Tutorial on the structure of distributed optimization algorithms

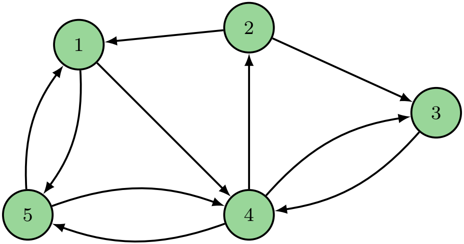
Bryan Van Scoy
Miami University

Laurent Lessard
Northeastern University

IEEE Conference on Decision and Control
December 13–15, 2023

Distributed optimization

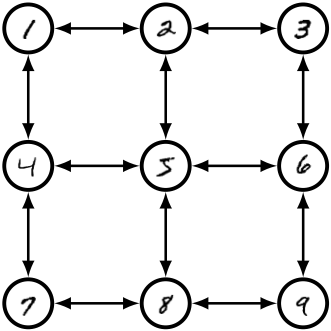
$$\begin{aligned} &\text{minimize} \quad && \sum_{i=1}^{n} f_i(y_i) \\ &\text{subject to} \quad && y_1 = y_2 = \ldots = y_n \end{aligned}$$



Want each agent to compute the global optimizer by communicating with local neighbors and performing local computations.

Application: Distributed machine learning

- each agent has a set of data and a local model
- agents collaboratively construct a global model



$$\begin{aligned} \underset{\theta_1,\ldots,\theta_n}{\text{minimize}} \quad & \sum_{i=1}^{n} \sum_{(x,y)\in \mathsf{data}_i} \ell(y - m_{\theta_i}(x)) \\ \text{subject to} \quad & \theta_1 = \theta_2 = \ldots = \theta_n \end{aligned}$$

MNIST dataset: `http://yann.lecun.com/exdb/mnist/`

Objectives

a) Provide an overview of distributed optimization.

b) Describe the structure of distributed algorithms.

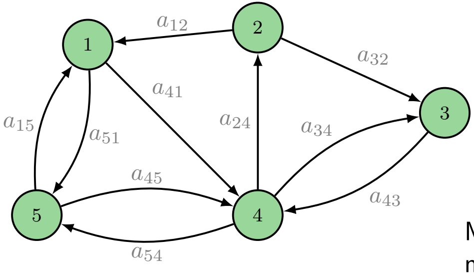
c) Use simulations to illustrate some algorithmic properties.

Context

- This talk: qualitative approach to algorithm analysis using control
- Alternative approach: quantitative

Model the communication network as a weighted directed graph.

Graph	Meaning
node	agent
edge	flow of information between two agents
weight	amount by which information is weighted



Multiplication by the Laplacian matrix L diffuses information.

$$(Lx)_i = \sum_{j=1}^{n} a_{ij}(x_i - x_j)$$

Gradient tracking

One well-known algorithm for consensus optimization is gradient tracking.

$$x_i^{k+1} = \sum_{j=1}^{n} a_{ij}\, x_j^k - \alpha y_i^k$$

$$y_i^{k+1} = \sum_{j=1}^{n} a_{ij}\, y_j^k + \nabla f_i(x_i^{k+1}) - \nabla f_i(x_i^k)$$

- agents communicate local information and compute local gradients
- y_i estimates the average gradient
- x_i applies gradient descent to the estimated average gradient

General algorithm form

Each agent i can...

- evaluate its local gradient

$$u_i = \nabla f_i(y_i)$$

- communicate information with neighbors

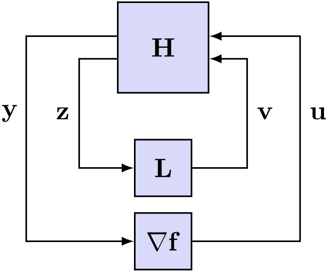
$$v_i = \sum_{j=1}^{n} a_{ij} \left(z_i - z_j \right)$$

- choose the points at which to evaluate the gradient and communicate

$$\begin{bmatrix} y_i \\ z_i \end{bmatrix} = H \begin{bmatrix} u_i \\ v_i \end{bmatrix}$$

