Distributed and Decentralized Optimization

Ernest K. Ryu

Large-Scale Convex Optimization:
Algorithms and Analyses via Monotone Operators

Stanford University
EE392F Spring 2023

Distributed and decentralized optimization

In this section, 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, 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

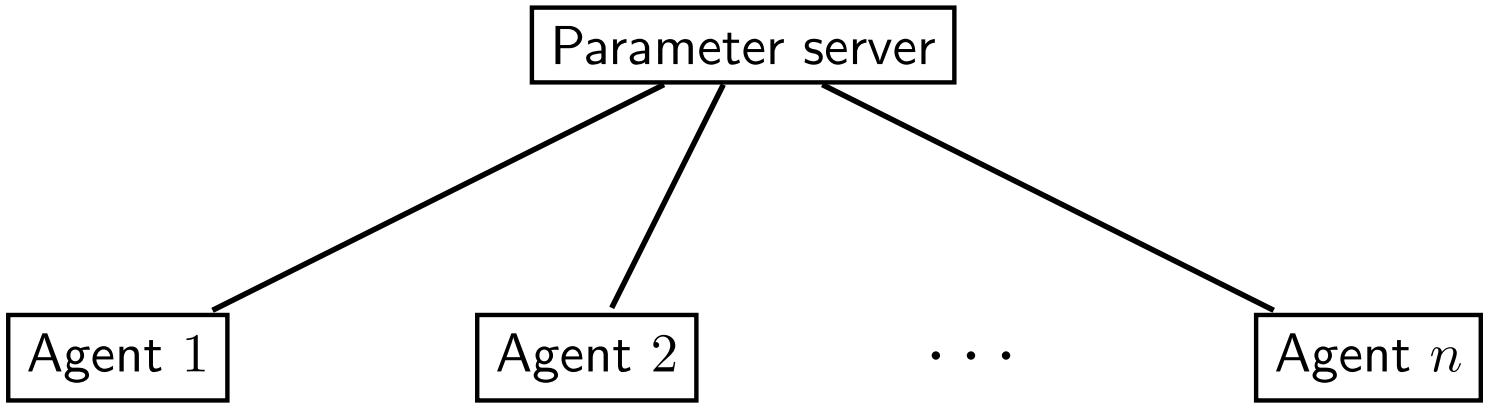
Distributed optimization with centralized consensus

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.

Distributed gradient method

Consider

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

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

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

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.)

This method is (centralized) distributed:

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

(ii) Agents coordinate to compute 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:

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

Rewrite to fit ADMM's form:

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

Distributed ADMM

Apply ADMM:

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

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

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

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

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

$$u_i^{k+1} = u_i^k + \alpha(x_i^{k+1} - \bar{x}^{k+1})$$

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

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.

Primal decomposition

Consider

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

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

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

Use distributed gradient method

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

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

See Exercise 11.X 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

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

Dual decomposition

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

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

Use projected gradient in a distributed manner

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

where $\bar{g}^k = (1/n)(g_1^k + \cdots + g_n^k)$. See Exercise 11.X 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* of uses this first meaning.

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

Networks and graphs

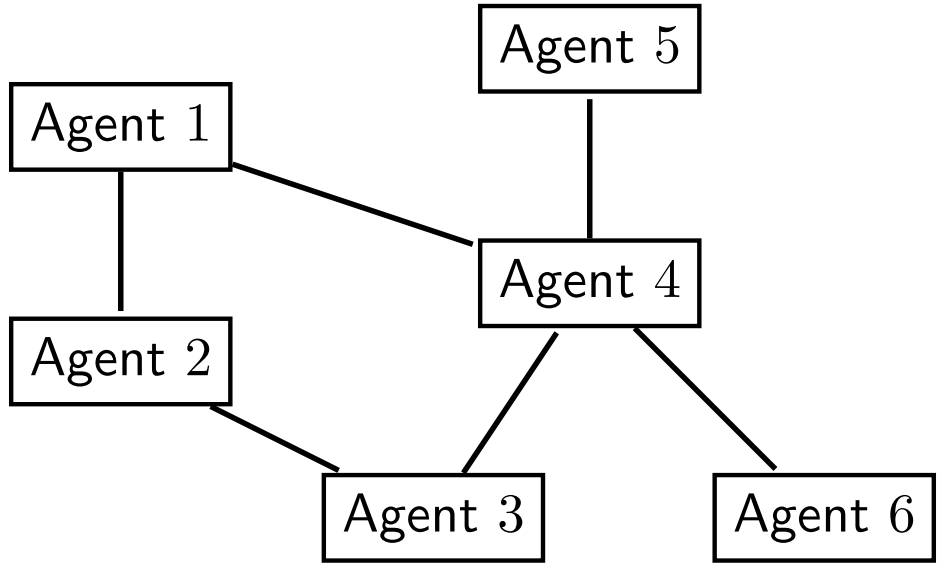
A graph $G = (V, E)$ represents a network. V is set of nodes and E is set of edges. Assume

- 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

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

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

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

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.

