Distributed and Decentralized Optimization

Ernest K. Ryu

Mathematical and Numerical Optimization
Fall 2021

Distributed and decentralized optimization

In this section, we solve

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

where $r, f_1, \ldots, f_n$ are CCP (and proximable) and $h_1, \ldots, h_n$ are CCP and differentiable, in a computational setup where agents $i = 1, \ldots, n$ each perform local computation with f_i and h_i and communicate over a network to find the (shared) solution $x^\star$.

We distinguish *distributed* and *decentralized* methods:
distributed methods compute over a network (broader class) while decentralized methods do so without central coordination (special case).

Distributed and decentralized applications

One application of distributed optimization is solving extremely large optimization problems that require the computing power of a cluster of computers communicating over a network.

Another application is controlling a fleet of autonomous vehicles (such as drones) or a wireless sensor network, where individual agents make real-time decisions based on data gathered by itself and other agents.

Decentralized methods are effective for these setups as they minimize the high cost and latency of communication.

Outline

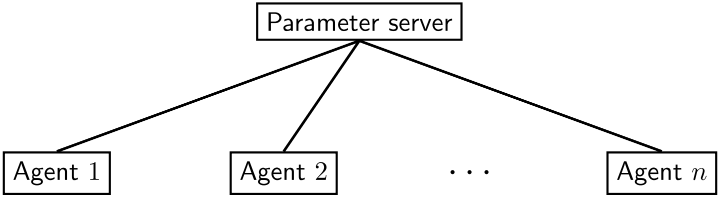
Distributed optimization with centralized consensus

Decentralized optimization with graph consensus

Decentralized optimization with mixing matrices

Centralized consensus

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



We study distributed methods based on the consensus technique.

Throughout this section, write $C = \{(x_1, \ldots, x_n) \,|\, x_1 = \cdots = x_n \in \mathbb{R}^p\}$ for the consensus set, an unbound index i is assumed to range from $i = 1, \ldots, n$, and we write the mean over $i = 1, \ldots, n$ with a bar notation as in $\bar{x}^k = (1/n)(x_1^k + \cdots + x_n^k)$ and $\bar{g}^k = (1/n)(g_1^k + \cdots + g_n^k)$.

Distributed proximal gradient method

Consider

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

where $h_1, \dots, h_n$ are differentiable. With consensus technique, obtain equivalent problem

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

FBS is:

$$\begin{aligned} x_i^{k+1/2} &= x^k - \alpha \nabla h_i(x^k) \\ x^{k+1} &= \mathrm{Prox}_{\alpha r}\left(\frac{1}{n}\sum_{i=1}^{n} x_i^{k+1/2}\right). \end{aligned}$$

Distributed proximal gradient method

Equivalent to:

$$
\begin{aligned}
g_i^k &= \nabla h_i(x^k) \\
x^{k+1} &= \mathrm{Prox}_{\alpha r}\left(x^k - \alpha \bar{g}^k\right).
\end{aligned}
$$

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

Distributed ADMM

Consider

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

With consensus technique, obtain equivalent problem:

$$\begin{aligned} \underset{\substack{x_1,\ldots,x_n \in \mathbb{R}^p \\ y \in \mathbb{R}^p}}{\text{minimize}} \quad & \sum_{i=1}^{n} f_i(x_i) \\ \text{subject to} \quad & x_i = y. \end{aligned}$$

Rewrite to fit ADMM's form:

$$\begin{aligned} \underset{\substack{x_1,\ldots,x_n \in \mathbb{R}^p \\ y \in \mathbb{R}^p}}{\text{minimize}} \quad & \sum_{i=1}^{n} f_i(x_i) \\ \text{subject to} \quad & \begin{bmatrix} I & 0 & \cdots & 0 \\ \vdots & \ddots & & \vdots \\ 0 & 0 & \cdots & I \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} + \begin{bmatrix} -I \\ \vdots \\ -I \end{bmatrix} y = 0. \end{aligned}$$

Distributed ADMM

Apply ADMM:

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

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

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

This is *distributed (centralized) ADMM*. Convergence follows from convergence of ADMM.

Distributed ADMM

Distributed ADMM

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

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.

Primal decomposition technique

The primal decomposition technique obtains a *master problem* through minimizing away local variables. This is a special case of the infimal postcomposition technique.

Consider the problem

$$\underset{\substack{x_i \in \mathbb{R}^{p_i} \\ y \in \mathbb{R}^q}}{\text{minimize}} \quad r(y) + \frac{1}{n} \sum_{i=1}^{n} f_i(x_i, y).$$

For a fixed $y \in \mathbb{R}^q$, the minimization over $x_1, \dots, x_n$ decomposes into n embarrassingly parallel tasks. We call $x_1, \dots, x_n$ *local variables* and y the *coupling variable*.

