(x_i) + \sum_{i=1}^{n} \sum_{j \in N(i)} \lambda'_{ij}(A_{ij}x_j - z_{ij}) + \frac{c}{2} \sum_{i=1}^{n} \sum_{j \in N(i)} ||A_{ij}x_j - z_{ij}||_2^2.$$

We now use ADMM algorithm (see e.g. [4]) to solve (5). ADMM algorithm generates primal-dual sequences $\{x_j(t)\}, \{z_{ij}(t)\}$, and $\{\lambda_{ij}(t)\}$ which at iteration $t+1$ are updated as follows:

1. For any $j = 1, \ldots, n$, we update x_j as

$$x_j(t+1) \in \text{argmin}_{x_j \in \mathbb{R}^d} \left[f_j(x_j) + \sum_{i \in N(j)} \lambda'_{ij}(t) A_{ij} x_j + \frac{c}{2} \left| |A_{ij}x_j - z_{ij}(t)| \right|_2^2 \right]. \quad (7)$$

2. For any $i = 1, \ldots, n$, we update the vector $\mathbf{z}_i = [z_{ij}]_{j \in N(i)}$ as

$$\mathbf{z}_i(t+1) \in \text{argmin}_{\mathbf{z}_i \in Z_i} \left[\sum_{j \in N(i)} \left(-\lambda'_{ij}(t) z_{ij} + \frac{c}{2} ||A_{ij}x_j(t+1) - z_{ij}||_2^2 \right) \right], \quad (8)$$

   where $Z_i = \{\mathbf{z}_i \mid \sum_{j \in N(i)} z_{ij} = 0\}$.

3. For $i = 1, \ldots, n$ and $j \in N(i)$ we update λ_{ij} as

$$\lambda_{ij}(t+1) = \lambda_{ij}(t) + c(A_{ij}x_j(t+1) - z_{ij}(t+1)). \quad (9)$$

One can implement this algorithm in a distributed manner, where node i maintains variables $\lambda_{ij}(t)$ and $z_{ij}(t)$ for all $j \in N(i)$. However, using the inherent symmetries in the problem, we can significantly reduce the number of variables that each node requires to maintain from $O(|E|)$ to $O(|V|)$.

The first observation is that for all t, i, and $j \in N(i)$, we have $\lambda_{ij}(t) = p_i(t)$. This reduction shows that the algorithm need not maintain dual variables $\lambda_{ij}(t)$ for each i and its neighbors j, but instead can operate with the lower dimensional node-based dual variable $p_i(t)$. The dual variable $p_i(t)$ can be updated as in (12) using primal variables $x_j(t)$ for all $j \in N(i)$. The second observation is that $z_{ij}(t) = A_{ij}x_j(t) - y_i(t)$, where $y_i(t) = \frac{1}{d_i+1} \left([A]^i\right)' \mathbf{x}(t)$ and $\left([A]^i\right)' = (P_{i1}, \ldots, P_{in}) \otimes I_d$. This reduction shows that the algorithm need not maintain primal variables $z_{ij}(t)$ for each i and its neighbors j, but instead can operate with the lower dimensional node-based primal variables $y_i(t)$, where $y_i(t)$ is node i's estimate of the primal variable (obtained as the average of primal variables of his own neighbors). The steps of the algorithm can be implemented in a distributed way, meaning that each node first updates her estimates based on the information received from her neighboring nodes and then broadcasts her updated estimates to her neighboring nodes.

Proposition 1. *The sequence $\{x_i(t)\}_{t=0}^{\infty}$ for $i = 1, \ldots, n$ generated by implementing the steps presented in Algorithm* (1) *is the same as the sequence generated by ADMM algorithm.*

3 Convergence Analysis

3.1 Preliminary Results

In order to establish the linear rate of convergence, we adopt the following standard assumptions.

Assumption 3 (Strongly convex and Lipschitz Gradient). For any $i = 1, \ldots, n$, the function f_i is differentiable and has Lipschitz continuous gradient, i.e.,

$$|\nabla f_i(x) - \nabla f_i(y)| \leq L_{f_i} ||x - y||_2, \text{ for any } x, y \in \mathbb{R}^d,$$

for some $L_{f_i} \geq 0$. The function f_i is also strongly convex with parameter $\nu_{f_i} > 0$, i.e., $f_i(x) - \frac{\nu_{f_i}}{2} ||x||_2^2$ is convex.

