ADMM for Distributed Optimization

Hailiang ZHAO @ ZJU-CS

http://hliangzhao.me

October 29, 2022

Outline

I Centralized Distributed Optimization

II Decentralized Distributed Optimization

III Asynchronous Distributed ADMM

IV Nonconvex Distributed ADMM

V ADMM with Generally Linear Constraints

References

Outline

I Centralized Distributed Optimization

II Decentralized Distributed Optimization

III Asynchronous Distributed ADMM

IV Nonconvex Distributed ADMM

V ADMM with Generally Linear Constraints

References

The Distributed Optimization Problem

Consider the following problem in a distributed environment:

$$\min_{\boldsymbol{x}\in\mathbb{R}^d} f(\boldsymbol{x}) \equiv \sum_{i=1}^{m} f_i(\boldsymbol{x}_i), \tag{1}$$

where m agents form a connected and undirected network and the local function f_i is only accessible by agent i due to storage or privacy reasons. Two kinds of networks:

- The centralized network with one centralized master agent and m worker agents. Each worker agent is connected to the master agent
- The decentralized network, which does not have the centralized agent and each agent only communicates with its neighbors

All the agents cooperate to solve (1).

Centralized Optimization

In the centralized network, we reformulate (1) as the following linearly constrained one:

$$\min_{\boldsymbol{x}_i, \boldsymbol{z}} \sum_{i=1}^{m} f_i(\boldsymbol{x}_i), \quad s.t. \quad \boldsymbol{x}_i = \boldsymbol{z}, \forall i \in [m]. \tag{2}$$

Obviously, the vanilla ADMM can solve it. Introduce the augmented Lagrangian function:

$$L_\beta(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}) = \sum_{i=1}^{m} \left(f_i(\boldsymbol{x}_i) + \langle \boldsymbol{\lambda}_i, \boldsymbol{x}_i - \boldsymbol{z} \rangle + \frac{\beta}{2} \|\boldsymbol{x}_i - \boldsymbol{z}\|^2 \right). \tag{3}$$

CADMM

The corresponding iterations are:

$$\begin{aligned}
\boldsymbol{z}^{k+1} &= \operatorname*{argmin}_{\boldsymbol{z}} \sum_{i=1}^{m} \left(\langle \boldsymbol{\lambda}_i^k, \boldsymbol{x}_i^k - \boldsymbol{z} \rangle + \frac{\beta}{2} \|\boldsymbol{x}_i^k - \boldsymbol{z}\|^2 \right) \\
&= \frac{1}{m} \sum_{i=1}^{m} (\boldsymbol{x}_i^k + \frac{1}{\beta} \boldsymbol{\lambda}_i^k), \qquad (4) \\
\boldsymbol{x}_i^{k+1} &= \operatorname*{argmin}_{\boldsymbol{x}_i} \left(f_i(\boldsymbol{x}_i) + \langle \boldsymbol{\lambda}_i^k, \boldsymbol{x}_i^k - \boldsymbol{z}^{k+1} \rangle + \frac{\beta}{2} \|\boldsymbol{x}_i^k - \boldsymbol{z}^{k+1}\|^2 \right) \\
&= \operatorname{prox}_{\beta^{-1} f_i}(\boldsymbol{z}^{k+1} - \frac{1}{\beta} \boldsymbol{\lambda}_i^k), \quad i \in [m], \qquad (5) \\
\boldsymbol{\lambda}_i^{k+1} &= \boldsymbol{\lambda}_i^k + \beta(\boldsymbol{x}_i^{k+1} - \boldsymbol{z}^{k+1}), \quad i \in [m]. \qquad (6)
\end{aligned}$$

These steps are executed by different roles.

CADMM

The master should do —— In each iteration k:

1. Wait until receiving $\boldsymbol{x}_i^k$ and $\boldsymbol{\lambda}_i^k$ from all workers
2. Update $\boldsymbol{z}^{k+1}$ with (4)
3. Send $\boldsymbol{z}^{k+1}$ to all the workers

Each worker i should do —— In each iteration k:

1. Send $\boldsymbol{x}_i^k$ and $\boldsymbol{\lambda}_i^k$ to the master
2. Wait until receiving $\boldsymbol{z}^{k+1}$ from the master
3. Update $\boldsymbol{x}_i^{k+1}$ and $\boldsymbol{\lambda}_i^{k+1}$ with (5) and (6), respectively

We call this method Centralized ADMM for Master and Worker (*CADMM-M and CADMM-W*).

Convergence Rate of CADMM

A KKT point of (2) is denoted by $(\boldsymbol{x}_1^*, ..., \boldsymbol{x}_m^*, \boldsymbol{z}^*, \boldsymbol{\lambda}_1^*, ..., \boldsymbol{\lambda}_m^*)$. CADMM has a similar convergence to the vanilla ADMM.

Theorem I.1 (*similar to Theorem I.3 of Slide 2*)

Suppose that $i \in [m]\left[f_i(\boldsymbol{x}_i) \text{ is convex}\right]$. Then for CADMM, we have

$$\left| \sum_i f_i(\hat{\boldsymbol{x}}_i^{K+1}) - \sum_i f_i(\boldsymbol{x}_i^*) \right| \leq \frac{C}{2(K+1)} + \frac{2\sqrt{C}\sqrt{\sum_i \|\boldsymbol{\lambda}_i^*\|^2}}{\sqrt{\beta}(K+1)} \tag{7}$$

$$\sqrt{\sum_i \|\hat{\boldsymbol{x}}_i^{K+1} - \hat{\boldsymbol{z}}^{K+1}\|^2} \leq \frac{2\sqrt{C}}{\sqrt{\beta}(K+1)}, \tag{8}$$

where $C = \frac{1}{\beta}\sum_i \|\boldsymbol{\lambda}_i^0 - \boldsymbol{\lambda}_i^*\|^2 + \beta \sum_i \|\boldsymbol{x}_i^0 - \boldsymbol{x}_i^*\|^2$.

Convergence Rate of CADMM

We also have:

Theorem I.2 (*similar to Theorem I.4 of Slide 2*)

Suppose that *each f_i is μ-strongly convex and L-smooth*. Then for CADMM we have

$$\sum_i \left(\frac{1}{2\beta} \|\boldsymbol{\lambda}_i^{k+1} - \boldsymbol{\lambda}_i^*\|^2 + \frac{\beta}{2} \|\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^*\|^2 \right)$$

$$\leq \left(1 + \frac{1}{2}\sqrt{\frac{\mu}{L}}\right)^{-1} \sum_i \left(\frac{1}{2\beta} \|\boldsymbol{\lambda}_i^{k} - \boldsymbol{\lambda}_i^*\|^2 + \frac{\beta}{2} \|\boldsymbol{x}_i^{k} - \boldsymbol{x}_i^*\|^2 \right). \quad (9)$$

CLADMM

LADMM can also be used to solve (2). We can linearize f_i in (5) simplify the computation, if the proximal mapping of f_i is not easily computable. The iterations are:

$$\begin{aligned} \boldsymbol{z}^{k+1} &= \operatorname*{argmin}_{\boldsymbol{z}} \sum_{i=1}^{m} \left(\langle \boldsymbol{\lambda}_i^k, \boldsymbol{x}_i^k - \boldsymbol{z} \rangle + \frac{\beta}{2} \|\boldsymbol{x}_i^k - \boldsymbol{z}\|^2 \right) \\ &= \frac{1}{m} \sum_{i=1}^{m} (\boldsymbol{x}_i^k + \frac{1}{\beta} \boldsymbol{\lambda}_i^k), \qquad \text{(unchanged)} \end{aligned} \tag{10}$$

CLADMM

The iterations are (cont'd):

$$\begin{aligned}
\boldsymbol{x}_i^{k+1} &= \operatorname*{argmin}_{\boldsymbol{x}_i} \Big(f_i(\boldsymbol{x}_i) + \langle \boldsymbol{\lambda}_i^k, \boldsymbol{x}_i^k - \boldsymbol{z}^{k+1} \rangle \\
&\quad + \frac{\beta}{2} \|\boldsymbol{x}_i^k - \boldsymbol{z}^{k+1}\|^2 + D_{\Psi_i}(\boldsymbol{x}_i, \boldsymbol{x}_i^k) \Big) \\
&= \operatorname*{argmin}_{\boldsymbol{x}_i} \Big(\langle \nabla f_i(\boldsymbol{x}_i^k), \boldsymbol{x}_i - \boldsymbol{x}_i^k \rangle + \frac{L}{2} \|\boldsymbol{x}_i - \boldsymbol{x}_i^k\|^2 \\
&\quad + \langle \boldsymbol{\lambda}_i^k, \boldsymbol{x}_i - \boldsymbol{z}^{k+1} \rangle + \frac{\beta}{2} \|\boldsymbol{x}_i - \boldsymbol{z}^{k+1}\|^2 \Big) \\
&= \frac{1}{L+\beta} \Big(L\boldsymbol{x}_i^k + \beta \boldsymbol{z}^{k+1} - \nabla f_i(\boldsymbol{x}_i^k) - \boldsymbol{\lambda}_i^k \Big), \qquad (11)
\end{aligned}$$

$$\boldsymbol{\lambda}_i^{k+1} = \boldsymbol{\lambda}_i^k + \beta(\boldsymbol{x}_i^{k+1} - \boldsymbol{z}^{k+1}), \quad i \in [m], \quad \text{(unchanged)} \qquad (12)$$

by choosing $\Psi_i(\boldsymbol{x}_i) = \frac{L}{2}\|\boldsymbol{x}_i\|^2 - f_i(\boldsymbol{x}_i)$.