Primal decomposition technique

With $\phi_i(y) = \inf_{x_i \in \mathbb{R}^{p_i}} f_i(x_i, y)$, get equivalent master problem

$$\underset{y \in \mathbb{R}^q}{\text{minimize}} \quad r(y) + \frac{1}{n} \sum_{i=1}^{n} \phi_i(y).$$

If $\phi_1, \dots, \phi_n$ are smooth, use proximal gradient method:

$$y^{k+1} = \mathrm{Prox}_{\alpha r}\left(y^k - \alpha \frac{1}{n} \sum_{i=1}^{n} \nabla \phi_i(y^k) \right).$$

Using Exercise 11.2, we express the method as

$$\begin{aligned} x_i^\star(y^k) &\in \underset{x_i \in \mathbb{R}^{p_i}}{\mathrm{argmin}}\, f_i(x_i, y^k) \\ (0, g_i^k) &\in \partial f_i(x_i^\star(y^k), y^k) \\ y^{k+1} &= \mathrm{Prox}_{\alpha r}\left(y^k - \alpha \bar{g}^k \right), \end{aligned}$$

provided that the argmins exists.

When r is proximable but $\phi_1, \dots, \phi_n$ are not smooth, we can apply the proximal subgradient method of §7.

Example: Common bound problem

Agents $i = 1, \ldots, n$ each reduce its cost $f_i(x_i)$ subject to the constraint $g_i(x_i) \preceq y$ while paying a common cost $r(y)$:

$$\begin{array}{ll} \underset{\substack{x_i \in \mathbb{R}^{p_i} \\ y \in \mathbb{R}^q}}{\text{minimize}} & r(y) + \sum_{i=1}^{n} f_i(x_i) \\ \text{subject to} & g_i(x_i) \preceq y. \end{array}$$

This problem is equivalent to the master problem

$$\underset{y \in \mathbb{R}^q}{\text{minimize}} \quad r(y) + \frac{1}{n} \sum_{i=1}^{n} \phi_i(y),$$

where

$$\phi_i(y) = \inf_{x_i \in \mathbb{R}^{p_i}} \left\{ n f_i(x_i) + \delta_{\{(x_i, y) \,|\, g_i(x_i) \preceq y\}}(x_i, y) \right\}.$$

See Exercise 11.6 for evaluating the subdifferential $\partial \phi_i(y)$.

Example: Resource sharing problem

Agents $i = 1, \ldots, n$ each reduces its cost $f_i(x_i)$ subject to a total resource constraint $\sum_{i=1}^n g_i(x_i) \preceq y$ while paying a common cost $r(y)$:

$$\begin{array}{ll} \underset{\substack{x_i \in \mathbb{R}^{p_i} \\ y \in \mathbb{R}^q}}{\text{minimize}} & r(y) + \sum_{i=1}^n f_i(x_i) \\ \text{subject to} & \sum_{i=1}^n g_i(x_i) \preceq y. \end{array}$$

This problem is equivalent to the master problem

$$\underset{y_1, \ldots, y_n \in \mathbb{R}^q}{\text{minimize}} \quad r(y_1 + \cdots + y_n) + \frac{1}{n} \sum_{i=1}^n \phi_i(y_i),$$

where $\phi_i(y_i) = \inf_{x_i \in \mathbb{R}^{p_i}} \{ n f_i(x_i) + \delta_{\{(x_i, y_i) \,|\, g_i(x_i) \preceq y_i\}}(x_i, y_i) \}$. See Exercise 11.6 for evaluating $\partial \phi_i(y_i)$.

Dual decomposition technique

Dual decomposition technique obtains a master problem by taking the dual. This is the dualization technique, but the focus is on obtaining a sum structure so that we can apply the base distributed methods.

Consider

$$\underset{\substack{x_i\in\mathbb{R}^{p_i}\\ y\in\mathbb{R}^q}}{\text{minimize}} \quad \sum_{i=1}^n f_i(x_i,y).$$

The equivalent primal problem

$$\begin{array}{ll} \underset{\substack{x_i\in\mathbb{R}^{p_i},\, z_i=\in\mathbb{R}^q\\ y\in\mathbb{R}^q}}{\text{minimize}} & \displaystyle\sum_{i=1}^n f_i(x_i,z_i) \\ \text{subject to} & z_i = y \end{array}$$

is generated by the Lagrangian

$$\mathbf{L}(x_1,\dots,x_n,y,z_1,\dots,z_n,v_1,\dots,v_n)=\sum_{i=1}^n \left(f_i(x_i,z_i)-\langle v_i,z_i-y\rangle\right).$$

Dual decomposition technique

With

$$\inf_{y\in\mathbb{R}^q}\sum_{i=1}^{n}\langle v_i, y\rangle = \begin{cases} 0, & \text{if } v_1+\cdots+v_n=0 \\ -\infty, & \text{otherwise.}\end{cases}$$

and

$$\psi_i(v_i) = \sup_{\substack{x_i\in\mathbb{R}^p \\ z_i\in\mathbb{R}^q}} \left\{-f_i(x_i,z_i)+\langle v_i, z_i\rangle\right\},$$

we obtain the master dual problem

$$\underset{v_1,\ldots,v_n\in\mathbb{R}^q}{\text{maximize}} \quad -\delta_{C^\perp}(v_1,\ldots,v_n)-\sum_{i=1}^{n}\psi_i(v_i),$$

where $C^\perp = \{(v_1,\ldots,v_n)\,|\,v_1+\cdots+v_n=0\}$. (Cf. Exercise 11.8.)

