Control Tools for Distributed Optimization

Optimization and distributed algorithms

Prof. Ivano Notarnicola

Dept. of Electrical, Electronic, Information Eng.
Università di Bologna
ivano.notarnicola@unibo.it

Prof. Ruggero Carli

Dept. of Information Engineering
Università di Padova
ruggero.carli@unipd.it

SIDRA Ph.D. Summer School
July, 10-12 2025 • Bertinoro, Italy

Outline

- Descent algorithms (Gradient/ Newton-Raphson)
- On Operators (monotone, strongly monotone, Lipschitz continuous, Co-coercive)
- Convex set, convex and strongly convex functions
- Properties of the gradient operator
- Gradient algorithm
- Consensus Optimization over networks
- Elements of Graph Theory
- Consensus algorithms
- Distributed Gradient Descent
- Distributed Gradient Tracking

Outline

- Descent algorithms (Gradient/ Newton-Raphson)
- On Operators (monotone, strongly monotone, Lipschitz continuous, Co-coercive)
- Convex set, convex and strongly convex functions
- Properties of the gradient operator
- Gradient algorithm
- Consensus Optimization over networks
- Elements of Graph Theory
- Consensus algorithms
- Distributed Gradient Descent
- Distributed Gradient Tracking

Descent algorithms

Consider the following optimization problem

$$\min_x f(x)\,, \qquad f:\mathbb{R}^n \to \mathbb{R}, \qquad f \text{ convex, twice differentiable}$$

Iterative algorithms:

$$x_{k+1} = x_k + \alpha_k \Delta x_k$$

- α_k step-size;
- Δx_k descent direction, that is, $\nabla f(x_k)^\top \Delta x_k < 0$ (*if x_k is not a minimizer, i.e., $\nabla f(x_k) \neq 0$*)

α_k must be chosen in such a way

$$f(x_{k+1}) - f(x_k) < 0 \qquad \text{descent condition}$$

Descent algorithms

Gradient method :

$$\Delta x_k = -\nabla f(x_k) \qquad \Longrightarrow \qquad x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

Observe that $\nabla f(x_k)^\top \Delta x_k = -\|\nabla f(x_k)\|^2 < 0$ *(if x_k is not a minimizer, i.e., $\nabla f(x_k) \neq 0$)*

Descent algorithms

Gradient method :

$$\Delta x_k = -\nabla f(x_k) \qquad \Longrightarrow \qquad x_{k+1} = x_k - \alpha_k \nabla f(x_k)$$

Observe that $\nabla f(x_k)^\top \Delta x_k = -\|\nabla f(x_k)\|^2 < 0$ (*if* x_k *is not a minimizer, i.e.,* $\nabla f(x_k) \neq 0$)

Newton-Raphson method :

$$\Delta x_k = -\left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k) \qquad \Longrightarrow \qquad x_{k+1} = x_k - \alpha_k \left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k)$$

Observe that $\nabla f(x_k)^\top \Delta x_k = -\nabla f(x_k)^\top \left(\nabla^2 f(x_k)\right)^{-1} \nabla f(x_k) < 0$ (*if* x_k *is not a minimizer, i.e.,* $\nabla f(x_k) \neq 0$)

Today : gradient method

In this set of slides we will focus on **gradient method** for unconstrained problems and we will assume the function is differentiable.

Question : What to do if f is not differentiable?

Remark : Methods for nondifferentiable or constrained problems

- subgradient method
- proximal gradient method
- smoothing methods
- cutting-plane method

Outline

- Descent algorithms (Gradient/ Newton-Raphson)
- On Operators (monotone, strongly monotone, Lipschitz continuous, Co-coercive)
- Convex set, convex and strongly convex functions
- Properties of the gradient operator
- Gradient algorithm
- Consensus Optimization over networks
- Elements of Graph Theory
- Consensus algorithms
- Distributed Gradient Descent
- Distributed Gradient Tracking

Monotone operator

Consider a (finite-dimensional) *operator* $T : \mathbb{R}^n \to \mathbb{R}^n$ and let us introduce the following definitions

Definition. An operator $T : \mathbb{R}^n \to \mathbb{R}^n$ is *monotone* if for all x_A, x_B it holds

$$(T(x_A) - T(x_B))^\top (x_A - x_B) \geq 0$$

Remark. Informally, a monotone operator preserves the sign (in the scalar case at least) of its input increment
If $x_A - x_B$ is negative (positive), then $T(x_A) - T(x_B)$ stays negative (positive)

Strongly monotone operators

Definition. An operator $T : \mathbb{R}^n \to \mathbb{R}^n$ is *strongly monotone* if for all x_A, x_B it holds

$$(T(x_A) - T(x_B))^\top (x_A - x_B) \geq \mu \|x_A - x_B\|^2$$

for some $\mu > 0$

Remark. A μ-strongly monotone operator is also called μ-coercive.

Lipschitz continuous operators

Definition. An operator $T : \mathbb{R}^n \to \mathbb{R}^n$ is *Lipschitz continuous* if for all x_A, x_B it holds

$$\|T(x_A) - T(x_B)\| \leq L\|x_A - x_B\|$$

for some $L > 0$

Remark. For $L = 1$, the operator T is said to be *nonexpansive*, whereas for $L < 1$, the operator T is called a *contraction* (with contraction factor L)