CLADMM

The master should do —— In each iteration k:

1. Wait until receiving $\boldsymbol{x}_i^k$ and $\boldsymbol{\lambda}_i^k$ from all workers
2. Update $\boldsymbol{z}^{k+1}$ with (10)
3. Send $\boldsymbol{z}^{k+1}$ to all the workers

Each worker i should do —— In each iteration k:

1. Send $\boldsymbol{x}_i^k$ and $\boldsymbol{\lambda}_i^k$ to the master
2. Wait until receiving $\boldsymbol{z}^{k+1}$ from the master
3. Update $\boldsymbol{x}_i^{k+1}$ and $\boldsymbol{\lambda}_i^{k+1}$ with (11) and (12), respectively

We call this method Centralized Linearized ADMM for Master and Worker (*CLADMM-M and CLADMM-W*).

Convergence Rate of CLADMM

Of course CLADMM can achieve sublinear convergence rate (Theorem II.1 of *Slide 2*). Further, based on Theorem 3.8 of the ADMM book and using $L_\Psi \leq L - \mu$, where $\Psi(\boldsymbol{x}) = \sum_i \Psi_i(\boldsymbol{x}_i)$, CLADMM can achieve linear convergence rate.

Theorem I.3

Suppose that each $f_i(\boldsymbol{x}_i)$ is μ-strongly convex and L-smooth. Let $\beta = \sqrt{\mu}(2L - \mu)$. Then for CLADMM, we have

$$\begin{aligned}
&\sum_i \left(\frac{1}{2\beta} \|\boldsymbol{\lambda}_i^{k+1} - \boldsymbol{\lambda}_i^*\|^2 + \frac{\beta}{2} \|\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^*\|^2 + D_{\Psi_i}(\boldsymbol{x}_i^*, \boldsymbol{x}_i^{k+1}) \right) \\
&\qquad \leq \left[1 + \frac{1}{3} \min \left(\sqrt{\frac{\mu}{2L - \mu}}, \frac{\mu}{L - \mu} \right) \right]^{-1} \\
&\times \sum_i \left(\frac{1}{2\beta} \|\boldsymbol{\lambda}_i^{k} - \boldsymbol{\lambda}_i^*\|^2 + \frac{\beta}{2} \|\boldsymbol{x}_i^{k} - \boldsymbol{x}_i^*\|^2 + D_{\Psi_i}(\boldsymbol{x}_i^*, \boldsymbol{x}_i^{k}) \right).
\end{aligned} \tag{13}$$

Acc-CLADMM

Accelerated Linearized ADMM can also be used to solve (2). Based on Acc-LADMM-3, we have the following iterations:

$$\boldsymbol{w}_i^k = \theta \boldsymbol{x}_i^k + (1-\theta)\tilde{\boldsymbol{x}}_i^k, \tag{14}$$

$$\begin{aligned}\boldsymbol{z}^{k+1} &= \operatorname*{argmin}_{\boldsymbol{z}} \sum_i \left(\langle \boldsymbol{\lambda}_i^k, \boldsymbol{x}_i^k - \boldsymbol{z} \rangle + \frac{\beta\theta}{2} \|\boldsymbol{x}_i^k - \boldsymbol{z}\|^2 \right) \\ &= \frac{1}{m} \sum_i \left(\boldsymbol{x}_i^k + \frac{1}{\beta\theta} \boldsymbol{\lambda}_i^k \right), \end{aligned} \tag{15}$$

$$\begin{aligned}\boldsymbol{x}_i^{k+1} = \frac{1}{\frac{\theta}{\alpha} + \mu} \Big\{ \mu \boldsymbol{w}_i^k + \frac{\theta}{\alpha} \boldsymbol{x}_i^k - \big[\nabla f_i(\boldsymbol{x}_i^k) + \boldsymbol{\lambda}_i^k \\ + \beta\theta(\boldsymbol{x}_i^k - \boldsymbol{z}^{k+1}) \big] \Big\}, \end{aligned} \tag{16}$$

Acc-CLADMM

Based on Acc-LADMM-3, we have the following iterations (cont'd):

$$\tilde{\boldsymbol{z}}^{k+1} = \theta \boldsymbol{z}^{k+1} + (1-\theta)\tilde{\boldsymbol{z}}^{k}, \tag{17}$$

$$\tilde{\boldsymbol{x}}_i^{k+1} = \theta \boldsymbol{x}_i^{k+1} + (1-\theta)\tilde{\boldsymbol{x}}_i^{k}, \tag{18}$$

$$\boldsymbol{\lambda}_i^{k+1} = \boldsymbol{\lambda}_i^{k} + \beta\theta(\boldsymbol{x}_i^{k+1} - \boldsymbol{z}^{k+1}). \tag{19}$$

Acc-CLADMM

The master should do —— In each iteration k:

1. Wait until receiving $\boldsymbol{x}_i^k$ and $\boldsymbol{\lambda}_i^k$ from all workers
2. Update $\boldsymbol{z}^{k+1}$ and $\tilde{\boldsymbol{z}}^{k+1}$ with (15) and (17), respectively

Each worker i should do —— In each iteration k:

1. Send $\boldsymbol{x}_i^k$ and $\boldsymbol{\lambda}_i^k$ to the master
2. Wait until receiving $\boldsymbol{z}^{k+1}$ from the master
3. Update $\boldsymbol{x}_i^{k+1}$ and $\boldsymbol{\lambda}_i^{k+1}$ with (5) and (6), respectively

We call this method Accelerated Centralized Linearized ADMM for Master and Worker (*Acc-CLADMM-M and Acc-CLADMM-W*).

Convergence Rate of Acc-CLADMM

Similar to Theorem III.1 of *ADMM Slide 2*, we have the following convergence rate for Acc-CLADMM.

Theorem I.4

Suppose that each $f_i(\boldsymbol{x}_i)$ is μ-strongly convex and L-smooth, $i \in [m]$. Let $\alpha = \frac{1}{4L}, \beta = L,$ and $\theta = \sqrt{\frac{\mu}{L}}$. Then for Acc-CLADMM, we have

$$l_{k+1} \leq \left(1 - \sqrt{\frac{\mu}{L}}\right) l_k, \tag{20}$$

where

$$\begin{aligned} l_k = (1-\theta) \sum_i \left(f_i(\tilde{\boldsymbol{x}}_i^k) - f_i(\boldsymbol{x}_i^*) + \langle \boldsymbol{\lambda}_i^*, \tilde{\boldsymbol{x}}_i^k - \tilde{\boldsymbol{z}}^k \rangle \right) \\ + \frac{\theta^2}{2\alpha} \sum_i \|\boldsymbol{x}_i^k - \boldsymbol{x}_i^*\|^2 + \frac{1}{2\beta} \|\boldsymbol{\lambda}_i^k - \boldsymbol{\lambda}^*\|^2. \end{aligned} \tag{21}$$

Complexity Comparisons

We can find that Acc-CLADMM is faster than CLAMM with a better dependence on the condition number L/μ. CADMM has the same convergence rate as the Acc-CLADMM. However, CADMM may need to solve subproblem iteratively at each iteration, while Acc-CLADMM only performs a gradient descent update.

