Distributed and Decentralized Optimization

Ernest K. Ryu and Wotao Yin

Large-Scale Convex Optimization via Monotone Operators

All the ADMM methods in this slides are classical 2-block ADMM, equivalent to Douglas-Rachford thus convergence is implied by only convexity for any step size

Distributed optimization

Distributed optimization uses a set of networked computers, called agents, to solve optimization problems.

Motivation: When an algorithm running on one computer does not meet the required performance, one can

1. upgrade the computer (CPU, memory) and run the same algorithm;
2. use more computers, decompose the problem, and run a distributed optimization algorithm.

Distributed optimization

Distributed optimization uses a set of networked computers, called agents, to solve optimization problems.

Motivation: When an algorithm running on one computer does not meet the required performance, one can

1. upgrade the computer (CPU, memory) and run the same algorithm;
2. use more computers, decompose the problem, and run a distributed optimization algorithm.

Approach 1 is less cost effective and may never reach the required performance.

Approach 2 is the more favorable (often the only) choice of solving extremely large optimization problems.

Decentralized optimization

We distinguish between *distributed methods* and *decentralized methods*.

Decentralized optimization

We distinguish between *distributed methods* and *decentralized methods*.

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

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

Decentralized optimization

We distinguish between *distributed methods* and *decentralized methods*.

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

Problem and setup

In this lecture, we solve

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

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

Problem and setup

In this lecture, we solve

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

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

Setup: 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 $x^\star$.

Problem and setup

In this lecture, we solve

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

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

Setup: 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 $x^\star$.

The textbook chapter considers the more general formulation

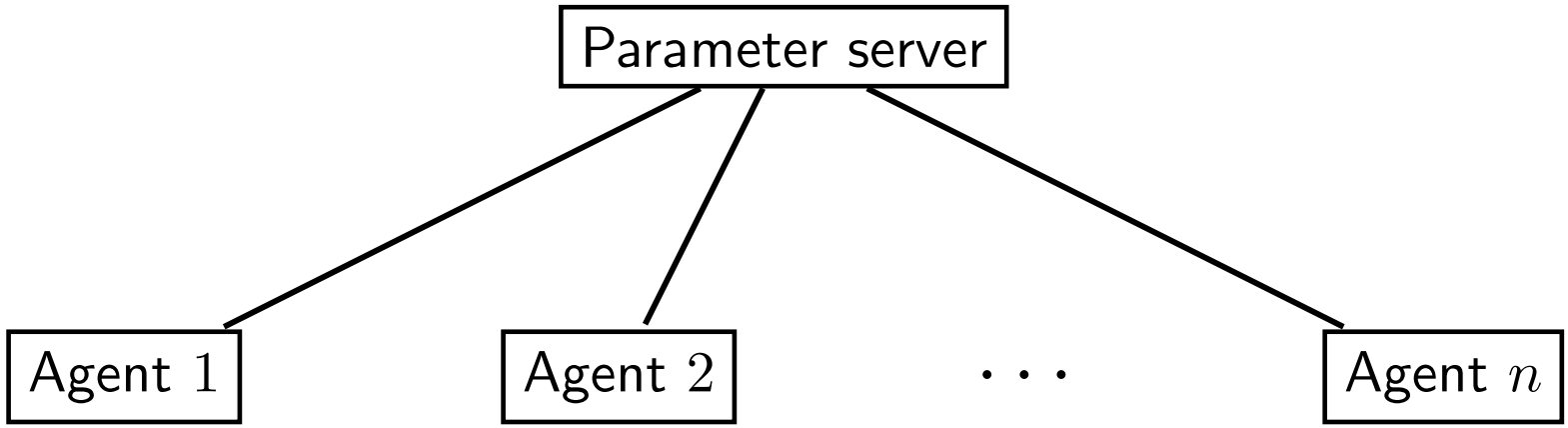
$$\underset{x\in\mathbb{R}^p}{\text{minimize}} \quad r(x) + \sum_{i=1}^{n} (f_i(x) + h_i(x)), \tag{2}$$

where r is CCP and proximable.

Outline

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.

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

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

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

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

FBS is:

$$x_i^{k+1/2} = x^k - \alpha\nabla h_i(x^k)$$

$$x^{k+1} = \frac{1}{n}\sum_{i=1}^{n} x_i^{k+1/2}$$

Distributed gradient method

Equivalent to:

$$\bar{g}^k = \frac{1}{n}\sum_{i=1}^{n} \nabla h_i(x^k)$$
$$x^{k+1} = x^k - \alpha \bar{g}^k$$

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.)