Not delegating any agent as the central agent 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

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

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

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

Decentralized ADMM

Express ADMM as

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

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

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

We simplify further.

Decentralized ADMM

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

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

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

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

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 $i \in V$

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

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

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

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

Decentralized ADMM

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 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

Decentralized notation

Define stack operator and use boldface to denote stacked variables:

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

Write $x^\star \in \mathbb{R}^p$ and $\mathbf{x}^\star = \mathrm{stack}(x^\star, \ldots, x^\star) \in \mathbb{R}^{n \times p}$ 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, $\|\mathbf{x}\|^2 = \|\mathbf{x}\|_I^2 = \langle \mathbf{x}, \mathbf{x} \rangle$.

Decentralized notation

Define

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

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

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

We say $\mathbf{x} = \mathrm{stack}(x_1, \ldots, x_n)$ is *in consensus* if $x_1 = \cdots = x_n$.
A solution (or 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 $x_i \in \mathbb{R}^p$ 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 $\mathbf{x}^0 = \mathrm{stack}(x_1, \ldots, x_n)$ and a decentralized mixing matrix $W \in \mathbb{R}^{n\times n}$. Converges for all $\mathbf{x}^0$ if and only if $W\mathbf{1} = \mathbf{1}$, $\mathbf{1}^\mathsf{T}W = \mathbf{1}^\mathsf{T}$, and $1 = |\lambda_1| > |\lambda_2| \geq \cdots \geq |\lambda_n|$. (Cf. Exercise 11.4.)

Condition $W\mathbf{1} = \mathbf{1}$ implies $\mathbf{x}$-vectors in consensus are fixed points.
Condition $\mathbf{1}^\mathsf{T}W = \mathbf{1}^\mathsf{T}$ 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:

$$W = W^\intercal \tag{3a}$$
$$\mathcal{N}(I - W) = \operatorname{span}(\mathbf{1}) \tag{3b}$$
$$1 = |\lambda_1| > \max\{|\lambda_2|, \ldots, |\lambda_n|\}. \tag{3c}$$

(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.X.)

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 $\mathcal{N}(I - W) = \mathrm{span}(\mathbf{1})$ 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.

Dynamic mixing matrix

We assume the mixing matrix, once given, is fixed and do not change with iteration. It is possible however, to use a series of dynamic mixing matrices.

Even when the graph is fixed, dynamic mixing matrices can be used to get faster convergence. It is possible to compute a decentralized average in $\log_2(n)$ steps using so-called exponential-2 dynamic mixing matrices. This is impossible with a fixed mixing matrix (unless the graph is complete.) See Exercises 11.X.

Inexact decentralized methods

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

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

We consider inexact decentralized methods that 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 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. (Exercise 11.X.)

DGD vs. Diffusion

Advantages of diffusion:

- Diffusion allows larger stepsizes, which, loosely speaking, often leads to faster convergence.
- Solutions to the penalty formulation of diffusion better approximate solutions to the original problem (5) then those of DGD. See Exercises 11.X and 11.X.

Advantages of DGD:

- Does not require the additional assumption $W \succ 0$.

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

Exact decentralized methods

Consider two symmetric mixing matrices $W, \widetilde{W} \in \mathbb{R}^{n\times n}$ satisfying

$$W \preceq \widetilde{W} \preceq \frac{1}{2}(I+W),$$
$$\mathcal{N}(\widetilde{W}-W) = \mathcal{N}(I-W) = \operatorname{span}(\mathbf{1}).$$

$\widetilde{W} = \frac{1}{2}(W+I)$ is a common choice. Since $\widetilde{W} - W \succeq 0$, there exists a symmetric $U \in \mathbb{R}^{n\times n}$ such that

$$U^2 = \widetilde{W} - W,$$

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

Consider the general problem (1), which is equivalent to

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

PG-EXTRA

Consider the Lagrangian

$$\mathbf{L}(\mathbf{x}, \mathbf{y}) = f(\mathbf{x}) + h(\mathbf{x}) + \langle \mathbf{y}, U\mathbf{x} \rangle,$$

whose saddle subdifferential is

$$\partial \mathbf{L}(\mathbf{x}, \mathbf{y}) = \begin{bmatrix} \partial f(\mathbf{x}) \\ 0 \end{bmatrix} + \begin{bmatrix} \nabla h & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} + \begin{bmatrix} 0 & U \\ -U & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}.$$