Co-coercive operators

Definition. An operator $T : \mathbb{R}^n \to \mathbb{R}^n$ is *co-coercive* with factor $\frac{1}{L}$ if for all x_A, x_B it holds

$$(T(x_A) - T(x_B))^\top (x_A - x_B) \geq \frac{1}{L} \|T(x_A) - T(x_B)\|^2$$

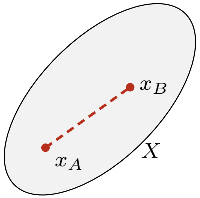
Outline

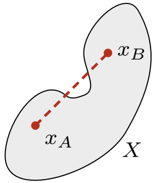
- Descent algorithms (Gradient/ Newton-Raphson)
- On Operators (monotone, strongly monotone, Lipschitz continuous, Co-coercive)
- Convex set, convex and strongly convex functions
- Properties of the gradient operator
- Gradient algorithm
- Consensus Optimization over networks
- Elements of Graph Theory
- Consensus algorithms
- Distributed Gradient Descent
- Distributed Gradient Tracking

Convexity

Definition. A set $X \subset \mathbb{R}^n$ is *convex* if for any two points $x_A, x_B \in X$ and for all $\theta \in [0,1]$, it holds

$$\theta x_A + (1-\theta)x_B \in X$$





Convex functions

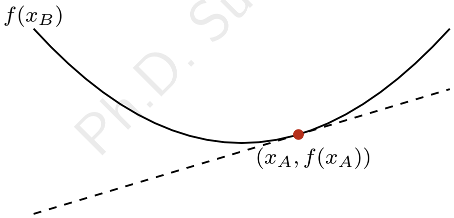
Definition. Let $X \subset \mathbb{R}^n$ be a convex set. A function $f : X \to \mathbb{R}$ is convex if for any two points $x_A, x_B \in X$ and for all $\theta \in [0, 1]$, it holds

$$f(\theta x_A + (1-\theta)x_B) \leq \theta f(x_A) + (1-\theta) f(x_B)$$

(also known as Jensen's inequality)

First order condition. If $f : X \to \mathbb{R}$ is convex and differentiable, then for any two points $x_A, x_B \in X$ it holds

$$f(x_B) \geq f(x_A) + \nabla f(x_A)^\top (x_B - x_A)$$



Convex functions

Definition. Let $X \subset \mathbb{R}^n$ be a convex set. A function $f : X \to \mathbb{R}$ is convex if for any two points $x_A, x_B \in X$ and for all $\theta \in [0,1]$, it holds

$$f(\theta x_A + (1-\theta)x_B) \leq \theta f(x_A) + (1-\theta) f(x_B)$$

(also known as Jensen's inequality)

First order condition. If $f : X \to \mathbb{R}$ is convex and differentiable, then for any two points $x_A, x_B \in X$ it holds

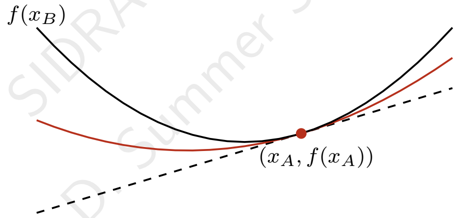
$$f(x_B) \geq f(x_A) + \nabla f(x_A)^\top (x_B - x_A)$$

Second order condition. For twice differentiable function, $\nabla^2 f(x) \geq 0$ for all x

Strongly convex smooth functions

Definition. A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is *strongly convex* with parameter $\mu > 0$ if for all $x_A, x_B \in \mathbb{R}^n$ it holds

$$f(x_B) \geq f(x_A) + \nabla f(x_A)^\top (x_B - x_A) + \frac{\mu}{2} \|x_B - x_A\|^2$$



Outline

Characterization of gradient operator

Definition. A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is *L-smooth* if its gradient is L-Lipschitz continuous, that is, if for all $x_A, x_B \in \mathbb{R}^n$, it holds

$$\|\nabla f(x_a) - \nabla f(x_B)\| \leq L\|x_B - x_A\|$$

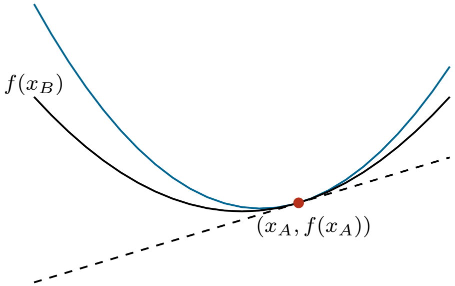
Characterization of gradient operator

Definition. A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is *L-smooth* if its gradient is L-Lipschitz continuous, that is, if for all $x_A, x_B \in \mathbb{R}^n$, it holds

$$\|\nabla f(x_a) - \nabla f(x_B)\| \leq L\|x_B - x_A\|$$

Proposition. A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ having a Lipschitz continuous gradient with parameter $L > 0$ (i.e., f is L-smooth) satisfies for all $x_A, x_B \in \mathbb{R}^d$ the inequality

$$f(x_B) \leq f(x_A) + \nabla f(x_A)^\top (x_B - x_A) + \frac{L}{2}\|x_B - x_A\|^2$$



Characterization of gradient operator

Proposition Let $f : \mathbb{R}^n \to \mathbb{R}$ be L-smooth and let

$$x^+ = x - \alpha \nabla f x,$$

for some $0 \leq \alpha \leq \frac{1}{L}$. Then,

$$f(x^+) \leq f(x) - \frac{\alpha}{2} \|\nabla f(x)\|^2$$

Characterizations of the gradient operator

Property A differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if and only if

$$(\nabla f(x_A) - \nabla f(x_B))^\top (x_A - x_B) \geq 0,$$

for all x_A, x_B, i.e., the gradient mapping $\nabla f : \mathbb{R}^n \to \mathbb{R}^n$ is a monotone mapping.