Dual decomposition technique

If $\psi_1, \ldots, \psi_n$ are smooth, apply the projected gradient method:

$$g_i^k = \nabla \psi_i(v_i^k)$$
$$v_i^{k+1} = v_i^k - \alpha(g_i^k - \bar{g}^k)$$

with initialization $(v_1^0, \ldots, v_n^0) \in C^\perp$. Using Exercise 11.3 we have

$$(x_i^\star(v_i^k), g_i^k) \in \underset{\substack{x_i \in \mathbb{R}^p \\ g_i \in \mathbb{R}^q}}{\operatorname{argmin}} \left\{ -f_i(x_i, g_i) + \langle v_i^k, g_i \rangle \right\}$$
$$v_i^{k+1} = v_i^k - \alpha(g_i^k - \bar{g}^k),$$

provided that the argmins exist.

When $\phi_1, \ldots, \phi_n$ are not smooth, we can apply the projected subgradient method of §7 or distributed ADMM/DRS (cf. Exercise 11.9, 11.10).

Dual decomposition with inequality constraints

Consider the resource sharing problem:

$$\begin{array}{ll} \underset{\substack{x_i\in\mathbb{R}^{p_i}\\ y\in\mathbb{R}^q}}{\text{minimize}} & r(y)+\dfrac{1}{n}\displaystyle\sum_{i=1}^n f_i(x_i) \\ \text{subject to} & \displaystyle\sum_{i=1}^n g_i(x_i)\preceq y. \end{array}$$

This primal problem is generated by the Lagrangian

$$\mathbf{L}(x_1,\dots,x_n,y,u)=r(y)-\langle u,y\rangle+\frac{1}{n}\sum_{i=1}^n\left(f_i(x_i)+\langle u,g_i(x_i)\rangle\right)-\delta_{\mathbb{R}^n_+}(u),$$

where $\mathbb{R}^q_+$ denotes the nonnegative orthant. With

$$\psi_i(u)=\begin{cases}\sup_{x_i\in\mathbb{R}^{p_i}}\left(\langle -u,g_i(x_i)\rangle-f_i(x_i)\right) & \text{if } u\succeq 0\\ \infty & \text{otherwise,}\end{cases}$$

we obtain the master dual problem

$$\underset{u\in\mathbb{R}^q}{\text{maximize}}\quad -r^*(u)-\delta_{\mathbb{R}^q_+}(u)-\frac{1}{n}\sum_{i=1}^n\psi_i(u).$$

Dual decomposition with inequality constraints

If r^* is proximable and $\psi_1, \dots \psi_n$ are smooth, we can apply DYS and Exercise 11.3 to get

$$\begin{aligned} u^{k+1/2} &= \Pi_{\mathbb{R}^q_+}\left(\zeta^k\right) \\ x_i^{k+1} &\in \underset{x_i \in \mathbb{R}^{p_i}}{\operatorname{argmin}} \left\{ f_i(x_i) + \langle u^{k+1/2}, g_i(x_i) \rangle \right\} \\ u^{k+1} &= \operatorname{Prox}_{\alpha r^*}\left(2u^{k+1/2} - \zeta^k + \frac{\alpha}{n} \sum_{i=1}^n g_i(x_i^{k+1}) \right) \\ \zeta^{k+1} &= \zeta^k + u^{k+1} - u^{k+1/2}. \end{aligned}$$

Dual decomposition with inequality constraints

If $r = \delta_{\{b\}}$ and $\psi_1, \ldots \psi_n$ are smooth, then $r^*(u) = \langle u, b\rangle$ and we can apply the proximal gradient method and Exercise 11.3 to get

$$
\begin{aligned}
x_i^{k+1} &\in \operatorname*{argmin}_{x_i \in \mathbb{R}^{p_i}} \left\{ f_i(x_i) + \langle u^k, g_i(x_i)\rangle \right\} \\
u^{k+1} &= \Pi_{\mathbb{R}^q_+} \left(u^k + \frac{\alpha}{n} \sum_{i=1}^{n} \left(g_i(x_i^{k+1}) - b \right) \right).
\end{aligned}
$$

When $r = \delta_{\{b\}}$ but $\psi_1, \ldots \psi_n$ are not smooth, we can apply the projected subgradient method of §7.

Outline

Distributed optimization with centralized consensus

Decentralized optimization with graph consensus

Decentralized optimization with mixing matrices

Note on the word “graph”

“Graph” has two distinct meanings in mathematics.

The first meaning, as in “we plot the graph $\sin(x)$ on a graphing calculator”, concerns the relationship between the inputs and outputs of a function. The *graph* of an operator, which we denote as $\operatorname{Gra} \mathbb{A}$, and the scaled relative *graph* uses this first meaning.

Here, we consider the second meaning, the use in discrete mathematics for representing networks.