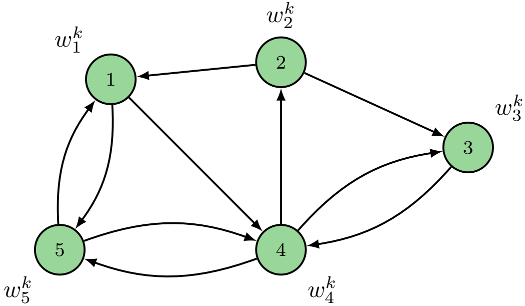
Compact form

- Bold signals are concatenated over all agents, e.g., $\boldsymbol{y} = (y_1, \ldots, y_n)$
- Combined gradient is $\nabla \boldsymbol{f} = \text{diag}(\nabla f_1, \ldots, \nabla f_n)$
- Combined Laplacian is $\boldsymbol{L} = L \otimes I_m$ where m is the dimension of z_i
- Combined system is $\boldsymbol{H} = \begin{bmatrix} I_n \otimes H^{11} & I_n \otimes H^{12} \\ I_n \otimes H^{21} & I_n \otimes H^{22} \end{bmatrix}$



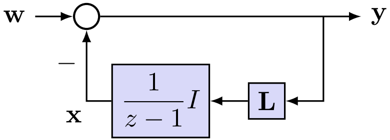
Dynamic average consensus

Each agent i has a (potentially time-varying) signal w_i^k.



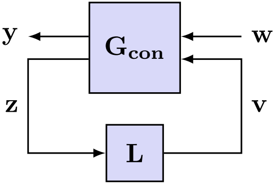
Want each agent to estimate the average signal $w_{\mathsf{avg}}^k = \dfrac{1}{n}\sum_{i=1}^{n} w_i^k$.

Proportional estimator



$$y_i^k = w_i^k - x_i^k$$
$$x_i^{k+1} = x_i^k + \sum_{j=1}^{n} a_{ij}\left(y_i^k - y_j^k\right)$$

General form



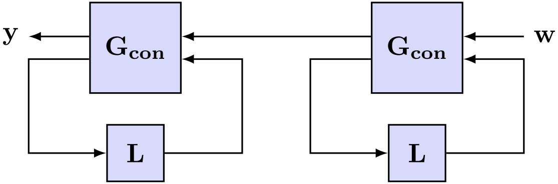
$$\begin{bmatrix} y_i \\ z_i \end{bmatrix} = G_{\text{con}} \begin{bmatrix} w_i \\ v_i \end{bmatrix}$$

$$v_i = \sum_{j=1}^{n} a_{ij}\left(z_i - z_j\right)$$

p^{th}-order estimator asymptotically tracks polynomials of degree $p-1$.

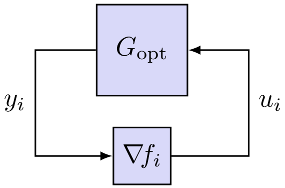
- A first-order estimator tracks constant signals
- A second-order estimator tracks ramp signals

One way to construct higher-order estimators: combine in series



Optimization methods

Consider the unconstrained optimization problem minimize $f(y)$.



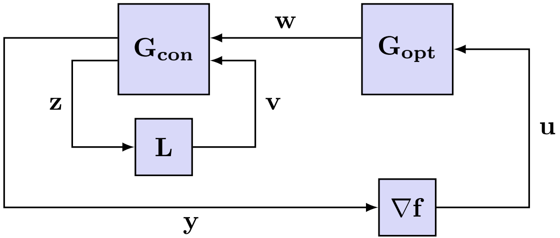
$$y_i = G_{\text{opt}}\, u_i$$

$$u_i = \nabla f_i(y_i)$$

Gradient method: $y^{k+1} = y^k - \alpha\, \nabla f(y^k)$ has $G_{\text{opt}}(z) = \frac{-\alpha}{z-1}$

The transfer function must have a pole at $z = 1$
so that all fixed points are stationary points.

Decomposition



- G_{opt} is an optimization method
- G_{con} is a second-order consensus estimator

Every algorithm decomposes in this form, and any optimization method and consensus estimator combine to form a valid algorithm.