Property If f is convex, differentiable and L-smooth (i.e., ∇f is L- Lipschitz continuous) then ∇f is co-coercive, i.e., for all $x_A, x_B \in \mathbb{R}^n$ it holds

$$(\nabla f(x_A) - \nabla f(x_B))^\top (x_A - x_B) \geq \frac{1}{L} \|\nabla f(x_A) - \nabla f(x_B)\|^2$$

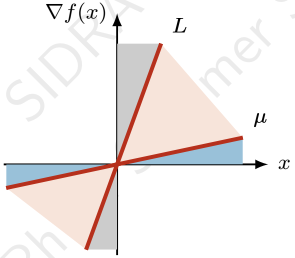
Property If f is μ strongly convex, differentiable and L-smooth (i.e., ∇f is L- Lipschitz continuous) then ∇f is co-coercive, i.e., for all $x_A, x_B \in \mathbb{R}^n$ it holds

$$(\nabla f(x_A) - \nabla f(x_B))^\top (x_A - x_B) \geq \frac{1}{\mu + L} \|\nabla f(x_A) - \nabla f(x_B)\|^2 + \frac{\mu L}{\mu + L} \|x_A - x_B\|^2$$

Characterizations of the gradient operator

It is equivalent to imposing upper and lower bounds on the gradient operator norm - sector bound

$$\mu\|x_A - x_B\| \leq \|\nabla f(x_A) - \nabla f(x_B)\| \leq L\|x_A - x_B\|$$



Properties of gradient operator

The cost function f is

- convex
- strongly convex
- convex and has Lipschitz continuous gradient
- strongly convex and has Lipschitz continuous gradient

The gradient operator ∇f is

- monotone
- strongly monotone
- co-coercive
- co-coercive with a slope-restricted factor

Outline

- Descent algorithms (Gradient/ Newton-Raphson)
- On Operators (monotone, strongly monotone, Lipschitz continuous, Co-coercive)
- Convex set, convex and strongly convex functions
- Properties of the gradient operator
- Gradient algorithm
- Consensus Optimization over networks
- Elements of Graph Theory
- Consensus algorithms
- Distributed Gradient Descent
- Distributed Gradient Tracking

Unconstrained optimization

Consider the unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} \; f(x)$$

with $f : \mathbb{R}^n \to \mathbb{R}$ having a L-Lipschitz continuous gradient

A minimum $x_\star \in \mathbb{R}^n$ of the problem must satisfy the (necessary) optimality condition given by

$$\nabla f(x_\star) = 0_n$$

Remark. If, additionally, f is convex, the optimality condition is also sufficient

Unconstrained convex optimization

From now on, consider the unconstrained optimization problem

$$\min_{x \in \mathbb{R}^n} \; f(x)$$

with $f : \mathbb{R}^n \to \mathbb{R}$ being *μ-strongly convex* and having a *L-Lipschitz continuous gradient* (it holds $L > \mu$)

Then, for all x, it holds

$$(\nabla f(x) - \nabla f(x_\star))^\top (x - x_\star) \geq \frac{1}{\mu+L} \|\nabla f(x) - \nabla f(x_\star)\|^2 + \frac{\mu L}{\mu+L} \|x - x_\star\|^2 \geq 0$$

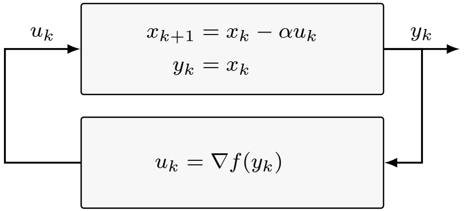
The gradient method

The gradient method is an iterative first-order optimization algorithm given by

$$\begin{aligned} x_{k+1} &= x_k - \alpha u_k \\ y_k &= x_k \\ u_k &= \nabla f(y_k) \end{aligned}$$

with $\alpha > 0$ being the so-called *stepsize*, while the initial condition $x_0 \in \mathbb{R}^n$ is arbitrary

The (unique) equilibrium $x_{\text{eq}} \in \mathbb{R}^n$ of the system is the (unique) minimum $x_\star$ of the optimization problem



Convergence result for the gradient method

Theorem. If f is strongly convex and has a Lipschitz continuous gradient, then the sequence of solution estimates $\{x_k\}_{k\in\mathbb{N}}$ generated by the gradient method with a sufficiently small, constant stepsize $\alpha > 0$ converges to the optimal solution $x_\star$ of the problem at a linear rate, i.e.,

$$\|x_k - x_\star\| \leq M\rho^k$$

with $\rho \in (0,1)$ and $M > 0$ depending on (μ, L) and $\|x^0 - x_\star\|$

The gradient method in error coordinates

Let $x_\star$ be the (unique) optimal solution/equilibrium and introduce the error coordinates

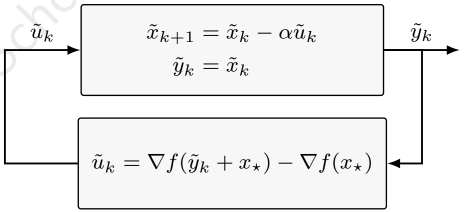
$$x \longmapsto \tilde{x} := x - x_\star$$

By shifting the input and the output as

$$u \longmapsto \tilde{u} := u - \nabla f(x_\star)$$
$$y \longmapsto \tilde{y} := y - x_\star$$

the resulting error dynamics is

$$\begin{aligned}\tilde{x}_{k+1} &= \tilde{x}_k - \alpha \tilde{u}_k \\ \tilde{y}_k &= \tilde{x}_k \\ \tilde{u}_k &= \nabla f(\tilde{y}_k + x_\star) - \nabla f(x_\star)\end{aligned}$$



$\tilde{u}_k$
$\tilde{x}_{k+1} = \tilde{x}_k - \alpha \tilde{u}_k$
$\tilde{y}_k = \tilde{x}_k$
$\tilde{y}_k$
$\tilde{u}_k = \nabla f(\tilde{y}_k + x_\star) - \nabla f(x_\star)$

The convergence analysis amounts to studying the stability properties of the origin $\tilde{x} = 0_n$

Remark. With these symbols, we can write $\tilde{u}_k^\top \tilde{y}_k \geq \frac{1}{\mu+L}\|\tilde{u}_k\|^2 + \frac{\mu L}{\mu+L}\|\tilde{y}_k\|^2$

