Review of Douglas-Rachford, ADMM, PDHG

Distributed & decentralized optimization

① $$\min f(x)+g(y) \quad \text{s.t.} \quad Ax+By=C \qquad (P)$$

$$-\min_z \Big(\underbrace{f^*(A^Tz)}_{F(z)} + \underbrace{g^*(B^Tz)+\langle z, c\rangle}_{G(z)} \Big) \qquad (D)$$

(ADMM)
$$\begin{cases} x_{k+1} = \arg\min_x f(x) + \langle z_k, Ax+By_k-c\rangle + \frac{\tau}{2}\|Ax+By_k-c\|^2 \\ y_{k+1} = \arg\min_y g(y) + \langle z_k, Ax_{k+1}+By-c\rangle + \frac{\tau}{2}\|Ax_{k+1}+By-c\|^2 \\ z_{k+1} = z_k + \tau(Ax_{k+1}+By_{k+1}-c) \end{cases}$$

$\Updownarrow$

(DR)
$$\begin{cases} u_{k+1} = \frac{I+R_F^\tau R_G^\tau}{2}(u_k) \\ z_k = \mathrm{Prox}_G^\tau(u_k) \end{cases}$$

$$\min_z F(z)+G(z)$$

$$F(z) = f^*(A^Tz)$$

$$G(z) = g^*(B^Tz)+\langle z, c\rangle$$

② $$\min_x f(Kx)+g(x)$$

PDHG (2010)
$$\begin{cases} x_{k+1} = (I+\eta\partial g)^{-1}[x_k - \eta K^* y_k] \\ y_{k+1} = (I+\tau\partial f^*)^{-1}[y_k + \tau K(2x_{k+1}-x_k)] \end{cases}$$

1) If $K=I$, PDHG $\Leftrightarrow$ ADMM $\Leftrightarrow$ DR on $\min_x f(x)+g(x)$

2) If $K\neq I$, PDHG is NOT ADMM

but PDHG $\Leftrightarrow$ DR $\frac{I+R_F R_G}{2}$ for F & G as follows:

$$\min_{u,v} \underbrace{f(Ku+Cv)}_{F} + \underbrace{g(u) + \iota_{\{v:\, v=0\}}}_{G}$$

$$C = (\gamma^{-2} I - KK^T)^{\frac{1}{2}}, \quad \gamma\|K\| \le 1$$

③ For $\min_x f(x) + g(x)$, (ex: $\min_x \|x\|_1 + \iota_{\{x:\, Ax=b\}}(x)$)

$$\left\{\begin{array}{l} \text{Primal Problem (P)} \\ \min_x f(x) + g(x) \end{array}\right. \iff \left\{\begin{array}{l} \text{Dual Problem (D)} \\ -\min_y f^*(y) + g^*(-y) \end{array}\right.$$

$$\begin{tikzcd}
\boxed{\begin{array}{c}\text{PDHG} \\ \text{with } \tau = \frac{1}{\eta}\end{array}} \arrow[r, Leftrightarrow] & \boxed{\begin{array}{c}\text{DR on (P)} \\ \text{with } \eta\end{array}} \arrow[r, Leftrightarrow] \arrow[d, Leftrightarrow] & \boxed{\begin{array}{c}\text{Simple} \\ \text{ADMM} \\ \text{on (D)} \\ \text{with } \tau=\bar{\tau}=\eta\end{array}} \arrow[r, Leftrightarrow] & \boxed{\begin{array}{c}\text{Split Bregman} \\ \text{on (D)} \\ \tau = \eta\end{array}} \\
 & \boxed{\begin{array}{c}\text{DR on (D)} \\ \text{with } \frac{1}{\eta} > 0\end{array}} \arrow[r, Leftrightarrow] & \boxed{\begin{array}{c}\text{Simple} \\ \text{ADMM} \\ \text{on (P)} \\ \text{with } \tau=\bar{\tau}=\frac{1}{\eta}\end{array}} \arrow[r, Leftrightarrow] & \boxed{\begin{array}{c}\text{Split Bregman} \\ \text{on (P)} \\ \tau = \frac{1}{\eta}\end{array}}
\end{tikzcd}$$

④ For $\min_x f(Kx) + g(x)$ (such as TV minimization)

1) ADMM on (P) $\iff$ DR on (D)