Algorithm 1 Multiagent Distributed ADMM

- **Initialization:** $x_i(0)$, $y_i(0)$, and $p_i(0)$ all in $\mathbb{R}^d$, for any $i \in V$ and matrix $A \in \mathbb{R}^{nd \times nd}$.
- **Algorithm:**
  1. for $i = 1, \ldots, n$, let

$$x_i(t+1) \in \text{argmin}_{x_i \in \mathbb{R}^d} f_i(x_i) + \sum_{j \in N(i)} p_j'(t) A_{ji} x_i + \frac{c}{2} ||y_j(t) + A_{ji}(x_i - x_i(t)||_2^2. \tag{10}$$

  2. for $i = 1, \ldots, n$, let

$$y_i(t+1) = \frac{1}{d_i + 1} \sum_{j \in N(i)} A_{ij} x_j(t+1). \tag{11}$$

  3. for $i = 1, \ldots, n$, let

$$p_i(t+1) = p_i(t) + c y_i(t+1) \tag{12}$$

- **Output:** $\{x_i(t)\}_{t=0}^{\infty}$ for any $i \in V$.

We let the global objective function to be $F(\mathbf{x}) = \sum_{i=1}^n f_i(x_i)$, where $\mathbf{x} = (x_1, \ldots, x_n) \in \mathbb{R}^{nd}$. Denoting $\nu = \min_{1 \le i \le n} \nu_{f_i}$ and $L = \max_{1 \le i \le n} L_{f_i}$, it follows from Assumption 3 that the function $F(\mathbf{x})$ has Lipschitz continuous gradient with parameter L and is strongly convex with parameter ν. We define the condition number of $F(\mathbf{x})$ (or the condition number of problem (4)) as $\kappa_f = \frac{L}{\nu}$.

Assumption 3 results in the following standard inequalities for function $F(\mathbf{x})$.

Lemma 2. *Under Assumption 3, we have*

(a) For any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{nd}$, we have

$$(\nabla F(\mathbf{x}) - \nabla F(\mathbf{y}))'(\mathbf{x} - \mathbf{y}) \ge \nu ||\mathbf{x} - \mathbf{y}||_2^2,$$

for $\nu = \min_{1 \le i \le n} \nu_{f_i}$.

(b) For any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^{nd}$, we have

$$(\nabla F(\mathbf{x}) - \nabla F(\mathbf{y}))'(\mathbf{x} - \mathbf{y}) \ge \frac{1}{L} ||\mathbf{x} - \mathbf{y}||_2^2,$$

for $L = \max_{1 \le i \le n} L_{f_i}$.

In what follows, for notational simplicity we assume $d = 1$, i.e., in (3) $x \in \mathbb{R}$. All the analysis generalizes to the case with $x \in \mathbb{R}^d$. We next provide a compact representation of the evolution of primal vector $\mathbf{x}(t)$ that will be used in the convergence proof. This is a core step in proving the convergence rate as it eliminates the dependence on the other variables $y_i(t)$'s and $p_i(t)$'s. Let M be a $n \times n$ diagonal matrix with $M_{ii} = \sum_{j \in N(i)} A_{ji}^2$ and D be a $n \times n$ diagonal matrix with $D_{ii} = d_i + 1$.

Lemma 3 (**Perturbed Linear Update**). *The update of Algorithm 1 can be written as*

$$\mathbf{x}(t+1) = -\frac{1}{c} M^{-1} \nabla F(\mathbf{x}(t+1)) + \left(I - M^{-1} A' D^{-1} A\right) \mathbf{x}(t) - M^{-1}(A' D^{-1} A) \sum_{s=0}^{t} \mathbf{x}(s). \tag{13}$$

We next show that both matrices $M - A'D^{-1}A$ and $A'D^{-1}A$ are positive semidefinite.