CADMM	CLADMM	Acc-CLADMM
$\mathcal{O}(\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon})$	$\mathcal{O}(\frac{L}{\mu}\log\frac{1}{\epsilon})$	$\mathcal{O}(\sqrt{\frac{L}{\mu}}\log\frac{1}{\epsilon})$

Outline

I Centralized Distributed Optimization

II Decentralized Distributed Optimization

III Asynchronous Distributed ADMM

IV Nonconvex Distributed ADMM

V ADMM with Generally Linear Constraints

References

Decentralized Optimization

Now we consider the decentralized topology. In this case, we cannot introduce the variable $\boldsymbol{z}$ since there is no central node any more.

Denote $\mathcal{E}$ as the set of edges. Assume that all the nodes are ordered from 1 to m. For any two nodes i and j, if i and j are directly connected in the network and $i < j$, we say $(i, j) \in \mathcal{E}$.

To simplify the presentation, we order the edges from 1 to $|\mathcal{E}|$. For each node i, we denote $\mathcal{N}_i$ as its neighborhood:

$$\mathcal{N}_i = \{j \mid (i, j) \in \mathcal{E} \text{ or } (j, i) \in \mathcal{E}\}, \tag{22}$$

and $d_i = |\mathcal{N}_i|$ as its degree.

Decentralized Optimization

Introduce auxiliary variables $\boldsymbol{z}_{ij}$ if $(i,j) \in \mathcal{E}$. Then we can reformulate (1) as follows:

$$\min_{\boldsymbol{x}_i, \boldsymbol{z}_{ij}} \sum_{i=1}^{m} f_i(\boldsymbol{x}_i) \quad s.t. \quad \boldsymbol{x}_i = \boldsymbol{z}_{ij}, \boldsymbol{x}_j = \boldsymbol{z}_{ij}, \forall (i,j) \in \mathcal{E}. \tag{23}$$

That is to say, each variable $\boldsymbol{x}_i$ corresponds to one node, while each variable $\boldsymbol{z}_{ij}(i<j)$ corresponds to one edge. The augmented Lagrangian function is

$$\begin{aligned} L_\beta(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}) = \sum_i f_i(\boldsymbol{x}_i) + \sum_{(i,j)\in\mathcal{E}} \Big(& \langle \boldsymbol{\lambda}_{ij}, \boldsymbol{x}_i - \boldsymbol{z}_{ij} \rangle + \langle \boldsymbol{\gamma}_{ij}, \boldsymbol{x}_j - \boldsymbol{z}_{ij} \rangle \\ & + \frac{\beta}{2} \|\boldsymbol{x}_i - \boldsymbol{z}_{ij}\|^2 + \frac{\beta}{2} \|\boldsymbol{x}_j - \boldsymbol{z}_{ij}\|^2 \Big). \end{aligned} \tag{24}$$

DADMM

We can use the vanilla ADMM to solve it. The iterations are:

$$\boldsymbol{x}_i^{k+1} = \underset{\boldsymbol{x}_i}{\operatorname{argmin}} \Bigg[f_i(\boldsymbol{x}_i) + \sum_{j:(i,j)\in\mathcal{E}} \Big(\langle \boldsymbol{\lambda}_{ij}^k, \boldsymbol{x}_i - \boldsymbol{z}_{ij}^k \rangle + \frac{\beta}{2} \|\boldsymbol{x}_i - \boldsymbol{z}_{ij}^k\|^2 \Big) + \sum_{j:(i,j)\in\mathcal{E}} \Big(\langle \boldsymbol{\gamma}_{ji}^k, \boldsymbol{x}_i - \boldsymbol{z}_{ji}^k \rangle + \frac{\beta}{2} \|\boldsymbol{x}_i - \boldsymbol{z}_{ji}^k\|^2 \Big) \Bigg], \tag{25}$$

$$\begin{aligned} \boldsymbol{z}_{ij}^{k+1} &= \underset{\boldsymbol{z}_{ij}}{\operatorname{argmin}} \Big(- \langle \boldsymbol{\lambda}_{ij}^k + \boldsymbol{\gamma}_{ij}^k, \boldsymbol{z}_{ij} \rangle + \frac{\beta}{2} \|\boldsymbol{x}_i^{k+1} - \boldsymbol{z}_{ij}\|^2 \\ &\quad + \frac{\beta}{2} \|\boldsymbol{x}_j^{k+1} - \boldsymbol{z}_{ij}\|^2 \Big) \\ &= \frac{1}{2\beta} (\boldsymbol{\lambda}_{ij}^k + \boldsymbol{\gamma}_{ij}^k) + \frac{1}{2} (\boldsymbol{x}_i^{k+1} + \boldsymbol{x}_j^{k+1}). \end{aligned} \tag{26}$$

DADMM

We can use the vanilla ADMM to solve it. The iterations are (cont'd):

$$\boldsymbol{\lambda}_{ij}^{k+1} = \boldsymbol{\lambda}_{ij}^{k} + \beta(\boldsymbol{x}_i^{k+1} - \boldsymbol{z}_{ij}^{k+1}), \tag{27}$$

$$\boldsymbol{\gamma}_{ij}^{k+1} = \boldsymbol{\gamma}_{ij}^{k} + \beta(\boldsymbol{x}_j^{k+1} - \boldsymbol{z}_{ij}^{k+1}). \tag{28}$$

Actually (25) $\sim$ (28) can be simplified. Summing (27) and (28) and using (26), we have

$$\boldsymbol{\lambda}_{ij}^{k+1} + \boldsymbol{\gamma}_{ij}^{k+1} = \mathbf{0}, \forall k \geq 0. \tag{29}$$

Initialize $\boldsymbol{\lambda}_{ij}^{0} = \boldsymbol{\gamma}_{ij}^{0} = \mathbf{0}$ we have

$$\boldsymbol{\lambda}_{ij}^{k} + \boldsymbol{\gamma}_{ij}^{k} = \mathbf{0}, \forall k \geq 0. \tag{30}$$

DADMM

Plugging (30) into (26) we have

$$\boldsymbol{z}_{ij}^{k+1} = \frac{1}{2}(\boldsymbol{x}_i^{k+1} + \boldsymbol{x}_j^{k+1}), \forall k \geq 0. \tag{31}$$

Similarly, we may initialize $\boldsymbol{z}_{ij}^0 = \frac{1}{2}(\boldsymbol{x}_i^0 + \boldsymbol{x}_j^0)$. Combing (31) and (27) we have

$$\boldsymbol{\lambda}_{ij}^{k+1} = \boldsymbol{\lambda}_{ij}^{k} + \frac{\beta}{2}(\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_j^{k+1}) \tag{32}$$

Thus we have

$$\boldsymbol{\lambda}_{ij}^{k+1} = \beta \sum_{t=1}^{k+1} \frac{1}{2}(\boldsymbol{x}_i^t - \boldsymbol{x}_j^t). \tag{33}$$

Similarly we have

$$\boldsymbol{\gamma}_{ij}^{k+1} = \beta \sum_{t=1}^{k+1} \frac{1}{2}(\boldsymbol{x}_j^t - \boldsymbol{x}_i^t). \tag{34}$$

DADMM

Note that we only define $\boldsymbol{\lambda}_{ij}$, $\boldsymbol{\gamma}_{ij}$, and $\boldsymbol{z}_{ij}$ for $i < j$. Now we define

$$\boldsymbol{\lambda}_{ij} \equiv \boldsymbol{\gamma}_{ji} \text{ and } \boldsymbol{z}_{ij} \equiv \boldsymbol{z}_{ji} \text{ for } i > j. \tag{35}$$

Then (31), (32) and (33) hold for both $i < j$ and $i > j$.