The gradient method in error coordinates

Let $x_\star$ be the (unique) optimal solution/equilibrium and introduce the error coordinates

$$x \longmapsto \tilde{x} := x - x_\star$$

By shifting the input and the output as

$$u \longmapsto \tilde{u} := u - \nabla f(x_\star)$$
$$y \longmapsto \tilde{y} := y - x_\star$$

the resulting error dynamics is

$$\begin{aligned}\tilde{x}_{k+1} &= \tilde{x}_k - \alpha \tilde{u}_k\\ \tilde{y}_k &= \tilde{x}_k\\ \tilde{u}_k &= \nabla f(\tilde{y}_k + x_\star) - \nabla f(x_\star)\end{aligned}$$

Remember: If f is μ- strongly convex and has L-Lipschitz continuous gradient then

$$\begin{aligned}&(\nabla f(x) - \nabla f(x_\star))^\top (x - x_\star) \geq\\ &\qquad \tfrac{1}{\mu+L}\|\nabla f(x) - \nabla f(x_\star)\|^2 + \tfrac{\mu L}{\mu+L}\|x - x_\star\|^2 \geq 0\end{aligned}$$

The convergence analysis amounts to studying the stability properties of the origin $\tilde{x} = 0_n$

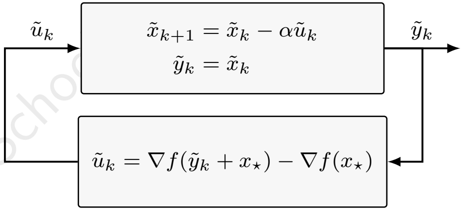
Remark. With these symbols, we can write $\tilde{u}_k^\top \tilde{y}_k \geq \frac{1}{\mu+L}\|\tilde{u}_k\|^2 + \frac{\mu L}{\mu+L}\|\tilde{y}_k\|^2$

Convergence proof

Consider a Lyapunov function $V(\tilde{x}) = \|\tilde{x}\|^2$, then its increment along trajectories of the gradient method satisfies

$$\begin{aligned} V(\tilde{x}_{k+1}) - V(\tilde{x}_k) &= \|\tilde{x}_{k+1}\|^2 - \|\tilde{x}_k\|^2 \\ &= -2\alpha(\tilde{u}_k)^\top \tilde{x}_k + \alpha^2\|\tilde{u}_k\|^2 \\ &\leq -2\alpha\gamma_1\|\tilde{x}_k\|^2 + \alpha(\alpha - 2\gamma_2)\|\tilde{u}_k\|^2 \end{aligned}$$

with $\gamma_1 := \frac{\mu L}{\mu + L}$ and $\gamma_2 := \frac{1}{\mu+L}$



$\tilde{u}_k$
$\tilde{x}_{k+1} = \tilde{x}_k - \alpha\tilde{u}_k$
$\tilde{y}_k = \tilde{x}_k$
$\tilde{y}_k$
$\tilde{u}_k = \nabla f(\tilde{y}_k + x_\star) - \nabla f(x_\star)$

For a small enough stepsize α (i.e., $\alpha \leq 2\gamma_2$), we can write

$$V(\tilde{x}_{k+1}) - V(\tilde{x}_k) \leq -2\alpha\gamma_1 V(\tilde{x}_k) \quad \Longrightarrow \quad \begin{aligned} \|\tilde{x}_{k+1}\|^2 &\leq (1 - 2\alpha\gamma_1)\|\tilde{x}_k\|^2 \\ &\leq (1-2\alpha\gamma_1)^k\|\tilde{x}_0\|^2 \end{aligned}$$

Therefore $\{\tilde{x}_k\}_{k\in\mathbb{N}}$ goes exponentially/geometrically fast to zero

Remark. Imposing $\alpha \leq 2\gamma_2$ implies that $(1 - 2\alpha\gamma_1) \in (0, 1)$

Explicit convergence rate

The *convergence rate* corresponding to the largest feasible stepsize α, namely for

$$\alpha = 2\gamma_2 = \tfrac{2}{\mu+L}$$

is given by

$$1 - 2\alpha\gamma_1 = 1 - \tfrac{4}{\mu+L}\tfrac{\mu L}{\mu+L} = \left(\tfrac{\mu-L}{\mu+L}\right)^2$$

Therefore, we can write

$$\|\tilde{x}_k\| \leq \left(\tfrac{L-\mu}{L+\mu}\right)^{2k}\|\tilde{x}_0\|$$

Outline

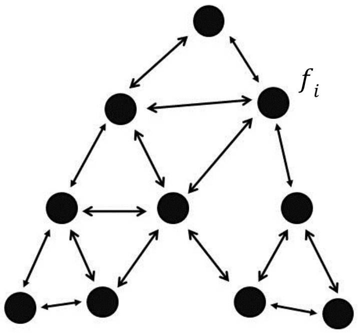
- Descent algorithms (Gradient/ Newton-Raphson)
- On Operators (monotone, strongly monotone, Lipschitz continuous, Co-coercive)
- Convex set, convex and strongly convex functions
- Properties of the gradient operator
- Gradient algorithm
- Consensus Optimization over networks
- Elements of Graph Theory
- Consensus algorithms
- Distributed Gradient Descent
- Distributed Gradient Tracking