Lemma 4. *The matrices $M - A'D^{-1}A$ and $A'D^{-1}A$ are positive semidefinite.*

3.2 Linear Rate of Convergence

In this section, we establish the linear convergence rate of our algorithm under Assumptions 1, 2, and 3. We first define two auxiliary sequences that we will use in proving the convergence rate. Since $A'D^{-1}A$ is positive semidefinite, we can define $Q = (A'D^{-1}A)^{1/2}$. In other words, we let $Q = V\Sigma^{1/2}V'$, where $A'D^{-1}A = V\Sigma V'$ is the singular value decomposition of the symmetric matrix $A'D^{-1}A$. We define the auxiliary sequences $\mathbf{r}(t) = \sum_{s=0}^{t} Q\mathbf{x}(s)$ and $\mathbf{q}(t) = \begin{pmatrix}\mathbf{r}(t)\\ \mathbf{x}(t)\end{pmatrix}$. The following lemma shows the relation between the auxiliary sequence $\mathbf{r}(t)$ and primal sequence $\mathbf{x}(t)$.

Lemma 5. *Suppose Assumptions 1 and 2 hold. The sequence $\{\mathbf{x}(t), \mathbf{r}(t)\}_{t=0}^{\infty}$ satisfies*

$$(M - A'D^{-1}A)(\mathbf{x}(t+1) - \mathbf{x}(t)) = -Q(\mathbf{r}(t+1) - \mathbf{r}^*) - \frac{1}{c}(\nabla F(\mathbf{x}(t+1)) - \nabla F(\mathbf{x}^*)), \quad (14)$$

for some $\mathbf{r}^*$ that satisfies $Q\mathbf{r}^* + \frac{1}{c}\nabla F(\mathbf{x}^*) = 0$. Moreover, $\mathbf{r}^*$ belongs to the column span of Q.

Using this relation together with strongly convex and Lipschitz gradient property of $F(\mathbf{x})$, we show that a particular weighted norm of sequence $\mathbf{q}(t) - \mathbf{q}^*$ contracts at each iteration thus providing the following linear rate.

Theorem 1. *Suppose Assumptions 1, 2, and 3 hold. For any value of the penalty parameter $c > 0$ and $\beta \in (0, 1)$, the sequence generated by Algorithm 1 $\{\mathbf{x}(t)\}_{t=1}^{\infty}$ satisfies*

$$||\mathbf{x}(t) - \mathbf{x}^*||_2^2 \leq \left(\frac{1}{1+\delta}\right)^t K,$$

where $\delta \leq \min\{\frac{2\beta\nu}{c\lambda_M(1+\frac{2}{\tilde{\lambda}_m})}, \frac{(1-\beta)c\tilde{\lambda}_m}{L}\}$, and $\tilde{\lambda}_m$ is the smallest non-zero eigenvalue of $A'D^{-1}A$, λ_M is the largest eigenvalue of $M - A'D^{-1}A$, and K is a constant that depends on the initial conditions.

The rate of convergence in Theorem 1 holds for any choice of penalty parameter $c > 0$. In other words, for any choice of $c > 0$, the convergence rate is linear.[2] We now optimize the rate of convergence over all choices of c and provide an explicit rate of convergence based on function condition number.

Theorem 2. *Suppose Assumptions 1, 2, and 3 hold. Let $\{\mathbf{x}(t)\}_{t=1}^{\infty}$ be the sequence generated by Algorithm 1. There exist some $c > 0$ such that $O(\sqrt{\kappa_f}\log(\frac{1}{\epsilon}))$ iterations suffices to reach to an ϵ-optimal neighborhood. More precisely, we have*

$$||\mathbf{x}(t) - \mathbf{x}^*||_2^2 \leq \rho^t K,$$

where the rate $\rho < 1$ is given by $\rho = \left(1 + \frac{1}{2}\sqrt{\frac{2\tilde{\lambda}_m^2}{\lambda_M(2+\tilde{\lambda}_m)}\frac{1}{\kappa_f}}\right)^{-1}$ and K is constant.

Remark 1. The iteration complexity depends on condition number κ_f as well as network structure and the choice of communication matrix. In particular, the larger $\tilde{\lambda}_m$ and the smaller λ_M results in a faster rate of convergence. It can be seen that $A'D^{-1}A$ is the generalized Laplacian matrix [9] and $M - A'D^{-1}A$ is the generalized singless Laplacian matrix [11] of the graph G (with weights given by the entries of A). The smallest non-zero eigenvalue of $A'D^{-1}A$, $\tilde{\lambda}_m$, also known as algebraic connectivity and the maximum eigenvalue of $M - A'D^{-1}A$, λ_M, are both related to connectedness of the graph[21, 8].