Distributed gradient method

Equivalent to:

$$\bar{g}^k = \frac{1}{n}\sum_{i=1}^{n} \nabla h_i(x^k)$$
$$x^{k+1} = x^k - \alpha \bar{g}^k$$

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.

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:

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

Rewrite to fit ADMM's form:

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

Apply ADMM:

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

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

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

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.

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

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

Primal decomposition

Consider

$$\underset{\substack{x_1,\ldots,x_n\in\mathbb{R}^p\\ z\in\mathbb{R}^q}}{\text{minimize}} \quad \frac{1}{n}\sum_{i=1}^{n} f_i(x_i, z).$$

With $\phi_i(z) = \inf_x f_i(x, z)$, problem is equivalent to

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

Primal decomposition

Consider

$$\underset{\substack{x_1,\ldots,x_n\in\mathbb{R}^p\\ z\in\mathbb{R}^q}}{\text{minimize}} \quad \frac{1}{n}\sum_{i=1}^{n} f_i(x_i,z).$$

With $\phi_i(z) = \inf_x f_i(x,z)$, problem is equivalent to

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

If $\phi_1,\ldots,\phi_n$ are differentiable, use distributed gradient method

$$g_i^k \in \nabla\phi_i(z^k)$$

$$z^{k+1} = z^k - \frac{\alpha}{n}\sum_{i=1}^{n} g_i^k$$

See Exercise 11.2 for computing subgradients of $\phi_1,\ldots,\phi_n$. When $\phi_1,\ldots,\phi_n$ not differentiable, can use subgradient method of §7.

Dual decomposition

Consider the equivalent problem

$$\begin{array}{ll} \underset{\substack{x_1,\ldots,x_n\in\mathbb{R}^p \\ z_1,\ldots,z_n\in\mathbb{R}^q \\ y\in\mathbb{R}^q}}{\text{minimize}} & \sum_{i=1}^n f_i(x_i,z_i) \\ \text{subject to} & z_i = y, \end{array}$$

generated by the Lagrangian

$$\mathbf{L}(x,y,z,v) = \sum_{i=1}^n f_i(x_i,z_i) - \langle v_i, z_i - y\rangle.$$

The dual problem is

$$\begin{array}{ll} \underset{v_1,\ldots,v_n \in \mathbb{R}^q}{\text{maximize}} & -\sum_{i=1}^n \psi_i(v_i) \\ \text{subject to} & v_1 + \cdots + v_n = 0, \end{array}$$

with

$$\psi_i(v_i) = \sup_{\substack{x_i \in \mathbb{R}^p \\ z_i \in \mathbb{R}^q}} \{-f_i(x_i, z_i) + \langle v_i, z_i \rangle\} = f_i^*(0, v_i).$$

When $\psi_1, \ldots \psi_n$ are differentiable, use projected gradient in a distributed manner

$$\begin{aligned} g_i^k &\in \nabla \psi_i(v_i^k) \\ v_i^{k+1} &= v_i^k - \alpha(g_i^k - \bar{g}^k) \end{aligned}$$

where $\bar{g}^k = (1/n)(g_1^k + \cdots + g_n^k)$. See Exercise 11.3 for computing subgradients of $\psi_1, \ldots, \psi_n$. (When $\psi_1, \ldots, \psi_n$ not differentiable, can use 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.