Optimization over networks

- $\mathcal{G} = (V, \mathcal{E})$ undirected
- $f_i : \mathbb{R} \to \mathbb{R}$, local function known only by node i

$$\min_x \sum_{i=1}^{N} f_i(x)$$

Goal : to design distributed and scalable algorithms



From optimization over networks to consensus optimization

- $\mathcal{G} = (V, \mathcal{E})$ undirected
- $f_i : \mathbb{R} \to \mathbb{R}$, local function known only by node i

$$\min_{x} \sum_{i=1}^{N} f_i(x)$$

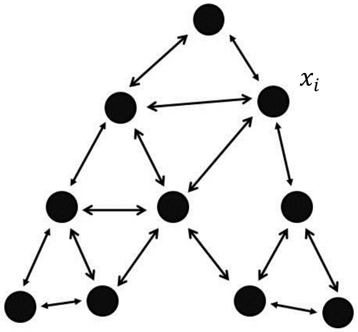
- x_i : *local copy* of x stored in memory by node i

$$\min_{x_1,\dots,x_N} \sum_{i=1}^{N} f_i(x_i)$$

$$\text{s.t. } x_1 = \ldots = x_N$$

consensus constraint

- The two problems are equivalent



From optimization over networks to consensus optimization

- $\mathcal{G} = (V, \mathcal{E})$ undirected
- $f_i : \mathbb{R} \to \mathbb{R}$, local function known only by node i

$$\min_{x} \sum_{i=1}^{N} f_i(x)$$

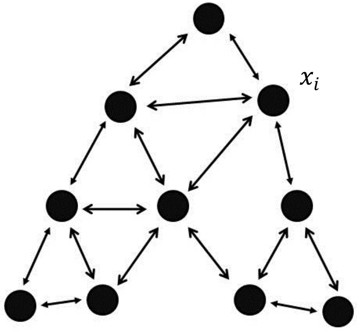
- x_i : *local copy* of x stored in memory by node i

$$\min_{x_1,\ldots,x_N} \sum_{i=1}^{N} f_i(x_i)$$

$$\text{s.t. } x_i = x_j \qquad \forall\ (i,j) \in \mathcal{E}$$

consensus constraint

- The two problems are equivalent if the graph $\mathcal{G}$ is *connected*



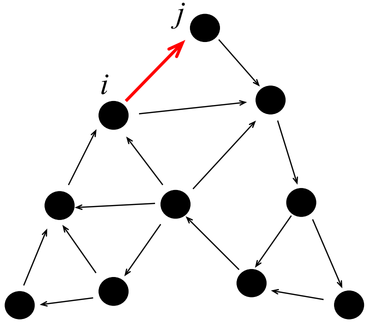
Outline

- Descent algorithms (Gradient/ Newton-Raphson)
- On Operators (monotone, strongly monotone, Lipschitz continuous, Co-coercive)
- Convex set, convex and strongly convex functions
- Properties of the gradient operator
- Gradient algorithm
- Consensus Optimization over networks
- Elements of Graph Theory
- Consensus algorithms
- Distributed Gradient Descent
- Distributed Gradient Tracking

Elements of Graph Theory

Directed Graph $\mathcal{G} = (V, \mathcal{E})$

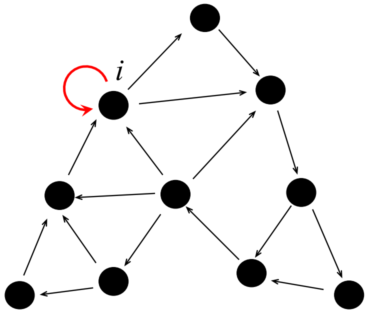
- V : set of nodes
  $V = \{1, 2, \ldots, N\}$
- $\mathcal{E} \subseteq V \times V$: set of edges
  (i, j) : edge getting out from the node i and getting in the node j
  Node i can send information to node j



Elements of Graph Theory

Directed Graph $\mathcal{G} = (V, \mathcal{E})$

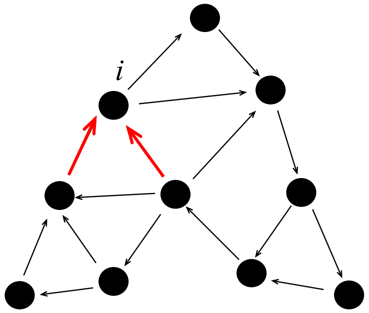
- V : set of nodes
  $V = \{1, 2, \ldots, N\}$
- $\mathcal{E} \subseteq V \times V$: set of edges
  (i, j) : edge getting out from the node i and getting in the node j
  Node i can send information to node j
- (i, i) self-loop
  We typically assume that the self-loops are present though not drawn



Elements of Graph Theory

Directed Graph $\mathcal{G} = (V, \mathcal{E})$

- V : set of nodes
  $V = \{1, 2, \dots, N\}$
- $\mathcal{E} \subseteq V \times V$: set of nodes
  (i, j) : edge getting out from the node i and getting in the node j
  Node i can send information to node j
- (i, i) self-loop
  We typically assume that the self-loops are present though not drawn

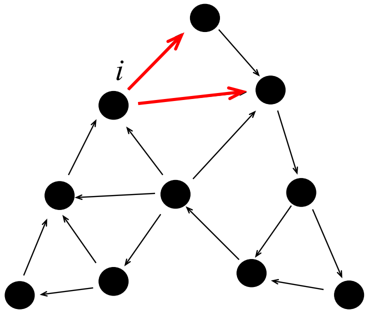


$\mathcal{N}_{\text{in}}^{i} = \{j | (j, i) \in \mathcal{E}\}$ **in - neighbors**
(*nodes transmitting information to node i*)

Elements of Graph Theory

Directed Graph $\mathcal{G} = (V, \mathcal{E})$

- V : set of nodes
  $V = \{1, 2, \dots, N\}$
- $\mathcal{E} \subseteq V \times V$: set of nodes
  (i, j) : edge getting out from the node i and getting in the node j
  Node i can send information to node j
- (i, i) self-loop
  We typically assume that the self-loops are present though not drawn



$\mathcal{N}^i_{\text{in}} = \{j | (j, i) \in \mathcal{E}\}$ **in - neighbors**
(nodes transmitting information to node i)