2) PDHG $\iff$ DR on

O'connor & Vandenberghe 2020

$$\boxed{\begin{array}{c} \min_{u,v} \underbrace{f(Ku+Cv)}_{F} + \underbrace{g(u) + \iota_{\{v:\, v=0\}}}_{G} \\ C = (\gamma^{-2} I - KK^T)^{\frac{1}{2}}, \quad \gamma\|K\| \le 1 \end{array}}$$

⑤ For Distributed & decentralized optimization, many popular algorithms are ADMM

$$\min f(x)+g(y) \quad \text{s.t. } Ax+By=c \qquad (P)$$

$$-\min_z \Big(\underbrace{f^*(A^T z)}_{F(z)} + \underbrace{g^*(B^T z) + \langle z, c\rangle}_{G(z)} \Big) \qquad (D)$$

(ADMM) on (P) $\Leftrightarrow$ (DR) on (D)

$$\begin{cases} x_{k+1} = \underset{x}{\operatorname{argmin}}\ f(x) + \langle z_k, Ax+By_k-c\rangle + \frac{\tau}{2}\|Ax+By_k-c\|^2 \\ y_{k+1} = \underset{y}{\operatorname{argmin}}\ g(y) + \langle z_k, Ax_{k+1}+By-c\rangle + \frac{\tau}{2}\|Ax_{k+1}+By-c\|^2 \\ z_{k+1} = z_k + \tau(Ax_{k+1}+By_{k+1}-c) \end{cases}$$

Distributed methods perform computation over a network (a broader class).

Decentralized methods do so without central coordination (a subclass).

Roughly speaking, when communication latency and bandwidth cost much more than computation, decentralized methods are preferred.

Examples: drone fleet control, wireless sensor network, applications of real-time decisions made based on agents' local data

$$\underset{x\in\mathbb{R}^p}{\text{minimize}} \quad \sum_{i=1}^{n} (f_i(x)+h_i(x)), \qquad (1)$$

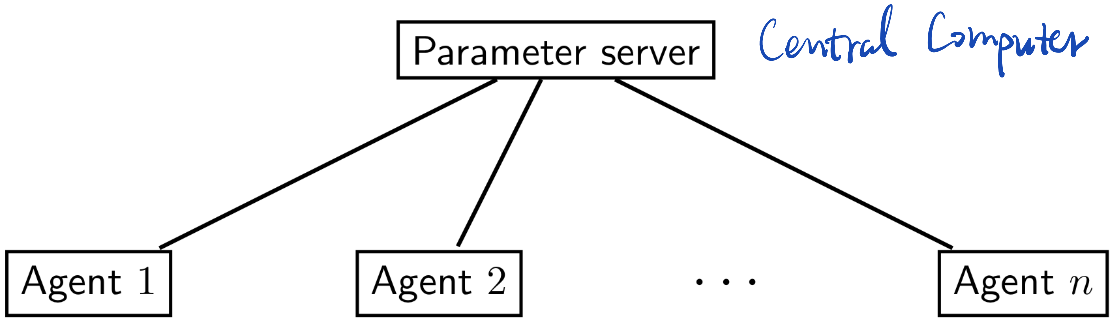
where $f_1,\dots,f_n$ are CCP (and proximable) and $h_1,\dots,h_n$ are CCP and differentiable.

Convex, closed, proper $\hookrightarrow$ Prox can be computed

Case I:

Centralized consensus

Consider a parameter-server network model with a centralized agent coordinating with n individual agents.



Central Computer

CPU/GPU 1 CPU/GPU 2 CPU/GPU n

Distributed gradient method

Consider

$$\underset{x\in\mathbb{R}^p}{\text{minimize}} \quad \frac{1}{n}\sum_{i=1}^{n} h_i(x),$$

where $h_1,\ldots,h_n$ are differentiable. With consensus set $C=\{(x_1,\ldots,x_n)\,|\,x_1=\cdots=x_n\}$, obtain the equivalent problem

$$\begin{array}{ll}\underset{x_1,\ldots,x_n\in\mathbb{R}^p}{\text{minimize}} & \frac{1}{n}\sum_{i=1}^{n} h_i(x_i) \quad f(y)+i_C(y) \\ \text{subject to} & \underbrace{(x_1,\ldots,x_n)}_{y}\in C.\end{array}$$