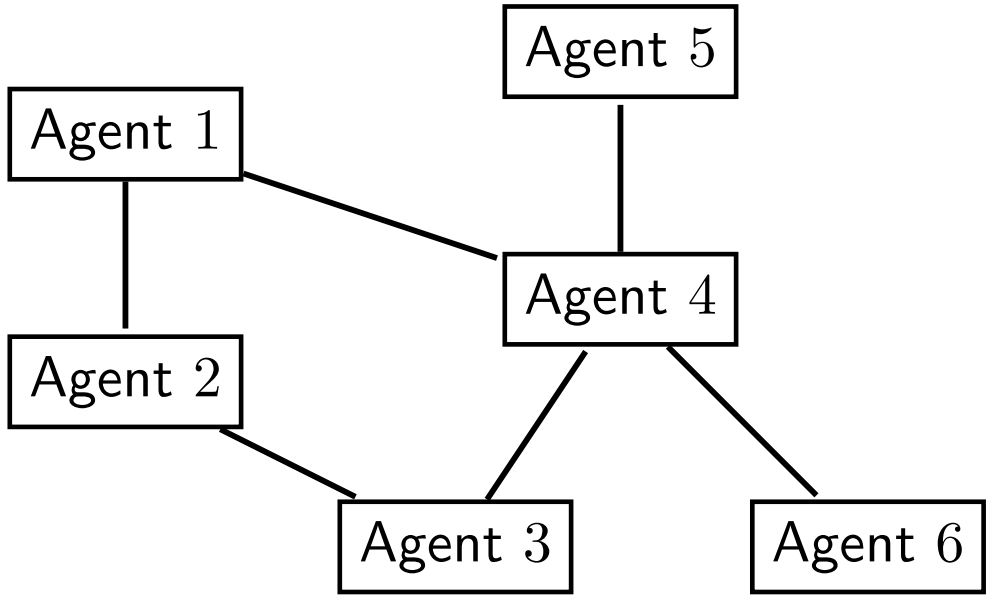
Networks and graphs

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-loop, 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

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

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.

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

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

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

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

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.

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

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

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

$$x_i^{k+1} = \underset{x_i \in \mathbb{R}^p}{\operatorname{argmin}} \left\{ f_i(x_i) + \sum_{j \in N_i} \left(\langle u^k_{\{i,j\},i}, x_i - y^k_{\{i,j\}} \rangle + \frac{\alpha}{2} \|x_i - y^k_{\{i,j\}}\|^2 \right) \right\} \quad \forall i \in V$$

$$y_e^{k+1} = \underset{y_e \in \mathbb{R}^p}{\operatorname{argmin}} \left\{ \sum_{t=i,j} \left(\langle u^k_{e,t}, x_t^{k+1} - y_e \rangle + \frac{\alpha}{2} \|x_t^{k+1} - y_e\|^2 \right) \right\} \quad \forall e = \{i,j\} \in E$$

$$u_{e,t}^{k+1} = u_{e,t}^k + \alpha(x_t^{k+1} - y_e^{k+1}) \qquad \forall e = \{i,j\} \in E,\ t = i,j.$$

We simplify further.

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

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

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

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

for all $i \in V$.

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 $i \in V$

$$x_i^{k+1} = \mathrm{Prox}_{(\alpha|N_i|)^{-1} f_i(x_i)}(v_i^k)$$

$$a_i^{k+1} = \frac{1}{|N_i|} \sum_{j \in N_i} x_j^{k+1}$$

$$v_i^{k+1} = v_i^k + a_i^{k+1} - \frac{1}{2} a_i^k - \frac{1}{2} x_i^k$$

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

Decentralized ADMM

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

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).

Synchronization

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.

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:

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

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

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

For $A \succeq 0$, define $\|\mathbf{x}\|_A^2 = \langle \mathbf{x}, A\mathbf{x} \rangle$. Specifically, $\|\mathbf{x}\|^2 = \|\mathbf{x}\|_I^2 = \langle \mathbf{x}, \mathbf{x} \rangle$.

Define

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

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

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

We say $\mathbf{x} = \mathrm{stack}(x_1, \ldots, x_n)$ is *in consensus* if $x_1 = \cdots = x_n$.
Any feasible point of (3) 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. W_{ij} may be nonzero only if i and j are directly linked.)

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

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

Example: Local averaging matrix

With mixing matrix

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

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

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

we can express decentralized ADMM as

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

Example: Decentralized averaging

Agent $i \in V$ has a vector $x_i \in \mathbb{R}^p$. The 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 $\mathbf{x}^0 = \mathrm{stack}(x_1, \ldots, x_n)$ and a decentralized mixing matrix $W \in \mathbb{R}^{n\times n}$.

Converges to $\bar{\mathbf{x}} = \mathrm{stack}(\bar{x}, \ldots, \bar{x})$ for all $\mathbf{x}^0$ if and only if $W\mathbf{1} = \mathbf{1}$, $\mathbf{1}^\intercal W = \mathbf{1}^\intercal$, and $1 = |\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$. (See Exercise 11.4)

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.
The 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:

$$W = W^\intercal \tag{4a}$$
$$\mathcal{N}(I - W) = \operatorname{span}(\mathbf{1}) \tag{4b}$$
$$1 = |\lambda_1| > \max\{|\lambda_2|, \ldots, |\lambda_n|\}. \tag{4c}$$

(4a) was not assumed in decentralized ADMM or averaging, but it is common; methods with symmetric W tend to be easier to analyze.
(4b) implies $\mathbf{x}$ is in consensus if and only if $\mathbf{x} = W\mathbf{x}$ and is required for almost all decentralized optimization methods.
(4c) is assumed to establish the convergence of certain methods. Note that (4a) implies the eigenvalues are real.