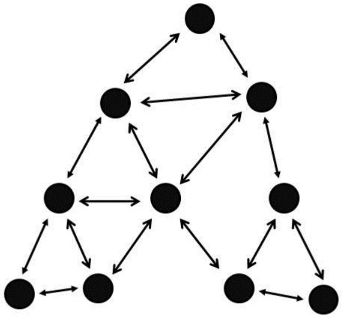
$\mathcal{N}^i_{\text{out}} = \{j | (i, j) \in \mathcal{E}\}$ **out - neighbors**
(nodes receiveing information from node i)

Elements of Graph Theory

Undirected Graph $\mathcal{G} = (V, \mathcal{E})$: if $(i,j) \in \mathcal{E}$ then also $(j,i) \in \mathcal{E}$

Hence $\mathcal{N}^i_{\text{in}} = \mathcal{N}^i_{\text{out}} = \mathcal{N}_i$

Degree of node i : $d_i = |\mathcal{N}_i|$



Elements of Graph Theory

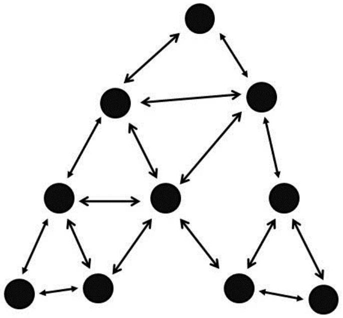
Undirected Graph $\mathcal{G} = (V, \mathcal{E})$: if $(i,j) \in \mathcal{E}$ then also $(j,i) \in \mathcal{E}$

Hence $\mathcal{N}^i_{\text{in}} = \mathcal{N}^i_{\text{out}} = \mathcal{N}_i$

Degree of node i : $d_i = |\mathcal{N}_i|$

Adjacency matrix A

$$[A]_{ij} = \begin{cases} 1 & \text{if} \quad (i,j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases}$$



Elements of Graph Theory

Undirected Graph $\mathcal{G} = (V, \mathcal{E})$: if $(i,j) \in \mathcal{E}$ then also $(j,i) \in \mathcal{E}$

Hence $\mathcal{N}^i_{\text{in}} = \mathcal{N}^i_{\text{out}} = \mathcal{N}_i$

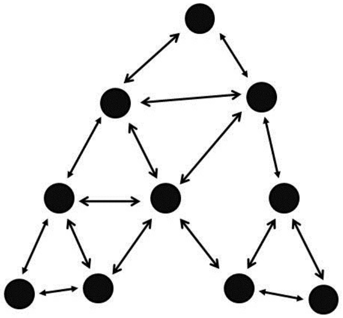
Degree of node i : $d_i = |\mathcal{N}_i|$

Adjacency matrix A

$$[A]_{ij} = \begin{cases} 1 & \text{if} \quad (i,j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases}$$

Degree matrix D

$$D = \text{diag}\{d_i\}$$



Elements of Graph Theory

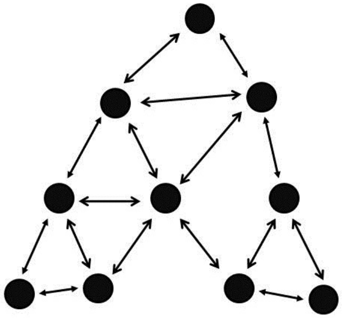
Undirected Graph $\mathcal{G} = (V, \mathcal{E})$: if $(i, j) \in \mathcal{E}$ then also $(j$

Hence $\mathcal{N}^i_{\text{in}} = \mathcal{N}^i_{\text{out}} = \mathcal{N}_i$

Degree of node i : $d_i = |\mathcal{N}_i|$

Adjacency matrix A

$$[A]_{ij} = \begin{cases} 1 & \text{if} \quad (i, j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases}$$



Degree matrix D

$$D = \text{diag}\{d_i\}$$

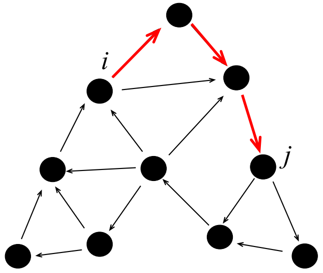
Laplacian matrix L

$$L = D - A$$

Elements of Graph Theory

Definition. A *directed graph* is said to be *strongly connected* if, given any pair of vertices i and j, i is connected with j

that is, there exists a direct path connecting i to j



Definition. A *undirected graph* is said to be *connected* if, given any pair of vertices i and j, i is connected with j

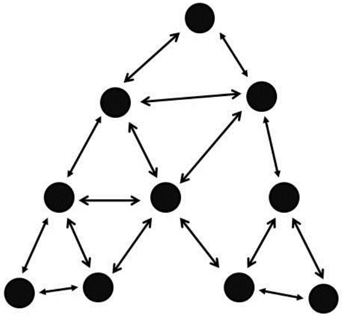
that is, there exists a undirect path connecting i to j

Recursive distributed algorithms consistent with a graph

Undirected Graph $\mathcal{G} = (V, \mathcal{E})$

Definition. A *recursive distributed algorithm*is said to be consistent with the graph $\mathcal{G}$ if the i-th node's update law depends only on the local variables of i and its neighbors, i.e.,

$$x_{i,k+1} = g_i\left(x_{i,k}, \{x_{j,k}\}_{j\in\mathcal{N}_i}, k\right)$$



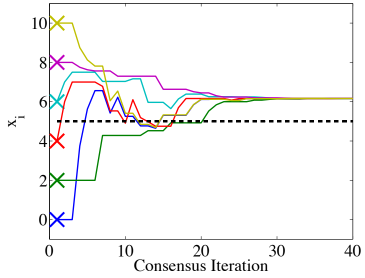
Outline

Consensus Problem

Definition. A *recursive distributed algorithm* consistent with the graph $\mathcal{G} = (V, \mathcal{E})$ is said to *asymptotically achieve consensus* if

$$x_{i,k} \to \alpha$$

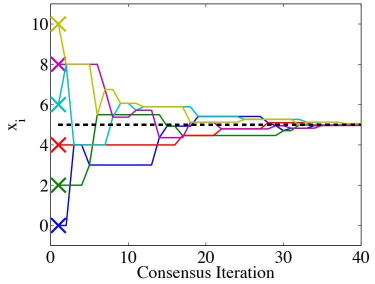
for all $i \in V$, for some $\alpha \in \mathbb{R}$.