4 Numerical Evaluation

In this section, we show numerical results based on the proposed algorithm to demonstrate its performance. We consider minimizing the function $F(x) = \frac{1}{2}\sum_{i=1}^{n}(x - a_i)^2$ where a_i is a scalar that

[2]The convergence rate is in fact R-linear because $||\mathbf{x}(t) - \mathbf{x}^*||_2$ is upper bounded by sequence $\left(\sqrt{\frac{1}{1+\delta}}\right)^t \left(\sqrt{\frac{c}{2\nu}||\mathbf{q}(0) - \mathbf{q}^*||_G^2}\right)$ that Q-linearly converges to zero. A sequence $\{a_t\}_{t=0}^{\infty}$ Q-linearly converges to zero if there exist $r \in (0, 1)$ such that $\frac{|a_{t+1}|}{|a_t|} \leq r$.

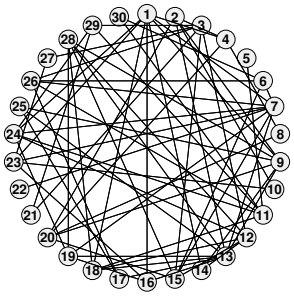

Figure 1: Sample network with $n = 30$ nodes.

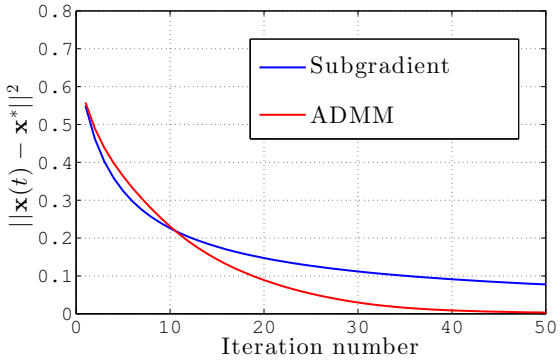

Figure 2: Performance of the algorithm over sample network, $||\mathbf{x}(t) - \mathbf{x}^*||_2^2$ as a function of t.

is known only to machine i. This problem appears in distributed estimation where the goal is to estimate the parameter x^*, using local measurements $a_i = x^* + N_i$ at each machine $i = 1, \ldots, n$. Here N_i represents measurements noise, which we assume to be jointly Gaussian with mean zero and variance one. The maximum likelihood estimate is the minimizer x^* of $F(x)$. We let $n = 30$ and consider the graph shown in Figure 1 with standard Laplacian matrix as communication matrix. We plot the value $||\mathbf{x}(t) - \mathbf{x}^*||_2^2$ as a function of t in Figure 2. We then compare our algorithm with the subgradient type algorithm with optimal step-size (see e.g. [37] and references therein).

References

[1] A. Agarwal and J. C. Duchi. Distributed delayed stochastic optimization. In *Advances in Neural Information Processing Systems, 2011*.

[2] Ö. Aslan, H. Cheng, X. Zhang, and D. Schuurmans. Convex two-layer modeling. In *Advances in Neural Information Processing Systems*, 2013.

[3] D. P. Bertsekas and J. Eckstein. Dual coordinate step methods for linear network flow problems. *Mathematical Programming*, 1988.

[4] D. P. Bertsekas, A. Nedić, and A. E. Ozdaglar. *Convex analysis and optimization*. Athena Scientific Belmont, 2003.

[5] D. P. Bertsekas and J. N. Tsitsiklis. *Parallel and distributed computation: numerical methods*. Prentice-Hall, Inc., 1989.

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

[7] J. Chen and A. H. Sayed. Diffusion adaptation strategies for distributed optimization and learning over networks. *IEEE Transactions on Signal Processing, 2012*.

[8] Y. Chen and L. Wang. Sharp bounds for the largest eigenvalue of the signless laplacian of a graph. *Linear Algebra and Its Applications*, 2010.

[9] F. R. Chung. *Spectral graph theory*. American Mathematical Soc., 1997.

[10] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 1995.

[11] D. Cvetković, P. Rowlinson, and S. K. Simić. Signless laplacians of finite graphs. *Linear Algebra and its applications*, 2007.

[12] D. Davis and W. Yin. Convergence rate analysis of several splitting schemes. *arXiv preprint arXiv:1406.4834*, 2014.