Networks and graphs

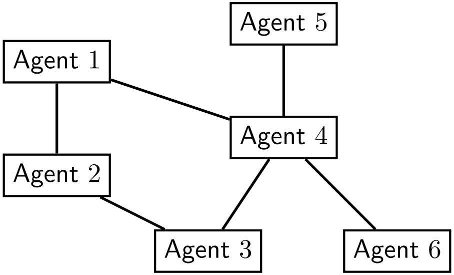
A graph $G = (V, E)$ represents a network. V is set of nodes and E is set of edges. Assume

- Network is finite and with nodes 1 through n, i.e., $V = \{1, \ldots, n\}$.
- Graph is undirected, i.e., an edge $\{i, j\} \in E$ is an unordered pair of distinct nodes i and j.
- Graph has no self-loops, i.e., $\{i, i\} \notin E$ for all $i \in V$.
- Graph is connected, i.e., for any $i, j \in V$ such that $i \neq j$, there is a sequence of edges

$$\{i, v_1\}, \{v_1, v_2\}, \ldots, \{v_{k-1}, v_k\}, \{v_k, j\} \in E.$$

Networks and graphs

With graphs, we can represent networks without a central coordinating agent. The following graph has $V = \{1, 2, 3, 4, 5, 6\}$ and $E = \{\{1, 2\}, \{1, 4\}, \{2, 3\}, \{3, 4\}, \{4, 5\}, \{4, 6\}\}$.



A node represents a computational agent that stores data and performs computation, and an edge $\{i, j\}$ represents a direct connection between i and j through which agents i and j can communicate.

Networks and graphs

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.

In the decentralized setup, assume $r = 0$. Using the notation of graphs, we can recast problem (1) into

$$\begin{array}{ll} \underset{\{x_i\}_{i \in V} \subset \mathbb{R}^p}{\text{minimize}} & \sum_{i \in V} f_i(x_i) + h_i(x_i) \\ \text{subject to} & x_i = x_j \quad \forall \, \{i, j\} \in E. \end{array} \tag{2}$$

Why decentralized optimization?

Because the network is connected, all agents can communicate with each other. (Any computer can communicate with any other computer over the internet.) 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.

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

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

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

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

Decentralized ADMM

Express ADMM as

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

We simplify further.

Decentralized ADMM

Substitute $y_e^{k+1} = \frac{1}{2}\sum_{t=i,j}(x_t^{k+1} + \frac{1}{\alpha}u_{e,t}^k)$:

$$\begin{aligned} u_{e,i}^{k+1} &= u_{e,i}^k + \alpha\left(x_i^{k+1} - \frac{1}{2}\sum_{t=i,j}\left(x_t^{k+1} + \frac{1}{\alpha}u_{e,t}^k\right)\right) \\ &= \frac{1}{2}(u_{e,i}^k - u_{e,j}^k) + \frac{\alpha}{2}(x_i^{k+1} - x_j^{k+1}), \quad \forall e = \{i,j\} \in E. \end{aligned}$$

Using $u_{e,i}^k + u_{e,j}^k = 0$ for all $e = \{i,j\}$ and $k = 1, 2, \ldots,$ write $y_e^k = \frac{1}{2}(x_i^k + x_j^k)$, $u_{e,i}^{k+1} = u_{e,i}^k + \frac{\alpha}{2}(x_i^{k+1} - x_j^{k+1})$, and

$$\begin{aligned} x_i^{k+1} &= \operatorname*{argmin}_{x_i \in \mathbb{R}^p}\left\{f_i(x_i) + \frac{\alpha}{2}\sum_{j \in N_i}\left\|x_i - \frac{1}{2}(x_i^k + x_j^k) + \frac{1}{\alpha}u_{\{i,j\},i}^k\right\|^2\right\} \\ &= \operatorname*{argmin}_{x_i \in \mathbb{R}^p}\left\{f_i(x_i) + \frac{\alpha|N_i|}{2}\left\|x_i - \frac{1}{|N_i|}\sum_{j \in N_i}\left(\frac{1}{2}(x_i^k + x_j^k) - \frac{1}{\alpha}u_{\{i,j\},i}^k\right)\right\|^2\right\} \end{aligned}$$

for all $i \in V$.

Decentralized ADMM

Defining $v_i^k = \frac{1}{|N_i|} \sum_{j \in N_i} \left(\frac{1}{2}(x_i^k + x_j^k) - \frac{1}{\alpha} u_{\{i,j\},i}^k \right)$ and $a_i^k = \frac{1}{|N_i|} \sum_{j \in N_i} x_j^k$ and obtain: for every $i \in V$

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

for $i \in V$. This is *decentralized ADMM*. Convergence follows from convergence of ADMM.

Decentralized ADMM

Decentralized ADMM

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

is decentralized:

(i) Each agent independently performs the v^k- and x^{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).

Synchronization

The decentralized methods we study 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.