DADMM

Thus, we can simplify (25) to

$$
\begin{aligned}
\boldsymbol{x}_i^{k+1} &= \operatorname*{argmin}_{\boldsymbol{x}_i} \Big[f_i(\boldsymbol{x}_i) + \sum_{j:(i,j)\in\mathcal{E}} \Big(\langle \boldsymbol{\lambda}_{ij}^k - \beta \boldsymbol{z}_{ij}^k, \boldsymbol{x}_i \rangle + \frac{\beta}{2} \|\boldsymbol{x}_i\|^2 \Big) \\
&+ \sum_{j:(j,i)\in\mathcal{E}} \Big(\langle \boldsymbol{\gamma}_{ji}^k - \beta \boldsymbol{z}_{ji}^k, \boldsymbol{x}_i \rangle + \frac{\beta}{2} \|\boldsymbol{x}_i\|^2 \Big) \Big] \\
&= \operatorname*{argmin}_{\boldsymbol{x}_i} \Big[f_i(\boldsymbol{x}_i) + \sum_{j\in\mathcal{N}_i} \Big(\langle \boldsymbol{\lambda}_{ij}^k - \beta \boldsymbol{z}_{ij}^k, \boldsymbol{x}_i \rangle + \frac{\beta}{2} \|\boldsymbol{x}_i\|^2 \Big) \Big] \\
&= \operatorname*{argmin}_{\boldsymbol{x}_i} \Big[f_i(\boldsymbol{x}_i) + \sum_{j\in\mathcal{N}_i} \Big(\langle \boldsymbol{\lambda}_{ij}^k - \beta \boldsymbol{z}_{ij}^k + \beta \boldsymbol{x}_i^k, \boldsymbol{x}_i \rangle + \frac{\beta}{2} \|\boldsymbol{x}_i - \boldsymbol{x}_i^k\|^2 \Big) \Big] \\
&= \operatorname*{argmin}_{\boldsymbol{x}_i} \Big[f_i(\boldsymbol{x}_i) + \sum_{j\in\mathcal{N}_i} \Big(\langle \boldsymbol{\lambda}_{ij}^k + \frac{\beta}{2}(\boldsymbol{x}_i^k - \boldsymbol{x}_j^k), \boldsymbol{x}_i \rangle + \frac{\beta}{2} \|\boldsymbol{x}_i - \boldsymbol{x}_i^k\|^2 \Big) \Big].
\end{aligned}
\tag{36}
$$

DADMM

Denote $\mathbf{L} \in \mathbf{R}^{m \times m}$ as the Laplacian matrix and $\mathbf{D}$ as the diagonal degree matrix with $\mathbf{D}_{ii} = d_i$. Note that $\mathbf{L}$ is symmetric and satisfies $\mathbf{0} \preceq \mathbf{L} \preceq 2\mathbf{D}$. Define

$$\mathbf{X} = \begin{pmatrix} \boldsymbol{x}_1^T \\ \vdots \\ \boldsymbol{x}_m^T \end{pmatrix} \in \mathbf{R}^{m \times d}, f(\mathbf{X}) = \sum_{i=1}^{m} f_i(\boldsymbol{x}_i), \tag{37}$$

$$\boldsymbol{v}_i = \sum_{j \in \mathcal{N}_i} \boldsymbol{\lambda}_{ij}, \text{ and } \boldsymbol{\Theta} = \begin{pmatrix} \boldsymbol{v}_1^T \\ \vdots \\ \boldsymbol{v}_m^T \end{pmatrix} \in \mathbf{R}^{m \times d}. \tag{38}$$

Then we have

$$\mathbf{L}_i^T \mathbf{X} = d_i \boldsymbol{x}_i^T - \sum_{j \in \mathcal{N}_i} \boldsymbol{x}_j^T, \tag{39}$$

where $\mathbf{L}_i$ is the i-th column of $\mathbf{L}$.

DADMM

With $\mathbf{L}$ and $\boldsymbol{v}_i$, (36) can be written as

$$\begin{aligned}\boldsymbol{x}_i^{k+1} &= \operatorname*{argmin}_{\boldsymbol{x}_i} \Big[f_i(\boldsymbol{x}_i) + \langle \boldsymbol{v}_i^k, \boldsymbol{x}_i \rangle \\ &\quad + \frac{\beta}{2} \langle \sum_{j \in \mathcal{N}_i} \mathbf{L}_{ij} \boldsymbol{x}_j^k, \boldsymbol{x}_i \rangle + \frac{\beta d_i}{2} \|\boldsymbol{x}_i - \boldsymbol{x}_i^k\|^2 \Big] \\ &= \operatorname{prox}_{(\beta d_i)^{-1} f_i} \Big(\boldsymbol{x}_i^k - \frac{1}{\beta d_i} \Big(\boldsymbol{v}_i^k + \frac{\beta}{2} \sum_{j \in \mathcal{N}_i} \mathbf{L}_{ij} \boldsymbol{x}_j^k \Big) \Big), \quad \forall i \in [m]. \end{aligned} \tag{40}$$

Summing (32) over $j \in \mathcal{N}_i$, we have

$$\boldsymbol{v}_i^{k+1} = \boldsymbol{v}_i^k + \frac{\beta}{2} \sum_{j \in \mathcal{N}_i} \mathbf{L}_{ij} \boldsymbol{x}_j^{k+1}, \forall i \in [m]. \tag{41}$$

DADMM

Each node i should do —— In each iteration k:

1. Update $\boldsymbol{x}_i^{k+1}$ by (40)
2. Send $\boldsymbol{x}_i^{k+1}$ to its neighbors
3. Wait until receiving $\boldsymbol{x}_j^{k+1}$ from all its neighbors, $j \in \mathcal{N}_i$
4. Update $\boldsymbol{v}_i^{k+1}$ by (41)

We call this method Decentralized ADMM (*DADMM*).

DADMM'

(40) and (41) can be written in a compact form:

$$\mathbf{X}^{k+1} = \underset{\mathbf{x}}{\operatorname{argmin}} \Big(f(\mathbf{X}) + \langle \mathbf{\Theta}^k + \frac{\beta}{2}\mathbf{L}\mathbf{X}^k, \mathbf{X} \rangle + \frac{\beta}{2} \| \sqrt{\mathbf{D}}(\mathbf{X} - \mathbf{X}^k) \|^2 \Big), \tag{42}$$

$$\mathbf{\Theta}^{k+1} = \mathbf{\Theta}^k + \frac{\beta}{2}\mathbf{L}\mathbf{X}^{k+1} = \mathbf{\Theta}^k + \beta \mathbf{W}^2 \mathbf{X}^{k+1}, \tag{43}$$

where $\mathbf{W} = \sqrt{\mathbf{L}/2}$. Let $\mathbf{\Theta}^0 \in \operatorname{span}(\mathbf{W}^2)$, we know that

$$\mathbf{\Theta}^k \in \operatorname{span}(\mathbf{W}^2), \forall k \geq 0, \tag{44}$$

and *there exists* $\mathbf{\Omega}^k$ *such that* $\mathbf{\Theta}^k = \mathbf{W}\mathbf{\Omega}^k$.

DADMM'

(42) and (43) can be further written as

$$\begin{aligned}\mathbf{X}^{k+1} &= \underset{\mathbf{X}}{\operatorname{argmin}} \Big(f(\mathbf{X}) + \langle \mathbf{\Omega}^k, \mathbf{W}\mathbf{X} \rangle \\ &+ \beta \langle \mathbf{W}^2\mathbf{X}^k, \mathbf{X} \rangle + \frac{\beta}{2} \|\sqrt{\mathbf{D}}(\mathbf{X} - \mathbf{X}^k)\|^2 \\ &= \underset{\mathbf{X}}{\operatorname{argmin}} \Big(f(\mathbf{X}) + \langle \mathbf{\Omega}^k, \mathbf{W}\mathbf{X} \rangle + \frac{\beta}{2} \|\mathbf{W}\mathbf{X}\|^2 + D_\Psi(\mathbf{X}, \mathbf{X}^k) \Big), \end{aligned} \tag{45}$$

$$\mathbf{\Omega}^{k+1} = \mathbf{\Omega}^k + \beta \mathbf{W}\mathbf{X}^{k+1}, \tag{46}$$

where

$$\Psi(\mathbf{X}) = \frac{\beta}{2} \|\sqrt{\mathbf{D}}\mathbf{X}\|^2 - \frac{\beta}{2} \|\mathbf{W}\mathbf{X}\|^2. \tag{47}$$

DADMM'

We can find that (45) and (46) is equal to using the LADMM to solve

$$\min_{\mathbf{X}} f(\mathbf{X}), \quad s.t. \quad \mathbf{WX} = \mathbf{0}. \tag{48}$$

We call (45) and (46) with general Ψ settings *DADMM'*.
DADMM' is not implementable in the distributed manner because $\mathbf{W} = \sqrt{\mathbf{L}/2}$, which requries the overall graph structure information.

Convergence Rate of DADMM'

Theorem II.1

Assume that *each f_i is μ-strongly convex and L-smooth*, $i \in [m]$, and $\Psi(\boldsymbol{y})$ is convex and L_Ψ-smooth. Initialize $\boldsymbol{\Omega}^0 = \mathbf{0}$. The for DADMM', we have

$$\begin{aligned}
&\frac{1}{2\beta}\|\boldsymbol{\Omega}^{k+1} - \boldsymbol{\Omega}^*\|^2 + \frac{\beta}{2}\|\mathbf{W}\mathbf{X}^{k+1} - \mathbf{W}\mathbf{X}^*\|^2 + D_\Psi(\mathbf{X}^*, \mathbf{X}^{k+1}) \\
&\qquad \le \left(1 + \frac{1}{3}\min\left\{\frac{\beta\sigma_{\mathbf{L}}}{2(L+L_\Psi)}, \frac{\mu}{\beta\|\mathbf{W}\|^2}, \frac{\mu}{L_\Psi}\right\}\right)^{-1} \\
&\times\left(\frac{1}{2\beta}\|\boldsymbol{\Omega}^{k} - \boldsymbol{\Omega}^*\|^2 + \frac{\beta}{2}\|\mathbf{W}\mathbf{X}^{k} - \mathbf{W}\mathbf{X}^*\|^2 + D_\Psi(\mathbf{X}^*, \mathbf{X}^{k})\right),
\end{aligned} \tag{49}$$

where $\sigma_{\mathbf{L}}$ is the smallest positive eigenvalue of $\mathbf{L}$.