Example: $h_i(x) = \|NN(x) - y_i\|^2$

x is the parameter of a neural network

y_i is the i-th set of training data

Scheme I

Forward Backward Splitting
$$\begin{cases} x_i^{k+1/2} = x_i^k - \alpha \nabla h_i(x_i^k) \\ x^{k+1} = \dfrac{1}{n}\sum_{i=1}^n x_i^{k+1/2} \end{cases} \iff \begin{cases} \bar{g}^k = \dfrac{1}{n}\sum_{i=1}^n \nabla h_i(x^k) \\ x^{k+1} = x^k - \alpha \bar{g}^k \end{cases}$$

distributed gradient method

This is the distributed gradient method. Assume a solution exists, $h_1, \ldots, h_n$ are L_h-smooth, and $\alpha \in (0, 2/L_h)$. Then $x^k \to x^\star$. (When $h_1, \ldots, h_n$ not differentiable, can use subgradient method of §7.)

This method is (centralized) distributed:

(i) Each agent independently computes $\nabla h_i(x^k)$

(ii) Agents coordinate to compute $\bar{g}^k$ (reduction operation) and the central agent computes and broadcasts x^{k+1} to all individual agents.

Scheme II

Distributed ADMM

Consider
$$\underset{x \in \mathbb{R}^p}{\text{minimize}} \quad \sum_{i=1}^n f_i(x).$$

With the consensus technique, obtain the equivalent problem:
$$\begin{array}{ll} \underset{\substack{x_1,\ldots,x_n \in \mathbb{R}^p \\ y \in \mathbb{R}^p}}{\text{minimize}} & \sum_{i=1}^n f_i(x_i) \\ \text{subject to} & x_i = y \qquad \text{for } i = 1, \ldots, n. \end{array}$$

$G \equiv 0$

Rewrite to fit ADMM's form: $X = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$ $\quad \min F(X) + G(y)$ $\quad \text{s.t. } AX + By = 0$
$$\begin{array}{ll} \underset{\substack{x_1,\ldots,x_n \in \mathbb{R}^p \\ y \in \mathbb{R}^p}}{\text{minimize}} & \underbrace{\sum_{i=1}^n f_i(x_i)}_{F(X)} \\ \text{subject to} & \begin{bmatrix} I & 0 & \cdots & 0 \\ \vdots & \ddots & & \vdots \\ 0 & 0 & \cdots & I \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} -I \\ \vdots \\ -I \end{bmatrix} y = 0. \end{array}$$

(ADMM) $$\begin{cases} x_{k+1} = \underset{x}{\operatorname{argmin}}\ f(x) + \langle z_k, Ax + By_k - c \rangle + \frac{\tau}{2} \|Ax + By_k - c\|^2 \\ y_{k+1} = \underset{y}{\operatorname{argmin}}\ g(y) + \langle z_k, Ax_{k+1} + By - c \rangle + \frac{\tau}{2} \|Ax_{k+1} + By - c\|^2 \\ z_{k+1} = z_k + \tau(Ax_{k+1} + By_{k+1} - c) \end{cases}$$

$$\begin{cases} x^{k+1} = \underset{x}{\operatorname{argmin}}\ F(x) + \langle u^k, Ax + By^k \rangle + \frac{\tau}{2} \|Ax + By^k\|^2 \\ y^{k+1} = \underset{y}{\operatorname{argmin}}\ G(y) + \langle u^k, Ax^{k+1} + By \rangle + \frac{\tau}{2} \|Ax^{k+1} + By\|^2 \\ u^{k+1} = u^k + \tau(Ax^{k+1} + By^{k+1}) \end{cases}$$

$$\begin{cases} x^{k+1} = \underset{x}{\operatorname{argmin}}\ \sum_i f_i(x_i) + \sum_i \langle u_i^k, x_i - y^k \rangle + \frac{\tau}{2} \sum_i \|x_i - y^k\|^2 \\ y^{k+1} = \underset{y}{\operatorname{argmin}}\ \sum_i \langle u_i^k, x_i^{k+1} - y \rangle + \frac{\tau}{2} \sum_i \|x_i^{k+1} - y\|^2 \\ u_i^{k+1} = u_i^k + \tau(x_i^{k+1} - y^{k+1}), \quad i = 1, \cdots, n \end{cases}$$

$$x_i^{k+1} = \underset{x_i \in \mathbb{R}^p}{\operatorname{argmin}} \left\{ f_i(x_i) + \langle u_i^k, x_i - y^k \rangle + \frac{\alpha}{2} \|x_i - y^k\|^2 \right\}$$