[13] D. Davis and W. Yin. Convergence rates of relaxed Peaceman-Rachford and ADMM under regularity assumptions. *arXiv preprint arXiv:1407.5210*, 2014.

[14] O. Dekel, R. Gilad-Bachrach, O. Shamir, and L. Xiao. Optimal distributed online prediction using mini-batches. *The Journal of Machine Learning Research, 2012*.

[15] W. Deng and W. Yin. On the global and linear convergence of the generalized alternating direction method of multipliers. Technical report, DTIC Document, 2012.

[16] M. F. Duarte and Y. H. Hu. Vehicle classification in distributed sensor networks. *Journal of Parallel and Distributed Computing, 2014*.

[17] J. C. Duchi, A. Agarwal, and M. J. Wainwright. Dual averaging for distributed optimization: convergence analysis and network scaling. *IEEE Transactions on Automatic Control*, 2012.

[18] J. C. Duchi, M. I. Jordan, M. J. Wainwright, and Y. Zhang. Optimality guarantees for distributed statistical estimation. *arXiv preprint arXiv:1405.0782*, 2014.

[19] J. Eckstein. Augmented lagrangian and alternating direction methods for convex optimization: A tutorial and some illustrative computational results. *RUTCOR Research Reports*, 32, 2012.

[20] J. Eckstein and M. C. Ferris. Operator-splitting methods for monotone affine variational inequalities, with a parallel application to optimal control. *INFORMS Journal on Computing*, 1998.

[21] M. Fiedler. Algebraic connectivity of graphs. *Czechoslovak Mathematical Journal*, 1973.

[22] P. A. Forero, A. Cano, and G. B. Giannakis. Consensus-based distributed support vector machines. *The Journal of Machine Learning Research*, 2010.

[23] D. Gabay and B. Mercier. A dual algorithm for the solution of nonlinear variational problems via finite element approximation. *Computers & Mathematics with Applications*, 1976.

[24] P. Giselsson and S. Boyd. Diagonal scaling in Douglas-Rachford splitting and ADMM. In *IEEE Conference on Decision and Control*, 2014.

[25] R. Glowinski and A. Marroco. Sur l'approximation, par e?le?ments finis d'ordre un, et la re?solution, par pe?nalisation-dualite? d'une classe de proble?mes de dirichlet non line?aires. *ESAIM: Mathematical Modelling and Numerical Analysis-Modélisation Mathématique et Analyse Numérique*, 1975.

[26] B. He and X. Yuan. On the o(1/n) convergence rate of the Douglas-Rachford alternating direction method. *SIAM Journal on Numerical Analysis*, 2012.

[27] M. E. Hellman and T. M. Cover. Learning with finite memory. *The Annals of Mathematical Statistics*, 1970.

[28] A. Jadbabaie, J. Lin, and A. S. Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on Automatic Control*, 2003.

[29] D. Jakovetic, J. Xavier, and J. M. Moura. Fast distributed gradient methods. *IEEE Transactions on Automatic Control, 2014*.

[30] B. Johansson, M. Rabi, and M. Johansson. A randomized incremental subgradient method for distributed optimization in networked systems. *SIAM Journal on Optimization*, 2009.

[31] L. Lessard, B. Recht, and A. Packard. Analysis and design of optimization algorithms via integral quadratic constraints. *arXiv preprint arXiv:1408.3595*, 2014.

[32] M. Li, D. G. Andersen, A. J. Smola, and K. Yu. Communication efficient distributed machine learning with the parameter server. In *Advances in Neural Information Processing Systems*, 2014.

[33] P.-L. Lions and B. Mercier. Splitting algorithms for the sum of two nonlinear operators. *SIAM Journal on Numerical Analysis*, 1979.

[34] S. H. Low and D. E. Lapsley. Optimization flow control?i: basic algorithm and convergence. *IEEE/ACM Transactions on Networking (TON), 1999*.

[35] A. Makhdoumi and A. Ozdaglar. Broadcast-based distributed alternating direction method of multipliers. In *Allerton Conference on Communication, Control, and Computing*, 2014.

[36] I. Mitliagkas, C. Caramanis, and P. Jain. Memory limited, streaming pca. In *Advances in Neural Information Processing Systems*, 2013.