Convergence Rate of DADMM

For DLADMM', when Ψ is set as (47) and $L_\Psi = \beta d_{max}$, where $d_{max} = \max_i\{d_i\}$. DLADMM' reduces to DLADMM, and

$$\|\mathbf{W}\|^2 \leq \frac{1}{2}\|\mathbf{L}\| \leq \|\mathbf{D}\| \leq d_{max}. \tag{50}$$

We thus have the following convergence result.

Theorem II.2

Assume that *each* f_i *is* μ*-strongly convex and* L*-smooth*, $i \in [m]$. Initialize $\mathbf{\Omega}^0 = \mathbf{0}$ and let $\beta = \mathcal{O}(\sqrt{\frac{\mu L}{\sigma_{\mathbf{L}} d_{max}}})$. Then DADMM needs $\mathcal{O}((\sqrt{\frac{L d_{max}}{\mu \sigma_{\mathbf{L}}}} + \frac{d_{max}}{\sigma_{\mathbf{L}}}) \log \frac{1}{\epsilon})$ iterations to find an ϵ-approximate solution $(\mathbf{X}, \mathbf{\Omega})$, i.e.,

$$\frac{1}{2\beta}\|\mathbf{\Omega} - \mathbf{\Omega}^*\|^2 + \frac{\beta}{2}\|\mathbf{WX} - \mathbf{WX}^*\|^2 + D_\Psi(\mathbf{X}^*, \mathbf{X}) \leq \epsilon. \tag{51}$$

DLADMM

LADMM can also be used to solve (23). We may linearize f_i:

$$
\begin{aligned}
\boldsymbol{x}_i^{k+1} = \underset{\boldsymbol{x}_i}{\operatorname{argmin}} \Big[& \langle \nabla f_i(\boldsymbol{x}_i^k), \boldsymbol{x}_i - \boldsymbol{x}_i^k \rangle + \frac{L}{2} \|\boldsymbol{x}_i - \boldsymbol{x}_i^k\|^2 \\
& + \sum_{j:(i,j)\in\mathcal{E}} \left(\langle \boldsymbol{\lambda}_{ij}^k, \boldsymbol{x}_i - \boldsymbol{z}_{ij}^k \rangle + \frac{\beta}{2} \|\boldsymbol{x}_i - \boldsymbol{z}_{ij}^k\|^2 \right) \\
& + \sum_{j:(j,i)\in\mathcal{E}} \left(\langle \boldsymbol{\gamma}_{ji}^k, \boldsymbol{x}_i - \boldsymbol{z}_{ji}^k \rangle + \frac{\beta}{2} \|\boldsymbol{x}_i - \boldsymbol{z}_{ji}^k\|^2 \right) \Big] \\
= \underset{\boldsymbol{x}_i}{\operatorname{argmin}} \Big[& \langle \nabla f_i(\boldsymbol{x}_i^k), \boldsymbol{x}_i - \boldsymbol{x}_i^k \rangle + \frac{L}{2} \|\boldsymbol{x}_i - \boldsymbol{x}_i^k\|^2 \\
& + \sum_{j\in\mathcal{N}_i} \left(\langle \boldsymbol{\lambda}_{ij}^k + \frac{\beta}{2}(\boldsymbol{x}_i^k - \boldsymbol{x}_j^k), \boldsymbol{x}_i \rangle + \frac{\beta}{2} \|\boldsymbol{x}_i - \boldsymbol{x}_i^k\|^2 \right) \Big] \\
= \boldsymbol{x}_i^k - \frac{1}{L + \beta d_i} & \left\{ \nabla f_i(\boldsymbol{x}_i^k) + \sum_{j\in\mathcal{N}_i} \left[\boldsymbol{\lambda}_{ij}^k + \frac{\beta}{2}(\boldsymbol{x}_i^k - \boldsymbol{x}_j^k) \right] \right\}. \qquad (52)
\end{aligned}
$$

DLADMM

Similar to (45) and (46), we can write these iterations in a compact form:

$$\begin{aligned}\mathbf{X}^{k+1} &= \operatorname*{argmin}_{\mathbf{X}} \Big(\langle \nabla f(\mathbf{X}^k), \mathbf{X} \rangle + \frac{L}{2}\|\mathbf{X} - \mathbf{X}^k\|^2 \\ &\quad + \langle \mathbf{\Omega}^k, \mathbf{W}\mathbf{X} \rangle + \beta \langle \mathbf{W}^2\mathbf{W}^k, \mathbf{X} \rangle + \frac{\beta}{2}\|\sqrt{\mathbf{D}}(\mathbf{X} - \mathbf{X}^k)\|^2 \Big) \\ &= \mathbf{X}^k - (L\mathbf{I} + \beta\mathbf{D})^{-1}\big(\beta\mathbf{W}^2\mathbf{X}^k + \nabla f(\mathbf{X}^k) + \mathbf{W}\mathbf{\Omega}^k\big), \quad (53) \\ \mathbf{\Omega}^{k+1} &= \mathbf{\Omega}^k + \beta\mathbf{W}\mathbf{X}^{k+1}, \quad (54)\end{aligned}$$

which is also a special case of DADMM' with

$$\Phi(\mathbf{X}) = \frac{L}{2}\|\mathbf{X}\|^2 - f(\mathbf{X}) + \frac{\beta}{2}\|\sqrt{\mathbf{D}}\mathbf{X}\|^2 - \frac{\beta}{2}\|\mathbf{W}\mathbf{X}\|^2 \quad (55)$$

and

$$L_\Phi = L + \beta d_{max}. \quad (56)$$

DLADMM

Each node i should do —— In each iteration k:

1. Update $\boldsymbol{x}_i^{k+1}$ by (52)
2. Send $\boldsymbol{x}_i^{k+1}$ to its neighbors
3. Wait until receiving $\boldsymbol{x}_j^{k+1}$ from all its neighbors, $j \in \mathcal{N}_i$
4. Update $\boldsymbol{v}_i^{k+1}$ by (41)

We call (52), (26) $\sim$ (28) *DLADMM*.

Convergence Rate of DLADMM

We have the following convergence result for DLADMM.

Theorem II.3

Assume that *each f_i is μ-strongly convex and L-smooth*, $i \in [m]$. Initialize $\mathbf{\Omega}^0 = \mathbf{0}$ and let $\beta = \mathcal{O}(\sqrt{\frac{\mu L}{\sigma_{\mathbf{L}} d_{max}}})$. Then DLADMM needs $\mathcal{O}((\frac{L}{\mu} + \frac{d_{max}}{\sigma_{\mathbf{L}}}) \log \frac{1}{\epsilon})$ iterations to find an ϵ-approximate solution $(\mathbf{X}, \mathbf{\Omega})$, i.e.,

$$\frac{1}{2\beta}\|\mathbf{\Omega} - \mathbf{\Omega}^*\|^2 + \frac{\beta}{2}\|\mathbf{WX} - \mathbf{WX}^*\|^2 + D_\Psi(\mathbf{X}^*, \mathbf{X}) \leq \epsilon. \quad (57)$$

Acc-DLADMM

Accelerated Linearized ADMM can also be used to solve (23). Based on Acc-LADMM-3, we have the following iterations:

$$\mathbf{Y}^k = \theta \mathbf{X}^k + (1-\theta)\tilde{\mathbf{X}}^k, \tag{58}$$

$$\mathbf{X}^{k=1} = \frac{1}{\frac{\theta}{\alpha}+\mu}\Big[\mu \mathbf{Y}^k + \frac{\theta}{\alpha}\mathbf{X}^k - \big(\nabla f(\mathbf{Y}^k) + \mathbf{W}\boldsymbol{\Omega}^k + \beta\theta \mathbf{W}^2\mathbf{X}^k\big)\Big], \tag{59}$$

$$\tilde{\mathbf{X}}^{k+1} = \theta \mathbf{X}^{k+1} + (1-\theta)\tilde{\mathbf{X}}^k, \tag{60}$$

$$\boldsymbol{\Omega}^{k+1} = \boldsymbol{\Omega}^k + \beta\theta \mathbf{W}\mathbf{X}^{k+1}. \tag{61}$$

The above formulas are written in compact form.