Example: Laplacian-based mixing matrix

The mixing matrix

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

where L is the so-called graph Laplacian defined by

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

for $i, j \in \{1, \ldots, n\}$ and τ is a constant satisfying $\tau > \frac{1}{2}\lambda_{\max}(L)$ satisfies $W = W^\intercal$, $W\mathbf{1} = \mathbf{1}$, and $1 = \lambda_1 > \max\{|\lambda_2|, \ldots, |\lambda_n|\}$.

Example: Metropolis mixing matrix

The mixing matrix

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

for $i, j \in \{1, \ldots, n\}$ and $\varepsilon > 0$ satisfies $W = W^\intercal$, $W\mathbf{1} = \mathbf{1}$, and $1 = \lambda_1 > \max\{|\lambda_2|, \ldots, |\lambda_n|\}$.

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

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

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 $\mathcal{N}(I - W) = \mathrm{span}(\mathbf{1})$ holds; if the Markov chain is irreducible and aperiodic, then $1 = \lambda_1 > \max\{|\lambda_2|, \ldots, |\lambda_n|\}$ holds.

A Markov chain is irreducible if every state can be reached from every other state. A state of a Markov chain is periodic if the chain can return to the state only at multiples of some integer larger than 1. A Markov chain is aperiodic if none of its states is periodic.

Inexact decentralized methods (using a mixing matrix)

Consider the setup with $f_1 = \cdots = f_n = 0$ and a mixing matrix $W \in \mathbb{R}^{n \times n}$ satisfying $W = W^\intercal$, $\mathcal{N}(I - W) = \operatorname{span}(\mathbf{1})$, and $\lambda_1 > \max\{\lambda_2, \ldots, \lambda_n\}$. We write (1) equivalently as

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^{n \times p}}{\text{minimize}} & h(\mathbf{x}) \\ \text{subject to} & (I - W)\mathbf{x} = 0. \end{array} \tag{5}$$

We consider inexact decentralized methods, DGD and Diffusion, which solve penalty formulations that approximate (5). These inexact methods, when they converge, converge to an approximation solution.

Decentralized gradient descent (DGD)

Consider the penalty formulation

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

We expect this formulation to approximate (5) well when $\alpha > 0$ is small.

Gradient descent with stepsize α applied to this penalty formulation is

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

This is decentralized gradient descent (DGD) or the combine-then-adapt method. If the penalty formulation has a solution, $h_1,\dots,h_n$ are L_h-smooth, and $\alpha \in (0, (1+\lambda_n(W))/L_h)$, then $\mathbf{x}^k$ converges to a solution of the penalty formulation.

Diffusion

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

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

Variable metric gradient descent §2.8 with $\alpha^{-1}W^{-1}$ as the metric is

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

This is the method of diffusion or the adapt-then-combine method. If 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.

DGD vs. Diffusion

Advantages of diffusion:

- Diffusion allows larger stepsizes, which often leads to faster convergence.

Advantages of DGD:

- Does not require the additional assumption $W \succ 0$; $W\mathbf{x}^k$ and $\alpha\nabla h(\mathbf{x}^k)$ can be computed simultaneously.

When $W \not\succ 0$, we can still use diffusion with $(1-\theta)I + \theta W$ and $\theta \in (0, 1/(1-\lambda_{\min}(W)))$. Since $\lambda_{\min}(W) > -1$, $\theta = \frac{1}{2}$ always works.

Exact decentralized methods (using a mixing matrix)

Since $I - W \succeq 0$, there exists a symmetric $U \in \mathbb{R}^{n \times n}$ such that

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

This U satisfies $\mathcal{N}(U) = \operatorname{span}(\mathbf{1})$.

The general problem (1) is equivalent to

$$\begin{array}{ll} \underset{\mathbf{x} \in \mathbb{R}^{n \times p}}{\text{minimize}} & f(\mathbf{x}) + h(\mathbf{x}) \\ \text{subject to} & U\mathbf{x} = 0. \end{array} \tag{6}$$

We present two methods, PG-EXTRA and NIDS, what converge to its exact solution. They utilize W, while U is used only in analysis.