Definition. A *recursive distributed algorithm* consistent with the graph $\mathcal{G} = (V, \mathcal{E})$ is said to *asymptotically achieve average consensus* if

$$x_{i,k} \to \frac{1}{N}\sum_{j=1}^{N} x_{j,0}$$

for all $i \in V$.



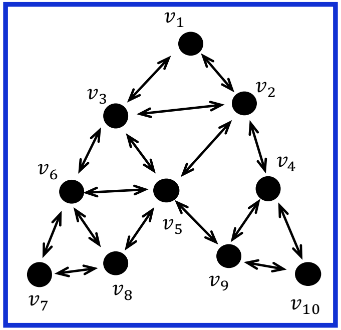
Average Consensus Problem : formulation

- $\mathcal{G} = (V, \mathcal{E})$ undirected
- State of node i is initialized with value v_i, i.e.,

$$x_{i,0} = v_i$$

- **Goal** : to compute the average of initial values, i.e

$$\frac{1}{N}\sum_{i=1}^{N} v_i$$



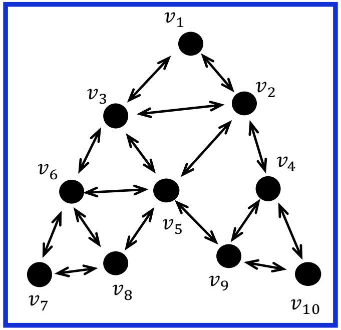
Consensus Algorithm

Algorithm:

$$x_{i,0} = v_i$$
$$x_{i,k+1} = \sum_{j \in \mathcal{N}_i} w_{ij}\, x_{j,k}$$

where

- $\sum_{j \in \mathcal{N}_i} w_{ij} = 1,\ w_{ij} \geq 0$ (*convex combination*);
- $\mathcal{N}_i = \{j | (i,j) \in \mathcal{E}\}$ (*distributed algorithm*)



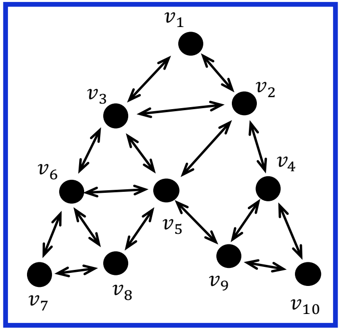
Consensus Algorithm

Algorithm:

$$x_{i,0} = v_i$$
$$x_{i,k+1} = \sum_{j \in \mathcal{N}_i} w_{ij}\, x_{j,k}$$

where

- $\sum_{j \in \mathcal{N}_i} w_{ij} = 1,\ w_{ij} \geq 0$ (*convex combination*);
- $\mathcal{N}_i = \{j | (i,j) \in \mathcal{E}\}$ (*distributed algorithm*)



Observations:

- $w_{ij} \neq 0$ only if $(i,j) \in \mathcal{E}$;
- If $(i,j) \notin \mathcal{E}$ then $w_{ij} = 0$.

As a consequence $\sum_{j \in \mathcal{N}_i} w_{ij} = \sum_{j=1}^{N} w_{ij} = 1$

Consensus Algorithm

Algorithm:

$$x_{i,0} = v_i$$
$$x_{i,k+1} = \sum_{j \in \mathcal{N}_i} w_{ij}\, x_{j,k}$$

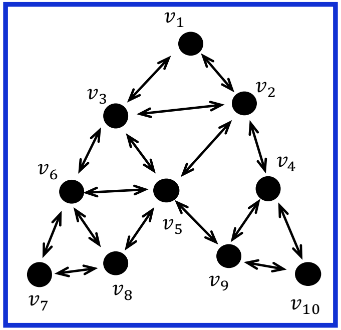
Vector form

Let $x = [x_1, \ldots, x_N]^\top$ and $v = [v_1, \ldots, v_N]^\top$ then

$$x_{k+1} = W x_k, \qquad x_0 = v$$

where W is row stochastic (*nonnegative matrix with sum of elements along each row equal to* 1)

- $\sum_{j=1}^{N} w_{ij} = 1$;
- $w_{ij} \geq 0$.



Consensus Algorithm

Algorithm:

$$x_{i,0} = v_i$$
$$x_{i,k+1} = \sum_{j \in \mathcal{N}_j} w_{ij}\, x_{j,k}$$

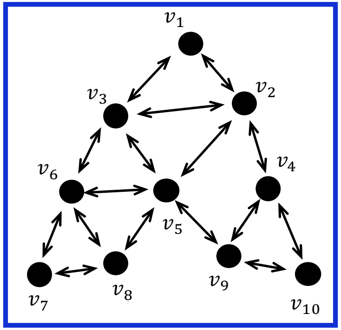
Vector form

Let $x = [x_1, \ldots, x_N]^\top$ and $v = [v_1, \ldots, v_N]^\top$ then

$$x_{k+1} = W x_k, \qquad x_0 = v$$

where W is row stochastic (*nonnegative matrix with sum of elements along each row equal to* 1)

$$W\mathbf{1} = \mathbf{1}, \qquad \mathbf{1} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix}$$