Acc-DLADMM

We call it *Acc-DLADMM* —— in the distributed form:

$$\boldsymbol{y}_i^k = \theta \boldsymbol{x}_i^k + (1-\theta)\tilde{\boldsymbol{x}}_i^k, \tag{62}$$

$$\boldsymbol{x}_i^{k=1} = \frac{1}{\frac{\theta}{\alpha}+\mu}\Big[\mu \boldsymbol{y}_i^k + \frac{\theta}{\alpha}\boldsymbol{x}_i^k - \big(\nabla f_i(\boldsymbol{y}_i^k) + \boldsymbol{v}_i^k + \frac{\beta\theta}{2}\sum_{j\in\mathcal{N}_i}\mathbf{L}_{ij}\boldsymbol{x}_j^k\big)\Big], \tag{63}$$

$$\tilde{\boldsymbol{x}}_i^{k+1} = \theta \boldsymbol{x}_i^{k+1} + (1-\theta)\tilde{\boldsymbol{x}}_i^k, \tag{64}$$

$$\boldsymbol{v}_i^{k+1} = \boldsymbol{v}_i^k + \frac{\beta\theta}{2}\sum_{j\in\mathcal{N}_i}\mathbf{L}_{ij}\boldsymbol{x}_j^{k+1}. \tag{65}$$

Each node i should do —— In each iteration k:

1. Update $\boldsymbol{y}_i^k, \boldsymbol{x}_i^{k+1}, \tilde{\boldsymbol{x}}_i^{k+1}$ by (62) $\sim$ (64), respectively
2. Send $\boldsymbol{x}_i^{k+1}$ to its neighbors
3. Wait until receiving $\boldsymbol{x}_j^{k+1}$ from all its neighbors, $j \in \mathcal{N}_i$
4. Update $\boldsymbol{v}_i^{k+1}$ by (65)

Convergence Rate of Acc-DLADMM

Theorem II.4

Suppose that each $f_i(\boldsymbol{x}_i)$ is μ-strongly convex and L-smooth, $i \in [m]$. Assume that $\frac{2d_{max}}{\sigma_{\mathbf{L}}} \leq \frac{L}{\mu}$, where $\sigma_{\mathbf{L}}$ is the smallest non-zero sigular value of $\mathbf{L}$. Let

$$\alpha = \frac{1}{4L}, \beta = \frac{L}{d_{max}}, \text{ and } \theta = \sqrt{\frac{2\mu d_{max}}{L\sigma_{\mathbf{L}}}}. \tag{66}$$

Convergence Rate of Acc-DLADMM

Similar to Theorem III.1 of *ADMM Slide 2*, we have the following convergence rate for Acc-DLADMM.

Theorem II.4 (cont'd)

Then for Acc-DLADMM, we have

$$l_{k+1} \leq \mathcal{O}\left(1 - \sqrt{\frac{\mu\sigma_{\mathbf{L}}}{2Ld_{max}}}\right) l_k, \tag{67}$$

where

$$\begin{aligned} l_k = (1-\theta)\Big(f(\tilde{\mathbf{X}}^k) - f(\mathbf{X}^*) + \langle \mathbf{\Omega}^*, \mathbf{W}\tilde{\mathbf{X}}^k \rangle\Big) \\ + \frac{\theta^2}{2\alpha}\|\mathbf{X}^k - \mathbf{X}^*\|^2 + \frac{1}{2\beta}\|\mathbf{\Omega}^k - \mathbf{\Omega}^*\|^2. \end{aligned} \tag{68}$$

Complexity Comparisons

For decentralized optimization, the algorithm complexity are listed as follows.

- DADMM: $\mathcal{O}\left(\left(\sqrt{\frac{Ld_{max}}{\mu\sigma_{\mathbf{L}}}}+\frac{d_{max}}{\sigma_{\mathbf{L}}}\right)\log\frac{1}{\epsilon}\right)$
- DLADMM: $\mathcal{O}\left(\left(\frac{L}{\mu}+\frac{d_{max}}{\sigma_{\mathbf{L}}}\right)\log\frac{1}{\epsilon}\right)$
- Acc-DLADMM: $\mathcal{O}\left(\sqrt{\frac{Ld_{max}}{\mu\sigma_{\mathbf{L}}}}\log\frac{1}{\epsilon}\right)$

Outline

I Centralized Distributed Optimization

II Decentralized Distributed Optimization

III Asynchronous Distributed ADMM

IV Nonconvex Distributed ADMM

V ADMM with Generally Linear Constraints

References

Asynchronous Distributed ADMM

Now let's go back to the centralized optimization. CADMM, CLADMM, and Acc-CLADMM are executed in a synchronous manner. That is, the master needs to wait for all the workers to finish their updates before it can proceed. When the workers have different delays, the master has to wait for the slowest worker before the next iteration, i.e., the system proceeds at the pace of the slowest worker.

In the asynchronous ADMM, the master does not wait for all the workers, but proceeds as long as it receives information from a partial set of workers instead.

Asynchronous Distributed ADMM

In the following, we describe how to adapt CADMM to the asynchronous version and gives its convergence analysis.

We denote the partial set at iteration k as $\mathcal{A}^k$, and $\mathcal{A}_c^k$ as the complementary set, which means the set of workers whose information does not arrive at iteration k. We suseue α to lower bound the size of $\mathcal{A}^k$. In the asynchronous ADMM, we often require that the master has to receive the updates from every worker at least once in every τ iterations. That is, we do not allow some workers to be absent for a long time. So we make the following bounded delay assumption.

Assumption III.1

The maximum tolerable delay for all i and k is upper bounded.

Denote the upper bound as τ, then it must be that for every i,

$$i \in \mathcal{A}^k \cup \mathcal{A}^{k-1} \cup \cdots \cup \mathcal{A}^{\max\{k-\tau+1,0\}}. \tag{69}$$

Async-ADMM-M

The master works as follows.

1. Initialize $\tilde{d}_1^1 = \cdots = \tilde{d}_m^1 = 0$
2. **for** $k = 1, 2, \ldots$ **do**
   - 2.1 Wait until receiving $\hat{\boldsymbol{x}}_i^k$ and $\hat{\boldsymbol{\lambda}}_i^k$ from workers $i \in \mathcal{A}^k$ such that $|\mathcal{A}^k| \geq \alpha$ and $\tilde{d}_j^k < \tau - 1$ for all $j \in \mathcal{A}_c^k$
   - 2.2 $\boldsymbol{x}_i^{k+1} = \begin{cases} \hat{\boldsymbol{x}}_i^k & \forall i \in \mathcal{A}^k \\ \boldsymbol{x}_i^k & \forall i \in \mathcal{A}_c^k \end{cases}$, $\boldsymbol{\lambda}_i^{k+1} = \begin{cases} \hat{\boldsymbol{\lambda}}_i^k & \forall i \in \mathcal{A}^k \\ \boldsymbol{\lambda}_i^k & \forall i \in \mathcal{A}_c^k \end{cases}$, $\tilde{d}_i^{k+1} = \begin{cases} 0 & \forall i \in \mathcal{A}^k \\ \tilde{d}_i^k + 1 & \forall i \in \mathcal{A}_c^k \end{cases}$
   - 2.3 $\boldsymbol{z}^{k+1} = \operatorname{argmin}_{\boldsymbol{z}} \left[\sum_{i=1}^m \left(\langle \boldsymbol{\lambda}_i^{k+1}, \boldsymbol{x}_i^{k+1} - \boldsymbol{z} \rangle + \frac{\beta}{2} \|\boldsymbol{x}_i^{k+1} - \boldsymbol{z}\|^2 \right) + \frac{\rho}{2} \|\boldsymbol{z} - \boldsymbol{z}^k\|^2 \right] = \frac{1}{\rho + m\beta} \left[\rho \boldsymbol{z}^k + \sum_{i=1}^m (\boldsymbol{\lambda}_i^{k+1} + \beta \boldsymbol{x}_i^{k+1}) \right]$
   - 2.4 Broadcast $\boldsymbol{z}^{k+1}$ to the workers in $\mathcal{A}^k$

Async-ADMM-W

The i-th worker work as follows.