Exercise 3.5: Condat-Vũ, the other version

Consider the problem

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ f(x)+h(x)+g(Ax),$$

In the derivation of Condat-Vũ, if we *instead* use the metric

$$M=\begin{bmatrix}(1/\alpha)I & A^\intercal\\ A & (1/\beta)I\end{bmatrix},$$

then we get the method

$$\begin{aligned}u^{k+1}&=\mathrm{Prox}_{\beta g^*}(u^k+\beta Ax^k)\\ x^{k+1}&=\mathrm{Prox}_{\alpha f}(x^k-\alpha A^\intercal(2u^{k+1}-u^k)-\alpha\nabla h(x^k)).\end{aligned}$$

If total duality holds, $\alpha,\beta>0$, $\alpha L/2+\alpha\beta\lambda_{\max}(A^\intercal A)<1$, and h is L-smooth, then $x^k\to x^\star$ and $u^k\to u^\star$.

PG-EXTRA

Apply the last slide to (6) with $g = \delta_0$ (thus $\mathrm{Prox}_{\beta g^*} = \mathbb{I}$) and $A = A^\intercal = U$ to get

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

Simplify the method by choosing $\beta = \alpha^{-1}$ and introduce $\mathbf{w}^k = \frac{1}{\beta} U \mathbf{u}^k$ to get the PG-EXTRA method:

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

(Initialize $\mathbf{x}^0$ arbitrarily and $\mathbf{w}^0 = 0$, which voids computing $U\mathbf{u}^0$)

PG-EXTRA is decentralized when W is decentralized. If total duality holds, $0 < \alpha < (1 + \lambda_{\min}(W))/L$ (using $\lambda_{\max}(U^2) = \frac{1}{2} - \frac{1}{2}\lambda_{\min}(W)$), and $h_1, \ldots, h_n$ are L-smooth, then we have $\mathbf{x}^k \to \mathbf{x}^\star$.

Review of PD3O

Consider

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

where h is L-smooth. PD3O is the method

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

(PD3O can be obtained by applying DYS FPI and BCV to the problem.)

If total duality holds, $\alpha,\beta>0$, $\alpha\beta\lambda_{\max}(A^\intercal A)\le 1$, and $\alpha\le 2/L$, then $x^{k+1/2}\to x^\star$.

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

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

To eliminate U, define $\mathbf{z}^k = \mathbf{x}^k - \alpha U \mathbf{u}^k - \alpha \nabla h(\mathbf{x}^k)$. Use $\beta = \alpha^{-1}$ to get

$$\mathbf{x}^{k+1} = \mathrm{Prox}_{\alpha f}(\mathbf{z}^k)$$

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

with arbitrary $\mathbf{x}^0$ and $\mathbf{z}^0 = \mathbf{x}^0 - \alpha \nabla h(\mathbf{x}^0)$. This uses $\mathbf{u}^0 = 0$, which avoids computing $U\mathbf{x}^0$.

This is the *Network InDependent Step-size* (NIDS) method. If total duality holds, $h_1, \ldots, h_n$ are L_h-smooth, and $\alpha \in (0, 2/L_h)$, then $\mathbf{x}^k \to \mathbf{x}^\star$.

The choice of $\alpha \in (0, 2/L_h)$ is independent of the mixing matrix and, thus, the network topology.

PG-EXTRA vs NIDS

The step size α of PG-EXTRA depends on the eigenvalues of W. 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, NIDS allows the stepsize α to be larger and to be chosen independent of W.

On the other hand, PG-EXTRA can compute $W\mathbf{x}^k$ and $\nabla h(\mathbf{x}^k)$ simultaneously, but NIDS cannot.

With $f = 0$ and $\widetilde{W} = \frac{1}{2}(W + I)$, the methods simplify to

$$\text{PG-EXTRA:} \quad \mathbf{x}^{k+1} = \widetilde{W}(2\mathbf{x}^k - \mathbf{x}^{k-1}) + \alpha(\nabla h(\mathbf{x}^{k-1}) - \nabla h(\mathbf{x}^k))$$

$$\text{NIDS:} \quad \mathbf{x}^{k+1} = \widetilde{W}\left(2\mathbf{x}^k - \mathbf{x}^{k-1} + \alpha(\nabla h(\mathbf{x}^{k-1}) - \nabla h(\mathbf{x}^k))\right).$$

PG-EXTRA resembles DGD while NIDS resembles diffusion.

Conclusion

Distributed optimization takes advantages of problem structures and can solve extremely large optimization problems.

With a central coordinator, the methods rely on aggregating distributed gradients or averaging distributed iterates through a reduce operation.

Decentralized optimization uses neighborhood communication (i.e., decentralized mixing) instead of a global reduce operation.

By applying splitting and variable metric techniques, we obtain decentralized optimization methods.