Outline

Distributed optimization with centralized consensus

Decentralized optimization with graph consensus

Decentralized optimization with mixing matrices

Decentralized notation

Define stack operator and use boldface to denote stacked variables:

$$\mathbf{x} = \mathrm{stack}(x_1, \dots, x_n) = \begin{bmatrix} — & x_1^\intercal & — \\ & \vdots & \\ — & x_n^\intercal & — \end{bmatrix} \in \mathbb{R}^{n \times p}.$$

Write $x^\star \in \mathbb{R}^p$ and $\mathbf{x}^\star = \mathrm{stack}(x^\star, \dots, x^\star) \in \mathbb{R}^{n \times p}$ for the solution.
For $\mathbf{x} = \mathrm{stack}(x_1, \dots, x_n)$ and $\mathbf{y} = \mathrm{stack}(y_1, \dots, y_n)$, define

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{n} \langle x_i, y_i \rangle.$$

For any symmetric positive semidefinite $A \in \mathbb{R}^{n \times n}$, define
$\|\mathbf{x}\|_A^2 = \langle \mathbf{x}, A\mathbf{x} \rangle$. Specifically, $\|\mathbf{x}\|^2 = \|\mathbf{x}\|_I^2 = \langle \mathbf{x}, \mathbf{x} \rangle$.

Decentralized notation

Define

$$f(\mathbf{x}) = \sum_{i=1}^{n} f_i(x_i), \qquad h(\mathbf{x}) = \sum_{i=1}^{n} h_i(x_i)$$

$$\mathrm{Prox}_{\alpha f}(\mathbf{x}) = \mathrm{stack}(\mathrm{Prox}_{\alpha f_1}(x_1), \dots, \mathrm{Prox}_{\alpha f_n}(x_n))$$

$$\nabla h(\mathbf{x}) = \mathrm{stack}(\nabla h_1(x_1), \dots, \nabla h_n(x_n)).$$

We say $\mathbf{x} = \mathrm{stack}(x_1, \dots, x_n)$ is *in consensus* if $x_1 = \cdots = x_n$.
A solution (or any feasible point) of (2) is in consensus. The methods of this section produce iterates that are in consensus in the limit.

Mixing matrices

Informally, $W \in \mathbb{R}^{n\times n}$ is a *mixing matrix* when an application of W represents a round of communication and the aggregation of the communicated information. Write $\lambda_1, \ldots, \lambda_n$ for the eigenvalues of W.

W is a *decentralized mixing matrix* with respect to $G = (V, E)$ if $W_{ij} = 0$ when $i \neq j$ and $\{i, j\} \notin E$. (W_{ii} may be nonzero.)

Wy can be evaluated in a decentralized manner if W is decentralized:

$$(Wy)_i = \sum_{j=1}^{n} W_{ij} y_j = \sum_{j \in N_i \cup \{i\}} W_{ij} y_j .$$

Example: Local averaging matrix

With mixing matrix

$$W_{ij} = \begin{cases} \frac{1}{|Ni|} & \text{if } \{i,j\} \in E \\ 0 & \text{otherwise} \end{cases}$$

for $i, j \in \{1, \dots, n\}$ and

$$\tilde{f}(\mathbf{x}) = \sum_{i=1}^{n} \frac{1}{|N_i|} f_i(x_i),$$

we can express decentralized ADMM as

$$\begin{aligned} \mathbf{x}^{k+1} &= \mathrm{Prox}_{\alpha \tilde{f}}(\mathbf{v}^k) \\ \mathbf{a}^{k+1} &= W\mathbf{x}^{k+1} \\ \mathbf{v}^{k+1} &= \mathbf{v}^k + \mathbf{a}^{k+1} - \frac{1}{2}\mathbf{a}^k - \frac{1}{2}\mathbf{x}^k. \end{aligned}$$

Decentralized averaging

Agent $i \in V$ has a vector $x_i \in \mathbb{R}^p$ and goal is to compute the average $\bar{x} = \frac{1}{n}\sum_{i=1}^n x_i$ in a decentralized manner.
(This is a special case of (1) with $f_i(x) = \frac{1}{2}\|x - x_i\|^2$.)

Decentralized averaging method:

$$\mathbf{x}^{k+1} = W\mathbf{x}^k$$

with the starting point $\mathbf{x}^0 = \mathrm{stack}(x_1, \dots, x_n)$ and a decentralized mixing matrix $W \in \mathbb{R}^{n\times n}$. Converges for all $\mathbf{x}^0$ if and only if $W\mathbf{1} = \mathbf{1}$, $\mathbf{1}^\intercal W = \mathbf{1}^\intercal$, and $1 = |\lambda_1| > |\lambda_2| \ge \cdots \ge |\lambda_n|$. (Cf. Exercise 11.14.)

Condition $W\mathbf{1} = \mathbf{1}$ implies $\mathbf{x}$-vectors in consensus are fixed points.
Condition $\mathbf{1}^\intercal W = \mathbf{1}^\intercal$ implies mean is preserved throughout the iteration.
Eigenvalue condition implies the iteration converges.