1. Initialize $\hat{\boldsymbol{x}}_i^0$ and $\hat{\boldsymbol{\lambda}}_i^0$, $i \in [m]$
2. **for** $k_i = 1, 2, \ldots$ **do**
   - 2.1 Wait until receiving $\boldsymbol{z}$ from the master
   - 2.2 $\hat{\boldsymbol{x}}_i^{k_i+1} = \operatorname{argmin}_{\boldsymbol{x}_i} \left(f_i(\boldsymbol{x}_i) + \langle \hat{\boldsymbol{\lambda}}_i^{k_i}, \boldsymbol{x}_i - \boldsymbol{z} \rangle + \frac{\beta}{2} \|\boldsymbol{x}_i - \boldsymbol{z}\|^2 \right) = \operatorname{prox}_{\beta^{-1} f_i}(\boldsymbol{z} - \frac{1}{\beta} \hat{\boldsymbol{\lambda}}_i^{k_i})$
   - 2.3 $\hat{\boldsymbol{\lambda}}_i^{k_i+1} = \hat{\boldsymbol{\lambda}}_i^{k_i} + \beta(\hat{\boldsymbol{x}}_i^{k_i+1} - \boldsymbol{z})$
   - 2.4 Send $\hat{\boldsymbol{x}}_i^{k_i+1}$ and $\hat{\boldsymbol{\lambda}}_i^{k_i+1}$ to the master

Note that we introduce d_i, the amount of delay, for each worker such that the bounded delay assumption holds. The master must wait if there exists one worker with $\tilde{d}_i = \tau - 1$.

We name the two subprocedures as *Async-ADMM-M* and *Async-ADMM-W*, respectively.

Existence of Convergence

To simplify the analysis, we rewrite the method from the master's point of view:

$$\boldsymbol{x}_i^{k+1} = \begin{cases} \operatorname{argmin}_{\boldsymbol{x}_i} \left(f_i(\boldsymbol{x}_i) + \langle \boldsymbol{\lambda}_i^{\bar{k}_i+1}, \boldsymbol{x}_i \rangle + \frac{\beta}{2} \|\boldsymbol{x}_i - \boldsymbol{z}^{\bar{k}_i+1}\|^2 \right) & i \in \mathcal{A}^k \\ \boldsymbol{x}_i^k & i \in \mathcal{A}_c^k, \end{cases} \tag{70}$$

$$\boldsymbol{\lambda}_i^{k+1} = \begin{cases} \boldsymbol{\lambda}_i^{\bar{x}_i+1} + \beta(\boldsymbol{x}_i^{k+1} - \boldsymbol{z}^{\bar{k}_i+1}) & i \in \mathcal{A}^k \\ \boldsymbol{\lambda}_i^k & i \in \mathcal{A}_c^k, \end{cases} \tag{71}$$

$$\boldsymbol{z}^{k+1} = \operatorname*{argmin}_{\boldsymbol{z}} \Bigg[\sum_{i=1}^{m} \left(\langle \boldsymbol{\lambda}_i^{k+1}, \boldsymbol{x}_i^{k+1} - \boldsymbol{z} \rangle + \frac{\beta}{2} \|\boldsymbol{x}_i^{k+1} - \boldsymbol{z}\|^2 \right) + \frac{\rho}{2} \|\boldsymbol{z} - \boldsymbol{z}^k\|^2 \Bigg], \tag{72}$$

where $\bar{k}_i$ is the last iteration before iteration k for which worker $i \in \mathcal{A}^k$ arrives, i.e., $i \in \mathcal{A}^{\bar{k}_i}$.

Existence of Convergence

Thus, for all workers $i \in \mathcal{A}^k$, we have

$$\boldsymbol{x}_i^{\bar{k}_i+1} = \boldsymbol{x}_i^{\bar{k}_i+2} = \ldots = \boldsymbol{x}_i^k,$$

$$\boldsymbol{\lambda}_i^{\bar{k}_i+1} = \boldsymbol{\lambda}_i^{\bar{k}_i+2} = \ldots = \boldsymbol{\lambda}_i^k, \text{ and} \tag{73}$$

$$\max\{k-\tau, 0\} \leq \bar{k}_i \leq k. \tag{74}$$

For each $i \in \mathcal{A}_c^k$ we denote $\tilde{k}_i$ as the last iteration before iteration k for which worker i arrives, i.e., $i \in \mathcal{A}^{\tilde{k}_i}$. Under the bounded delay assumption, we have

$$\max\{k-\tau+1, 0\} \leq \tilde{k}_i < k. \tag{75}$$

Thus, for all workers $i \in \mathcal{A}_c^k$, we have

$$\boldsymbol{x}_i^{\tilde{k}_i+1} = \boldsymbol{x}_i^{\tilde{k}_i+2} = \ldots = \boldsymbol{x}_i^k = \boldsymbol{x}_i^{k+1}, \tag{76}$$

$$\boldsymbol{\lambda}_i^{\tilde{k}_i+1} = \boldsymbol{\lambda}_i^{\tilde{k}_i+2} = \ldots = \boldsymbol{\lambda}_i^k = \boldsymbol{\lambda}_i^{k+1}. \tag{77}$$

Existence of Convergence

We also denote $\hat{k}_i$ as the last iteration before $\tilde{k}_i$ for which $i \in \mathcal{A}^{\tilde{k}_i}$ arrives, i.e., $i \in \mathcal{A}^{\hat{k}_i}$. We also have

$$\max\{\tilde{k}_i - \tau, 0\} \leq \hat{k}_i < \tilde{k}_i. \tag{78}$$

Thus, for all workers $i \in \mathcal{A}_c^k$, we have

$$\boldsymbol{x}_i^{k+1} = \boldsymbol{x}_i^{\tilde{k}_i+1} = \underset{\boldsymbol{x}_i}{\operatorname{argmin}} \left(f_i(\boldsymbol{x}_i) + \langle \boldsymbol{\lambda}_i^{\hat{k}_i+1}, \boldsymbol{x}_i \rangle + \frac{\beta}{2} \|\boldsymbol{x}_i - \boldsymbol{z}^{\hat{k}_i+1}\|^2 \right), \tag{79}$$

$$\boldsymbol{\lambda}_i^{k+1} = \boldsymbol{\lambda}_i^{\tilde{k}_i+1} = \boldsymbol{\lambda}_i^{\hat{k}_i+1} + \beta(\boldsymbol{x}_i^{\tilde{k}_i+1} - \boldsymbol{z}^{\hat{k}_i+1}), \tag{80}$$

and

$$\boldsymbol{x}_i^{\hat{k}_i+1} = \boldsymbol{x}_i^{\hat{k}_i+2} = \ldots = \boldsymbol{x}_i^{\tilde{k}_i}, \tag{81}$$

$$\boldsymbol{\lambda}_i^{\hat{k}_i+1} = \boldsymbol{\lambda}_i^{\hat{k}_i+2} = \ldots = \boldsymbol{\lambda}_i^{\tilde{k}_i}. \tag{82}$$

Existence of Convergence

Denote $(\boldsymbol{x}_1^*, \ldots, \boldsymbol{x}_m^*, \boldsymbol{z}^*, \boldsymbol{\lambda}_1^*, \ldots, \boldsymbol{\lambda}_m^*)$ to be a KKT point. We have

$$\sum_{i=1}^{m} \boldsymbol{\lambda}_i^* = \mathbf{0}, \boldsymbol{z}^* = \boldsymbol{x}_i^*, \text{ and } \nabla f_i(\boldsymbol{x}_i^*) + \boldsymbol{\lambda}_i^* = \mathbf{0}, i \in [m]. \tag{83}$$

Also denote $f^* = \sum_{i=1}^{m} f_i(\boldsymbol{z}^*)$.