Define

$$V_\eta = \eta_1 (I - \widetilde{W}) + \eta_2 (I - W) + \eta_3 U^2,$$

where $\eta_1, \eta_2, \eta_3 \in \mathbb{R}$ are nonnegative, and

$$\mathbb{A}(\mathbf{x}, \mathbf{y}) = \underbrace{\begin{bmatrix} \partial f(\mathbf{x}) \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & \frac{1}{2}U \\ -\frac{1}{2}U & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}}_{=\mathbb{F}(\mathbf{x}, \mathbf{y})} + \underbrace{\begin{bmatrix} \nabla h & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} + \begin{bmatrix} \alpha^{-1} V_\eta & \frac{1}{2}U \\ -\frac{1}{2}U & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix}}_{=\mathbb{H}(\mathbf{x}, \mathbf{y})}.$$

Then $\operatorname{Zer} \mathbb{A}(\mathbf{x}, \mathbf{y}) = \operatorname{Zer} \partial \mathbf{L}(\mathbf{x}, \mathbf{y})$, since $U\mathbf{x} = 0$ implies $V_\eta \mathbf{x} = 0$. We apply variable metric FBS to $\mathbb{A}$, rather than $\partial \mathbf{L}$.

PG-EXTRA

The matrix

$$M = \begin{bmatrix} \alpha^{-1} I & -\frac{1}{2} U \\ -\frac{1}{2} U & \beta^{-1} I \end{bmatrix}$$

satisfies $M \succ 0$ if $4I \succ \alpha\beta U^2$. FPI with $(M + \mathbb{F})^{-1}(M - \mathbb{H})$ is

$$\begin{aligned} \mathbf{x}^{k+1} &= \mathrm{Prox}_{\alpha f}\big((I - V_\eta)\mathbf{x}^k - \alpha \nabla h(\mathbf{x}^k) - \alpha U \mathbf{y}^k\big) \\ \mathbf{y}^{k+1} &= \mathbf{y}^k + \beta U \mathbf{x}^{k+1}. \end{aligned}$$

We initialize $\mathbf{y}^0 = \beta U \mathbf{x}^0$ (but set $\mathbf{x}^0$ arbitrarily). Let $\eta_1 = 1$, $\eta_2 = 0$, $\eta_3 = 0$, $\beta = 1/\alpha$ and substitute $\mathbf{y}^k = \sum_{i=0}^k U \mathbf{x}^i$ to get

$$\mathbf{x}^{k+1} = \mathrm{Prox}_{\alpha f}\left(W\mathbf{x}^k - \alpha \nabla h(\mathbf{x}^k) - \sum_{j=0}^{k-1} (\widetilde{W} - W)\mathbf{x}^j \right).$$

This is PG-EXTRA. Method is decentralized when W and $\widetilde{W}$ are decentralized mixing matrices. If $\mathbf{L}$ has a saddle point, $h_1, \ldots, h_n$ are L_h-smooth, and $\alpha \in (0, 2\lambda_{\min}(\widetilde{W})/L_h)$, then $\mathbf{x}^k \to \mathbf{x}^\star$.

NIDS

Consider the equivalent formulation with an indicator function

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

Apply PD3O (or equivalently PAPC/PDFP^{2}O):

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

NIDS

Define $\mathbf{z}^k = \mathbf{x}^k - \alpha U \mathbf{u}^k - \alpha \nabla h(\mathbf{x}^k)$:

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

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

This is the Network InDependent Step-size (NIDS) method. If $\mathbf{L}$ has a saddle point, $h_1, \ldots, h_n$ are L_h-smooth, $\alpha \in (0, 2/L_h)$ and $\beta > 0$ satisfies $I - \alpha\beta(\widetilde{W} - W) \succeq 0$, then $\mathbf{x}^k \to \mathbf{x}^\star$.

By setting $\widetilde{W} = \frac{1}{2}(W + I)$ and $\beta = \alpha^{-1}$, we obtain $I - \alpha\beta(\widetilde{W} - W) = \frac{1}{2}(W + I) \succeq 0$, so 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 $\widetilde{W}$. 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 and $\widetilde{W}$.

When $f = 0$, $\beta = 1$, 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.