Assumptions on mixing matrices

A mixing matrix $W \in \mathbb{R}^{n\times n}$ used in decentralized optimization often satisfies some or all of the following assumptions:

$$W = W^\intercal \tag{3a}$$

$$\mathcal{N}(I - W) = \operatorname{span}(\mathbf{1}) \tag{3b}$$

$$1 = |\lambda_1| > \max\{|\lambda_2|, \ldots, |\lambda_n|\}. \tag{3c}$$

(3a) was not assumed in decentralized ADMM or averaging, but it is common; methods with symmetric W tend to be easier to analyze.
(3b) implies $\mathbf{x}$ is in consensus if and only if $\mathbf{x} = W\mathbf{x}$ and is required for almost all decentralized optimization methods.
(3c) is assumed to establish the convergence of certain methods. Note that (3a) implies the eigenvalues are real (but not necessarily nonnegative) and assumption (3b) implies $1 = \lambda_1$.

Example: Laplacian-based mixing matrix

Consider the symmetric mixing matrix

$$W = I - \frac{1}{\tau} L \in \mathbb{R}^{n \times n},$$

where L is the *graph Laplacian*

$$L_{ij} = \begin{cases} |N_i| & \text{if } i = j \\ -1 & \text{if } \{i, j\} \in E \\ 0 & \text{otherwise} \end{cases}$$

for $i, j \in \{1, \ldots, n\}$ and τ is a constant satisfying $\tau > \frac{1}{2}\lambda_{\max}(L)$. Using standard arguments with the graph Laplacian, one can show that $W\mathbf{1} = \mathbf{1}$ and $1 = \lambda_1 > \max\{|\lambda_2|, \ldots, |\lambda_n|\}$.

Example: Metropolis mixing matrix

Consider the symmetric mixing matrix $W \in \mathbb{R}^{n \times n}$ defined by

$$W_{ij} = \begin{cases} \frac{1}{\max\{|N_i|,|N_j|\}+\varepsilon} & \text{if } \{i,j\} \in E \\ 1 - \sum_{j \in N_i} W_{ij} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

for $i, j \in \{1, \dots, n\}$, where $\varepsilon > 0$. Using standard arguments with the Perron–Frobenius theory (W is a stochastic matrix for an irreducible and aperiodic Markov chain), one can show $W\mathbf{1} = \mathbf{1}$ and $1 = \lambda_1 > \max\{|\lambda_2|, \dots, |\lambda_n|\}$.

Relationship with stochastic matrices

$P \in \mathbb{R}^{n\times n}$ satisfying $P_{ij} \geq 0 \; \forall\, i, j$ and $P\mathbf{1} = \mathbf{1}$ is a stochastic matrix.

Mixing matrices and stochastic matrices share some apparent similarities, but they do have some key differences.

One difference is that mixing matrices can have negative entries. (Cf. Exercise 11.16.)

Another difference is in their primary use as linear operators. With a stochastic matrix P satisfying $P\mathbf{1} = \mathbf{1}$ (total probability mass of 1 is preserved) the key operation is the vector-matrix product

$$(\pi^{k+1})^\intercal = (\pi^k)^\intercal P.$$

With mixing matrix W satisfying $W\mathbf{1} = \mathbf{1}$ (vector in consensus remains in consensus) the key operation is the matrix-(stacked vector) product

$$\mathbf{x}^{k+1} = W\mathbf{x}^k.$$

Relationship with stochastic matrices

When a mixing matrix is a stochastic matrix, one can utilize the classical Markov chain theory based on the Perron–Frobenius theorem. For example, if $W \in \mathbb{R}^{n \times n}$ is a stochastic matrix for an irreducible Markov chain, then $\mathcal{N}(I - W) = \operatorname{span}(\mathbf{1})$ holds; if the Markov chain is irreducible and aperiodic, then $1 = \lambda_1 > \max\{|\lambda_2|, \ldots, |\lambda_n|\}$ holds.

Dynamic mixing matrix

For simplicity, we assumed mixing matrices do not depend on the iteration.

However, when the connectivity of the underlying graph is dynamic, one has to use a series of dynamic mixing matrices instead of a fixed one.

Inexact decentralized methods

Consider

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

and a symmetric mixing matrix $W \in \mathbb{R}^{n \times n}$ satisfying $\mathcal{N}(I - W) = \mathrm{span}(\mathbf{1})$ and $1 = \lambda_1 > \max\{\lambda_2, \ldots, \lambda_n\}$. We can equivalently write the problem as

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^{n \times p}}{\text{minimize}} & h(\mathbf{x}) \\ \text{subject to} & (I - W)\mathbf{x} = 0. \end{array} \tag{4}$$

We now consider inexact decentralized methods that solve a penalty formulations that approximate (4). When these inexact methods converge, they converge to an approximation of the original solution.

Decentralized gradient descent (DGD)

Consider the penalty formulation

$$\underset{\mathbf{x}\in\mathbb{R}^{n\times p}}{\text{minimize}} \quad h(\mathbf{x}) + \frac{1}{2\alpha}\|\mathbf{x}\|_{I-W}^2.$$

We expect this formulation to approximate (4) well when $\alpha > 0$ is small.