Proof (idea): Can always factor H as $G_{\text{con}} \begin{bmatrix} G_{\text{opt}} & 0 \\ 0 & I_m \end{bmatrix}$

Simulation: Distributed least squares

- $n = 5$ agents
- objective function on agent i is the quadratic

$$f_i(y) = \tfrac{1}{2} y^\mathsf{T} A_i y - b_i^\mathsf{T} y$$

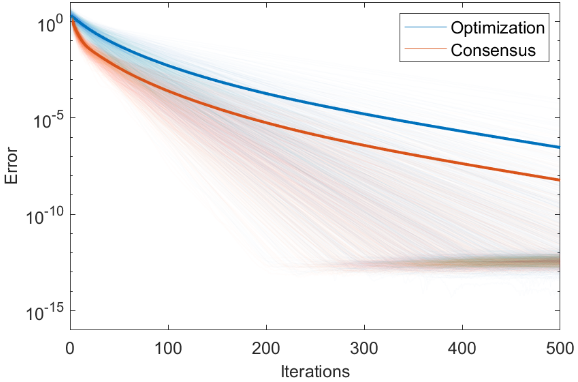
  parameterized by symmetric matrix $A_i \in \mathbb{R}^{3\times 3}$ and vector $b_i \in \mathbb{R}^3$
- sample A_i such that its eigenvalues are evenly spaced in $[\frac{1}{10}, 1]$
- sample each element of b_i from a standard normal distribution

Use simulations to illustrate algorithm properties of internal stability, acceleration, and robustness.

Optimility conditions

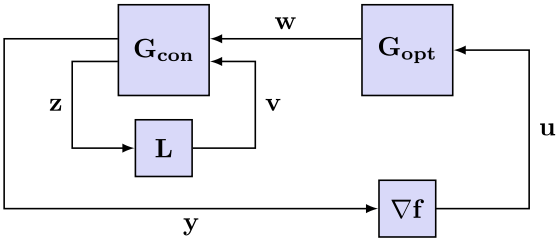
$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^{n} f_i(y_i) \\ \text{subject to} & y_1 = y_2 = \ldots = y_n \end{array}$$

	Condition	Error
Optimality	$\sum_{i=1}^{n} \nabla f_i(y_i) = 0$	$\left\lVert \sum_{i=1}^{n} \nabla f(y_i) \right\rVert$
Consensus	$y_1 = y_2 = \ldots = y_n$	$\sum_{i=1}^{n} \left\lVert y_i - \frac{1}{n} \sum_{j=1}^{n} y_j \right\rVert$



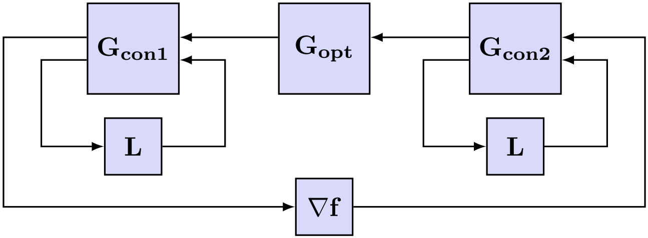
- Each thin trace is a trial (1000 total)
- The thick trace is the average over all trials
- The total error is the maximum of optimization and consensus errors

Internal stability

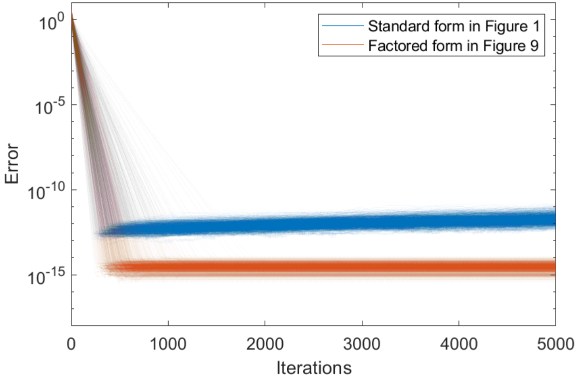


- At steady state, the average gradient is zero, but $u_i \neq 0$ in general
- This nonzero value is integrated by the optimization method
- The input w_i to the consensus estimator grows without bound

The algorithm is not internally stable.



- Can avoid this issue if the consensus estimator factors
- G_{con1} and G_{con2} are both *first-order* estimators
- The input to the optimization method is zero at steady state

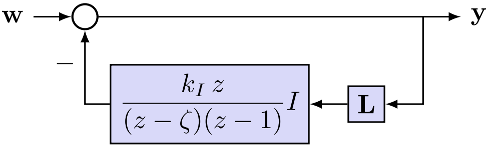


The factored form has better numerical conditioning.