$$y^{k+1} = \frac{1}{n} \sum_{i=1}^{n} \left(x_i^{k+1} + \frac{1}{\alpha} u_i^k \right)$$

$$u_i^{k+1} = u_i^k + \alpha(x_i^{k+1} - y^{k+1}).$$

Simplify the iteration by noting that $u_1^k, \ldots, u_n^k$ has mean 0 after the initial iteration and eliminating y^k:

$$x_i^{k+1} = \operatorname{Prox}_{(1/\alpha) f_i} \left(\bar{x}^k - (1/\alpha) u_i^k \right)$$
$$u_i^{k+1} = u_i^k + \alpha(x_i^{k+1} - \bar{x}^{k+1})$$

for $i = 1, \ldots, n$, where $\bar{x}^k = (1/n)(x_1^k + \cdots + x_n^k)$. This is distributed (centralized) ADMM. Convergence follows from convergence of ADMM.

Distributed ADMM

$$x_i^{k+1} = \mathrm{Prox}_{(1/\alpha)f_i}\left(\bar{x}^k - (1/\alpha)u_i^k\right)$$
$$u_i^{k+1} = u_i^k + \alpha(x_i^{k+1} - \bar{x}^{k+1})$$

is distributed:

(i) each agent independently performs the u^k- and x_i^{k+1}-updates with local computation

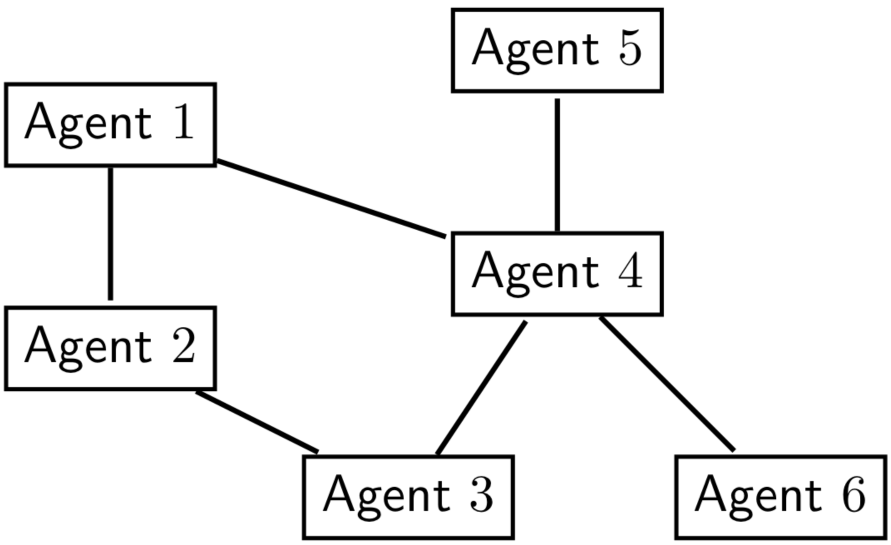
(ii) agents coordinate to compute $\bar{x}^{k+1}$ with a reduction.

Exercise 11.7: Obtain distributed ADMM by applying DRS to the equivalent problem

$$\begin{array}{ll} \underset{x_1,\dots,x_n\in\mathbb{R}^p}{\text{minimize}} & \frac{1}{n}\sum_{i=1}^n h_i(x_i) \\ \text{subject to} & (x_1,\dots,x_n)\in C. \end{array}$$

Case II:

Decentralized optimization with graph consensus



set of edges

If $\{i, j\} \in E$, then we say j is adjacent to i and that j is a neighbor of i (and vice-versa). Write

$$N_i = \{j \in V \mid \{i, j\} \in E\}$$

for the set of neighbors i and $|N_i|$ for the number of neighbors of i.

Using the notation of graphs, we can recast problem (1) into $\sum_i f_i(x) + h_i(x)$

$$\begin{array}{ll} \underset{\{x_i\}_{i \in V} \subset \mathbb{R}^p}{\text{minimize}} & \sum_{i \in V} (f_i(x_i) + h_i(x_i)) \\ \text{subject to} & x_i = x_j \quad \forall \{i, j\} \in E. \end{array} \tag{3}$$

As long as graph is connected, this is equivalent to

$$\min_{x \in \mathbb{R}^p} \sum_{i=1}^{n} [f_i(x) + h_i(x)]$$

Why decentralized optimization?

In a connected network, all agents can communicate with each other. Any optimization method can be executed over the network through relayed communication over multiple edges.