Gradient descent with stepsize α applied to this penalty formulation is

$$\begin{aligned}\mathbf{x}^{k+1} &= \mathbf{x}^k - \alpha\left(\nabla h(\mathbf{x}^k) + \frac{1}{\alpha}(I-W)\mathbf{x}^k\right)\\ &= W\mathbf{x}^k - \alpha\nabla h(\mathbf{x}^k).\end{aligned}$$

We call this *decentralized gradient descent (DGD)* or the *combine-then-adapt* method. DGD is decentralized when W is a decentralized mixing matrix. Assume the penalty formulation has a solution, $h_1, \ldots, h_n$ are L-smooth, and $\alpha \in (0, (1+\lambda_n(W))/L)$. Then $\mathbf{x}^k$ converges to a solution of the penalty formulation.

Diffusion

Further assume $W \succ 0$ and consider the penalty formulation

$$\underset{\mathbf{x}\in\mathbb{R}^{n\times p}}{\text{minimize}} \quad h(\mathbf{x}) + \frac{1}{2\alpha}\|\mathbf{x}\|^2_{W^{-1}-I}.$$

FBS with $(\mathbb{I}+\alpha\mathbb{B})^{-1}(\mathbb{I}-\alpha\mathbb{A})$, where $\mathbb{A}=\nabla h$, $\mathbb{B}=\frac{1}{\alpha}(W^{-1}-I)$, is

$$\mathbf{x}^{k+1} = W(\mathbf{x}^k - \alpha\nabla h(\mathbf{x}^k)).$$

Note that W^{-1} appears in the analysis and formulation of the algorithm, but not within the iteration $\mathbf{x}^{k+1} = W(\mathbf{x}^k - \alpha\nabla h(\mathbf{x}^k))$.

We call this *diffusion* or the *adapt-then-combine* method. Diffusion is decentralized when W is a decentralized mixing matrix. Assume the penalty formulation has a solution, $h_1,\dots,h_n$ are L_h-smooth, and $\alpha\in(0,2/L_h)$. Then $\mathbf{x}^k$ converges to a solution of the penalty formulation.

Diffusion

The stepsize condition for diffusion $\alpha < 2/L$ is wider than the stepsize condition for DGD $\alpha < (1 + \lambda_n(W))/L$. Loosely speaking, use of a larger stepsize often leads to faster convergence.

When $\min\{\lambda_2, \ldots, \lambda_n\} > 0$ does not hold, we can still use diffusion using the positive definite mixing matrix $(1 - \theta)I + \theta W$ with $\theta \in (0, 1/(1 - \min\{\lambda_2, \ldots, \lambda_n\}))$.

Exact decentralized methods

Consider a symmetric mixing matrix $W \in \mathbb{R}^{n \times n}$ satisfying $\mathcal{N}(I - W) = \mathrm{span}(\mathbf{1})$ and $1 = \lambda_1 > \max\{\lambda_2, \dots, \lambda_n\}$. Since $I - W$ is symmetric positive semidefinite, there exists a $U \in \mathbb{R}^{n \times n}$ such that

$$U^2 = \frac{1}{2}(I - W).$$

Note, $\mathcal{N}(U) = \mathrm{span}(\mathbf{1})$.

Consider

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

which is equivalent to

$$\underset{\mathbf{x} \in \mathbb{R}^{n \times p}}{\text{minimize}} \quad f(\mathbf{x}) + h(\mathbf{x}) + \delta_{\{0\}}(U\mathbf{x}).$$

We now present methods that converge to an exact solution. The algorithms utilize W, while U is used only in the analysis.

PG-EXTRA

Apply Condat–Vũ of Exercise 3.5 with $g = \delta_{\{0\}}$ (so $\mathrm{Prox}_{\beta g^*} = \mathbb{I}$) to get

$$\begin{aligned}\mathbf{u}^{k+1} &= \mathbf{u}^k + \beta U \mathbf{x}^k \\ \mathbf{x}^{k+1} &= \mathrm{Prox}_{\alpha f}\left(\mathbf{x}^k - \alpha \nabla h(\mathbf{x}^k) - \alpha U(2\mathbf{u}^{k+1} - \mathbf{u}^k)\right).\end{aligned}$$

Initialize $\mathbf{u}^0 = 0$ but set $\mathbf{x}^0$ arbitrarily. To eliminate U, define $\mathbf{w}^k = \frac{1}{\beta} U \mathbf{u}^k = \frac{1}{2}(I - W)\sum_{j=0}^{k-1} \mathbf{x}^k$. Choose $\beta = \alpha^{-1}$ for simplicity and rearrange the terms to get

$$\begin{aligned}\mathbf{x}^{k+1} &= \mathrm{Prox}_{\alpha f}(W\mathbf{x}^k - \alpha \nabla h(\mathbf{x}^k) - \mathbf{w}^k) \\ \mathbf{w}^{k+1} &= \mathbf{w}^k + \frac{1}{2}(I - W)\mathbf{x}^k,\end{aligned}$$

where we initialize $\mathbf{w}^0 = 0$ and set $\mathbf{x}^0$ arbitrarily.