Acceleration

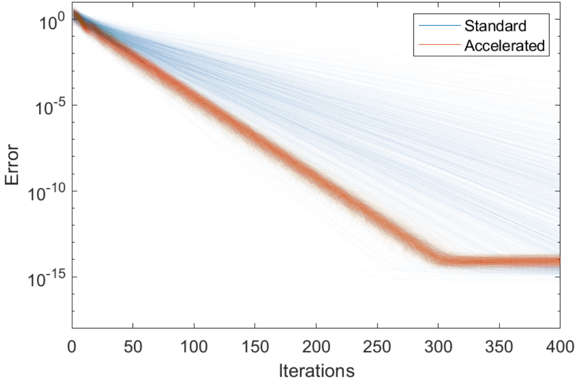
Accelerate convergence using extra (appropriately chosen) dynamics.

Accelerated consensus



Accelerated optimization

$$G_{\mathrm{opt}}(z) = \frac{-\alpha\,(z + \eta\,(z-1))}{(z-\beta)(z-1)}$$

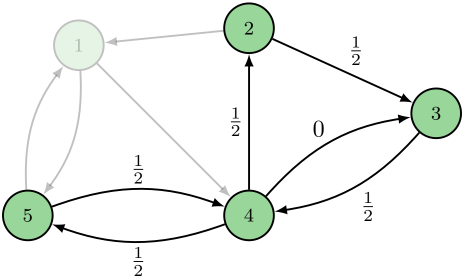


Using additional dynamics can accelerate convergence.

Parameters: $(\alpha, \beta, \eta) = (0.1, 0.8, 0)$ and $(\zeta, k_I) = (0.1, 1.1)$

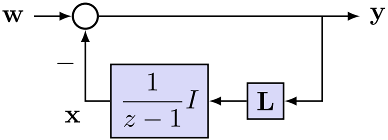
Robustness

- Suppose agent 1 leaves the network at iteration $k = 200$
- The other agents update their weights accordingly

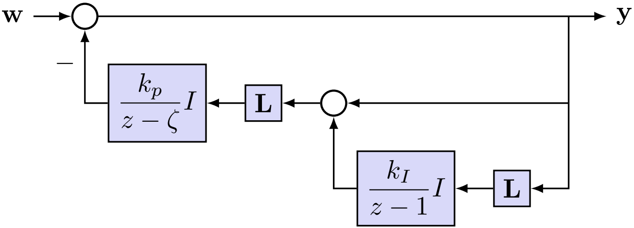


How does an algorithm respond to changes in the network?

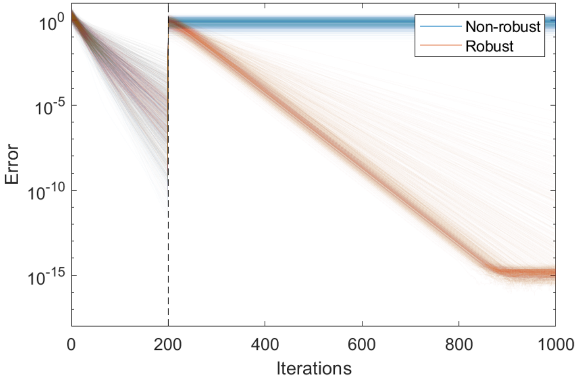
The P estimator requires the state to be initialized such that $\sum_i x_i = 0$.



The PI estimator has no such requirement.

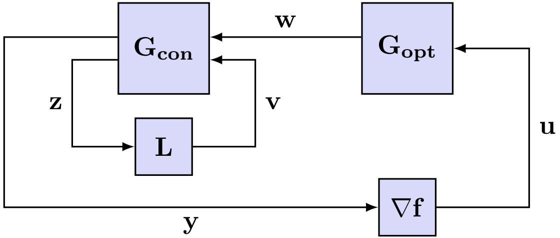


An estimator with no initialization requirement is called *robust*.



The algorithm with a robust estimator recovers from the change in network (after a transient).

Summary



a) Provided an overview of distributed optimization.

b) Described the structure of distributed algorithms.

c) Used simulations to illustrate some algorithmic properties.

- internal stability
- acceleration
- robustness