Theorem III.1

Suppose that each $f_i(\boldsymbol{x}_i)$ is μ-strongly convex and L-smooth, $i \in [m]$, and Assumption III.1 holds. Let

$$\beta > \frac{1 + L^2 + \sqrt{(1+L^2)^2 + 8L^2}}{2},$$
$$\rho > \frac{1}{2}\left[m(1+\beta^2)(\tau-1)^2 - m\beta\right]. \tag{84}$$

Existence of Convergence

Theorem III.1 (cont'd)

Suppose that $(\boldsymbol{x}_1^k, \ldots, \boldsymbol{x}_m^k, \boldsymbol{z}^k, \boldsymbol{\lambda}_1^k, \ldots, \boldsymbol{\lambda}_m^k)$ generated by (70) $\sim$ (72) are bounded, then $(\boldsymbol{x}_1^k, \ldots, \boldsymbol{x}_m^k, \boldsymbol{z}^k, \boldsymbol{\lambda}_1^k, \ldots, \boldsymbol{\lambda}_m^k)$ converge to the set of KKT points of (2) in the sense of

$$\sum_{i=1}^{m} \boldsymbol{\lambda}_i^k \to \mathbf{0}, \tag{85}$$

$$\boldsymbol{x}_i^{k+1} - \boldsymbol{z}^{k+1} \to \mathbf{0}, \tag{86}$$

$$\nabla f_i(\boldsymbol{x}_i^{k+1}) + \boldsymbol{\lambda}_i^{k+1} = \mathbf{0}, i \in [m]. \tag{87}$$

Linear Convergence Rate of Async-ADMM

Theorem III.2

Suppose that each $f_i(\boldsymbol{x}_i)$ is μ-strongly convex and L-smooth, $i \in [m]$, and Assumption III.1 holds. Let β and ρ be large enough such that

$$8m(\beta - \mu) \leq \rho, \tag{88}$$

$$\frac{m\beta + 2\rho}{2} - 1 - \tau 2^{2\tau} - \left(\frac{1+\beta^2}{2} + \frac{1}{2m}\right) m\tau 2^{\tau} \geq 0, \tag{89}$$

$$\frac{\beta}{2} - \frac{L^2}{\beta} - \frac{L^2}{2} - \frac{1}{2} - \frac{L^2}{4m\beta^2} - \frac{L^2}{4m\beta^2} 2^{\tau-1}\tau > 0. \tag{90}$$

Linear Convergence Rate of Async-ADMM

Theorem III.2 (cont'd)

Then we have

$$L_\beta(\boldsymbol{x}^{K+1},\boldsymbol{z}^{K+1},\boldsymbol{\lambda}^{K+1}) - f^* \leq (1+\frac{1}{\delta\rho})^{-(K+1)}(L_\beta(\boldsymbol{x}^0,\boldsymbol{z}^0,\boldsymbol{\lambda}^0) - f^*), \tag{91}$$

where $\delta \geq \max\{1, \frac{1}{\rho}, \frac{\rho+m\beta}{m\mu} - 1\}$.

We believe that in general the asynchronous ADMM needs more iterations than synchronous ADMM. It is unclear whether the time saved per iteration of the asynchronous ADMM can offset the cost of more iterations in theory, although it shows great advantages in practice.

Outline

I Centralized Distributed Optimization

II Decentralized Distributed Optimization

III Asynchronous Distributed ADMM

IV Nonconvex Distributed ADMM

V ADMM with Generally Linear Constraints

References

Nonconvex Distributed ADMM

Async-ADMM can also be used to solve nonconvex problems. In this case, we only assume that each f_i is L-smooth. THen, $L_\beta(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda})$ is $(\beta - L)$-strongly convergence w.r.t. $\boldsymbol{x}$, and we have the following intermediate result:

$$\begin{aligned} & L_\beta(\boldsymbol{x}^{k+1}, \boldsymbol{z}^k, \boldsymbol{\lambda}^k) - L_\beta(\boldsymbol{x}^k, \boldsymbol{z}^k, \boldsymbol{\lambda}^k) \\ \leq & \sum_{i \in \mathcal{A}^k} \left(\beta \langle \boldsymbol{z}^{\bar{k}_i+1} - \boldsymbol{z}^k, \boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k \rangle - \frac{\beta - L}{2} \|\boldsymbol{x}_i^{k+1} - \boldsymbol{x}_i^k\|^2 \right). \end{aligned} \tag{92}$$

Existence of Convergence

For nonconvex distributed optimization, Async-ADMM also converges.

Theorem IV.1

Suppose that each $f_i(\boldsymbol{x}_i)$ is L-smooth, $i \in [m]$, and Assumption III.1 holds. Let

$$\beta > \frac{1 + L + L^2 + \sqrt{(1 + L + L^2)^2 + 8L^2}}{2},$$
$$\rho > \frac{m(1 + \beta^2)(\tau - 1)^2 - m\beta}{2}. \tag{93}$$

Suppose that $(\boldsymbol{x}_1^k, ..., \boldsymbol{x}_m^k, \boldsymbol{z}^k, \boldsymbol{\lambda}_1^k, ..., \boldsymbol{\lambda}_m^k)$ generated by (70) $\sim$ (72) are bounded, then $(\boldsymbol{x}_1^k, ..., \boldsymbol{x}_m^k, \boldsymbol{z}^k, \boldsymbol{\lambda}_1^k, ..., \boldsymbol{\lambda}_m^k)$ converge to the set of KKT points of (2) in the sense of (85) $\sim$ (87) hold.

Outline

I Centralized Distributed Optimization

II Decentralized Distributed Optimization

III Asynchronous Distributed ADMM

IV Nonconvex Distributed ADMM

V ADMM with Generally Linear Constraints

References

Distributed Optimization with Generally Linear Constraints

In *ADMM Slide: Part 3*, we introduce the following multi-block problem:

$$\min_{\boldsymbol{x}_i} \sum_{i=1}^{m} f_i(\boldsymbol{x}_i), \quad s.t. \quad \sum_{i=1}^{m} \mathbf{A}_i \boldsymbol{x}_i = \boldsymbol{b}. \tag{94}$$

In the following, we presente the distributed version of LADMM-PS in a centralized network. We name it *CLADMM-PS-M* and *CLADMM-PS-W*, for the master and the workers, respectively.

CLADMM-PS-M

The master should do:

1. **for** $k = 0, 1, 2, \ldots$ **do**
   - 1.1 Wait until receiving $\boldsymbol{y}_i^{k+1}$ from all the workers $i \in [m]$
   - 1.2 $\boldsymbol{s}^{k+1} = \sum_{i=1}^{m} \boldsymbol{y}_i^{k+1}$
   - 1.3 $\boldsymbol{\lambda}^{k+1} = \boldsymbol{\lambda}^k + \beta(\boldsymbol{s}^{k+1} - \boldsymbol{b})$
   - 1.4 Send $\boldsymbol{s}^{k+1}$ and $\boldsymbol{\lambda}^{k+1}$ to all the workers

CLADMM-PS-W

Each worker i should do:

1. Initialize $\boldsymbol{x}_i^0$ and $\boldsymbol{\lambda}_i^0$, $i \in [m]$
2. $\boldsymbol{y}_i^0 = \mathbf{A}_i \boldsymbol{x}_i^0$
3. Send $\boldsymbol{y}_i^0$ to the master
4. Wait unitl receiving $\boldsymbol{s}^0$ and $\boldsymbol{\lambda}^0$ from the master
5. for $k = 0, 1, 2, \ldots$ do
   - 5.1 $\boldsymbol{x}_i^{k+1} = \operatorname{argmin}_{\boldsymbol{x}_i} \Big(f_i(\boldsymbol{x}_i) + \langle \boldsymbol{\lambda}^k, \mathbf{A}_i \boldsymbol{x}_i \rangle + \beta \langle \mathbf{A}_i^T (\boldsymbol{s}^k - \boldsymbol{b}), \boldsymbol{x}_i - \boldsymbol{x}_i^k \rangle + \frac{m\beta \|\mathbf{A}_i\|^2}{2} \|\boldsymbol{x}_i - \boldsymbol{x}_i^k\|^2 \Big) =$
     $\operatorname{prox}_{(m\beta\|\mathbf{A}_i\|^2)^{-1} f_i} \Big(\boldsymbol{x}_i^k - \frac{1}{m\beta\|\mathbf{A}_i\|^2} \mathbf{A}_i^T (\boldsymbol{\lambda}^k + \beta(\boldsymbol{s}^k - \boldsymbol{b})) \Big)$
   - 5.2 $\boldsymbol{y}_i^{k+1} = \mathbf{A}_i \boldsymbol{x}_i^{k+1}$
   - 5.3 Send $\boldsymbol{y}_i^{k+1}$ to the master
   - 5.4 Wait unitl receiving $\boldsymbol{s}^{k+1}$ and $\boldsymbol{\lambda}^{k+1}$ from the master

The convergence result of LADMM-PS can be kept.

Outline

I Centralized Distributed Optimization

II Decentralized Distributed Optimization

III Asynchronous Distributed ADMM

IV Nonconvex Distributed ADMM

V ADMM with Generally Linear Constraints

References

References

1. Lin, Zhouchen, Huan Li, and Cong Fang. *Alternating Direction Method of Multipliers for Machine Learning.* Springer Nature, 2022.
2. Boyd, Stephen, Stephen P. Boyd, and Lieven Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.