PG-EXTRA

This method is called *PG-EXTRA*

$$\begin{aligned} \mathbf{x}^{k+1} &= \mathrm{Prox}_{\alpha f}(W\mathbf{x}^k - \alpha\nabla h(\mathbf{x}^k) - \mathbf{w}^k) \\ \mathbf{w}^{k+1} &= \mathbf{w}^k + \frac{1}{2}(I - W)\mathbf{x}^k. \end{aligned}$$

Assume total duality holds, $h_1, \ldots, h_n$ are L-smooth, and $0 < \alpha < (1 + \lambda_{\min}(W))/L$. Then, $\mathbf{x}^k \to \mathbf{x}^\star$. The stepsize bound follows from the stepsize of Condat–Vũ and $\lambda_{\max}(U^2) = 1/2 - 1/2\lambda_{\min}(W)$. The method *EXTRA* is the special case of EXTRA with $f = 0$.

NIDS

Apply PD3O to

$$\underset{\mathbf{x}\in\mathbb{R}^{n\times p}}{\text{minimize}} \quad f(\mathbf{x}) + h(\mathbf{x}) + \delta_{\{0\}}(U\mathbf{x})$$

to get

$$\begin{aligned}
\mathbf{x}^{k+1} &= \mathrm{Prox}_{\alpha f}(\mathbf{x}^k - \alpha U\mathbf{u}^k - \alpha\nabla h(\mathbf{x}^k))\\
\mathbf{u}^{k+1} &= \mathbf{u}^k + \beta U\left(2\mathbf{x}^{k+1} - \mathbf{x}^k + \alpha\left(\nabla h(\mathbf{x}^k) - \nabla h(\mathbf{x}^{k+1})\right)\right).
\end{aligned}$$

Initialize $\mathbf{u}^0 = 0$ but set $\mathbf{x}^0$ arbitrarily. To eliminate U, define $\mathbf{z}^k = \mathbf{x}^k - \alpha U\mathbf{u}^k - \alpha\nabla h(\mathbf{x}^k)$. Choose $\beta = \alpha^{-1}$ for simplicity and rearrange the terms to get

$$\begin{aligned}
\mathbf{x}^{k+1} &= \mathrm{Prox}_{\alpha f}(\mathbf{z}^k)\\
\mathbf{z}^{k+1} &= \mathbf{z}^k - \mathbf{x}^{k+1} + \frac{1}{2}(I+W)\left(2\mathbf{x}^{k+1} - \mathbf{x}^k + \alpha\left(\nabla h(\mathbf{x}^k) - \nabla h(\mathbf{x}^{k+1})\right)\right),
\end{aligned}$$

where we initialize $\mathbf{z}^0 = \mathbf{x}^0 - \alpha\nabla h(\mathbf{x}^0)$ but set $\mathbf{x}^0$ arbitrarily.

NIDS

This method is called the *Network InDependent Step-size* (NIDS) method

$$\mathbf{x}^{k+1} = \mathrm{Prox}_{\alpha f}(\mathbf{z}^k)$$
$$\mathbf{z}^{k+1} = \mathbf{z}^k - \mathbf{x}^{k+1} + \frac{1}{2}(I+W)\left(2\mathbf{x}^{k+1} - \mathbf{x}^k + \alpha\left(\nabla h(\mathbf{x}^k) - \nabla h(\mathbf{x}^{k+1})\right)\right).$$

Assume total duality holds, $h_1, \ldots, h_n$ are L-smooth, and $\alpha \in (0, 2/L)$. Then $\mathbf{x}^k \to \mathbf{x}^\star$. Note that the choice of $\alpha \in (0, 2/L)$ is independent of the mixing matrix and, thus, the network topology.

PG-EXTRA vs. NIDS

The stepsize requirement of NIDS is more favorable than that of PG-EXTRA. The stepsize α of PG-EXTRA is affected by the eigenvalues of W, thus, also by the network structure. This not only limits the size of α but also make the choice of α more difficult when the network is not fully known. In contrast, the stepsize α of NIDS can be chosen independently of W.

On the other hand, PG-EXTRA can compute $W\mathbf{x}^k$ and $\nabla h(\mathbf{x}^k)$ simultaneously, but NIDS must do its corresponding steps sequentially. Therefore, PG-EXTRA can be implemented more efficiently than NIDS.

PG-EXTRA vs. NIDS

In the case $f = 0$, we can simplify the two methods to

$$\begin{aligned} \text{PG-EXTRA:} \quad \mathbf{x}^{k+1} &= \widetilde{W}(2\mathbf{x}^k - \mathbf{x}^{k-1}) + \alpha(\nabla h(\mathbf{x}^{k-1}) - \nabla h(\mathbf{x}^k)) \\ \text{NIDS:} \quad \mathbf{x}^{k+1} &= \widetilde{W}\left(2\mathbf{x}^k - \mathbf{x}^{k-1} + \alpha(\nabla h(\mathbf{x}^{k-1}) - \nabla h(\mathbf{x}^k))\right), \end{aligned}$$

where $\widetilde{W} = \frac{1}{2}(W + I)$. PG-EXTRA resembles DGD while NIDS resembles diffusion.