However, in distributed optimization, communication tends to be the bottleneck. So we consider algorithms that communicate across single edges

- without directly relying on long-range relayed communication,
- without creating a bottleneck by communicating with a single central node.

Not delegating any agent as the central agent also improves reliability against agent failure and helps data privacy.

Scheme III

Decentralized ADMM

Consider $h_1 = \cdots = h_n = 0$. For $e = \{i,j\}$, replace the constraint $x_i = x_j$ with $x_i = y_e$ and $x_j = y_e$ to obtain the equivalent problem

$$\begin{array}{ll} \underset{\substack{\{x_i\}_{i\in V}\\ \{y_e\}_{e\in E}}}{\text{minimize}} & \sum_{i\in V} f_i(x_i) \\ \text{subject to} & \begin{cases} x_i - y_e = 0 \\ x_j - y_e = 0 \end{cases} \quad \forall\ e = \{i,j\} \in E. \end{array}$$

$F(x) + G(y)$ $\quad G \equiv 0$

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} \qquad Ax + By = 0$$

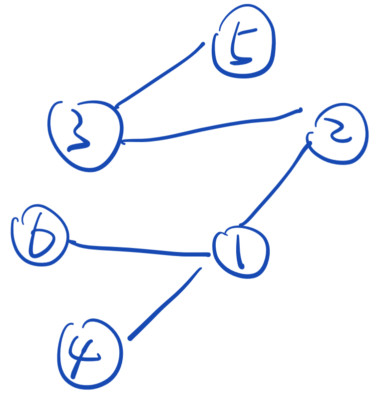
For each $e = \{i,j\} \in E$, introduce the dual variables $u_{e,i}$ for $x_i - y_e = 0$ and $u_{e,j}$ for $x_j - y_e = 0$. The augmented Lagrangian is

$$\begin{aligned} \mathbf{L}_\alpha(x,y,u) = \sum_i f_i(x_i) &+ \sum_{e=\{i,j\}} \left(\langle u_{e,i}, x_i - y_e\rangle + \langle u_{e,j}, x_j - y_e\rangle\right) \\ &+ \sum_{e=\{i,j\}} \frac{\alpha}{2}\left(\|x_i - y_e\|^2 + \|x_j - y_e\|^2\right). \end{aligned}$$

Apply ADMM and obtain

$$\begin{aligned} x_i^{k+1} &= \underset{x_i\in\mathbb{R}^p}{\operatorname{argmin}} \left\{ f_i(x_i) + \sum_{j\in N_i} \left(\langle u^k_{\{i,j\},i}, x_i - y^k_{\{i,j\}}\rangle + \frac{\alpha}{2}\|x_i - y^k_{\{i,j\}}\|^2 \right) \right\} \quad \forall i \in V \\ y_e^{k+1} &= \underset{y_e\in\mathbb{R}^p}{\operatorname{argmin}} \left\{ \sum_{t=i,j} \left(\langle u^k_{e,t}, x_t^{k+1} - y_e\rangle + \frac{\alpha}{2}\|x_t^{k+1} - y_e\|^2 \right) \right\} \quad \forall e = \{i,j\} \in E \\ u_{e,t}^{k+1} &= u^k_{e,t} + \alpha(x_t^{k+1} - y_e^{k+1}) \qquad \forall e = \{i,j\} \in E,\ t = i,j. \end{aligned}$$

which can be simplified to



$$
\begin{aligned}
x_i^{k+1} &= \mathrm{Prox}_{(\alpha|N_i|)^{-1} f_i(x_i)}(v_i^k) \\
a_i^{k+1} &= \frac{1}{|N_i|} \sum_{j \in N_i} x_j^{k+1} \\
v_i^{k+1} &= v_i^k + a_i^{k+1} - \frac{1}{2} a_i^k - \frac{1}{2} x_i^k
\end{aligned}
$$

$N_1 = \{2, 4, 6\}$

is decentralized:

(i) Each agent independently performs the x^{k+1}- and v^{k+1}-updates with local computation.

(ii) Agents send x_i^{k+1} to its neighbors and each agent computes a_i^{k+1} by averaging the x_j^{k+1}'s received from its neighbors (reduction operation in the neighborhood).

The above decentralized methods are synchronous, which can be an unrealistic requirement.

One can use asynchronous decentralized methods, which combine the asynchrony of §6 with the methods of this section.

a good choice of reading project