[37] A. Nedic and A. Ozdaglar. On the rate of convergence of distributed subgradient methods for multi-agent optimization. In *Proceedings of IEEE CDC*, 2007.

[38] A. Nedic, A. Ozdaglar, and P. A. Parrilo. Constrained consensus and optimization in multi-agent networks. *IEEE Transactions on Automatic Control*, 2010.

[39] R. Nishihara, L. Lessard, B. Recht, A. Packard, and M. I. Jordan. A general analysis of the convergence of ADMM. *arXiv preprint arXiv:1502.02009*, 2015.

[40] P. Patrinos, L. Stella, and A. Bemporad. Douglas-Rachford splitting: complexity estimates and accelerated variants. *arXiv preprint arXiv:1407.6723*, 2014.

[41] P. Patrinos, L. Stella, and A. Bemporad. Forward-backward truncated newton methods for large-scale convex composite optimization. *arXiv preprint arXiv:1402.6655*, 2014.

[42] M. Rabbat and R. Nowak. Distributed optimization in sensor networks. In *International symposium on Information processing in sensor networks, 2004*.

[43] S. S. Ram, A. Nedić, and V. V. Veeravalli. Distributed stochastic subgradient projection algorithms for convex optimization. *Journal of optimization theory and applications, 2010*.

[44] B. Recht, C. Re, S. Wright, and F. Niu. Hogwild: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in Neural Information Processing Systems, 2011*.

[45] B. Romera-Paredes and M. Pontil. A new convex relaxation for tensor completion. In *Advances in Neural Information Processing Systems*, 2013.

[46] H. Sedghi, A. Anandkumar, and E. Jonckheere. Multi-step stochastic ADMM in high dimensions: Applications to sparse optimization and matrix decomposition. In *Advances in Neural Information Processing Systems*, 2014.

[47] W. Shi, Q. Ling, G. Wu, and W. Yin. Extra: An exact first-order algorithm for decentralized consensus optimization. *arXiv preprint arXiv:1404.6264*, 2014.

[48] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin. On the linear convergence of the ADMM in decentralized consensus optimization. *IEEE Transactions on Signal Processing*, 2014.

[49] S. Shtern, E. Wei, and A. Ozdaglar. Distributed alternating direction method of multipliers (ADMM): Performance and network effects. In *Working paper*.

[50] J. N. Tsitsiklis. Problems in decentralized decision making and computation. Technical report, DTIC Document, 1984.

[51] J. N. Tsitsiklis, D. P. Bertsekas, and M. Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE Transactions on Automatic Control, 1986*.

[52] B. Wahlberg, S. Boyd, M. Annergren, and Y. Wang. An ADMM algorithm for a class of total variation regularized estimation problems. In *Preprints of the 16th IFAC Symposium on System Identification*, 2012.

[53] H. Wang and A. Banerjee. Online alternating direction method. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, 2012.

[54] H. Wang, A. Banerjee, C.-J. Hsieh, P. K. Ravikumar, and I. S. Dhillon. Large scale distributed sparse precision estimation. In *Advances in Neural Information Processing Systems*, 2013.

[55] L. Xiao, S. Boyd, and S.-J. Kim. Distributed average consensus with least-mean-square deviation. *Journal of Parallel and Distributed Computing*, 2007.

[56] C. Zhang, H. Lee, and K. G. Shin. Efficient distributed linear classification algorithms via the alternating direction method of multipliers. In *International Conference on Artificial Intelligence and Statistics*, 2012.

[57] R. Zhang and J. Kwok. Asynchronous distributed ADMM for consensus optimization. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, 2014.

[58] Y. Zhang, J. Duchi, M. I. Jordan, and M. J. Wainwright. Information-theoretic lower bounds for distributed statistical estimation with communication constraints. In *Advances in Neural Information Processing Systems*, 2013.

[59] Y. Zhang, J. Duchi, and M. Wainwright. Divide and conquer kernel ridge regression. In *Conference on Learning Theory*, 2013.

[60] Y. Zhang, M. J. Wainwright, and J. C. Duchi. Communication-efficient algorithms for statistical optimization. In *Advances in Neural Information Processing Systems*, 2012.

[61] Y. Zhang and L. Xiao. Communication-efficient distributed optimization of self-concordant empirical loss. *arXiv preprint arXiv:1501.00263*, 2015.