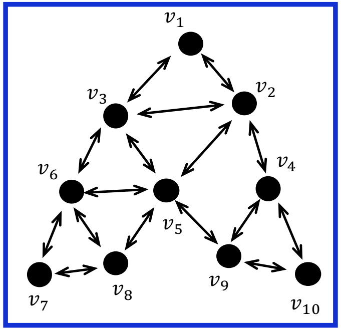
Average Consensus Algorithm

Algorithm:

$$x_{k+1} = W x_k, \qquad x_0 = v$$

where W is **doubly stochastic**, that is,

- $W\mathbf{1} = \mathbf{1}$ (*row stochastic*)
- $\mathbf{1}^\top W = \mathbf{1}^\top$ (*column stochastic*)



Properties

- Column stochastic = mass preservation

$$\mathbf{1}^\top x_{k+1} = \mathbf{1}^\top W x_k = \mathbf{1}^\top x_k = \ldots = x_0$$

- Consensus ($\lim_{k\to\infty} x_k = \alpha\mathbf{1}$) + mass preservation = average consensus ($\alpha = \frac{1}{N}\sum_{i=1}^N v_i$)

Indeed, $\mathbf{1}^\top x(\infty) = N\alpha = \mathbf{1}^\top x_0 = \sum_{i=1}^N v_i \to \alpha = \frac{1}{N}\sum_{i=1}^N v_i$

Consensus conditions

Algorithm:

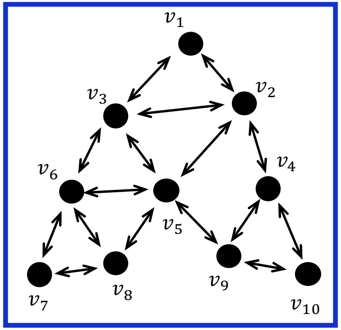
$$x_{k+1} = W x_k, \qquad x_0 = v$$

where W is *row stochastic* .

Question : When is consensus achieved?

Answer : When W is *primitive*

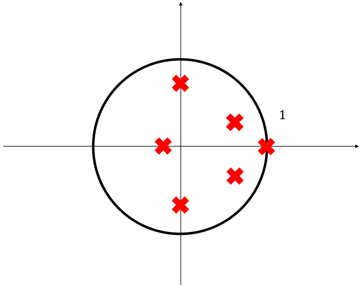
Remark. If W *doubly stochastic* + *primitive* then
average consensus.



Consensus conditions

When a row stochastic matrix W is primitive?

- eigenvalue 1 ($W\mathbf{1} = \mathbf{1}$) is simple.
- all the other eigenvalues are strictly inside the unitary circle.



Graph conditions for average consensus

When a doubly stochastic matrix W is primitive?

- eigenvalue 1 ($W\mathbf{1} = \mathbf{1}$) is simple.
- all the other eigenvalues are strictly inside the unitary circle.

Graph-based conditions for average consensus?

Given a matrix W we can associate a graph $\mathcal{G}_W(\mathcal{V}, \mathcal{E}_W)$ such that

$$\textit{if } w_{ij} \neq 0 \textit{ then } (j,i) \in \mathcal{E}_W \ \textit{(otherwise } (j,i) \notin \mathcal{E}_W \textit{)}$$

Proposition. Let W be a doubly stochastic matrix. If the following two conditions

- $w_{ii} \neq 0$ for all $i \in \mathcal{V}$;
- $\mathcal{E}_W$ is strongly connected.

are satisfied then W is primitive and, hence, average consensus is achieved.

General consensus (not just average)

What about if W is row stochastic, primitive, but not column stochastic?
We have consensus, that is,

$$x_i(t) \to \alpha \qquad \forall\ i, \qquad \textit{component-wise}$$
$$x(t) \to \alpha \mathbf{1} \qquad \textit{vector-form}$$

but, in general,

$$\alpha \neq \frac{1}{N} \sum_{i=1}^{N} v_i$$

Since W is not column stochastic ($\mathbf{1}^\top W \neq \mathbf{1}^\top$), mass is not preserved

$$\mathbf{1}^T x_{k+1} \neq \mathbf{1}^\top x_k, \qquad \mathbf{1}^\top x_k \neq \mathbf{1}^\top x_0$$

General consensus (not just average)

What about if W is row stochastic, primitive, but not column stochastic?

- eigenvalue 1 $(W\mathbf{1} = \mathbf{1})$ is simple.
- all the other eigenvalues are strictly inside the unitary circle.

Let $\lambda_i, i = 1, \ldots, N$ be the i-th eigenvalue and let

- $v^{(i)}$ be the corresponding right eigenvector;
- $w^{(i)}$ be the corresponding left eigenvector.
- $\lambda_1 = 1$, $v^{(1)} = \mathbf{1}$, $w^{(1)} \geq 0$ — we assume $\sum_{i=1}^N w_i^{(1)} = 1$.

$$W = \sum_{i=1}^N \lambda_i v^{(i)} {w^{(i)}}^\top \Rightarrow W^k = \sum_{i=1}^N \lambda_i^k v^{(i)} {w^{(i)}}^\top \Rightarrow W^k \mapsto \mathbf{1}{w^{(1)}}^\top$$

Hence

$$x(\infty) = \lim_{k\to\infty} W^k x_0 = \mathbf{1}{w^{(1)}}^\top x_0 = \left({w^{(1)}}^\top x_0\right)\mathbf{1} = \left(\sum_{i=1}^N w_i^{(1)} x_{i,0}\right)\mathbf{1}.$$

Going back to average consensus

W is doubly stochastic

$$\mathbf{1}^T W = \mathbf{1}^T \Rightarrow w^{(1)} = \frac{1}{N}\mathbf{1} \Rightarrow W^k \mapsto \frac{1}{N}\mathbf{1}\mathbf{1}^T$$

and, hence,

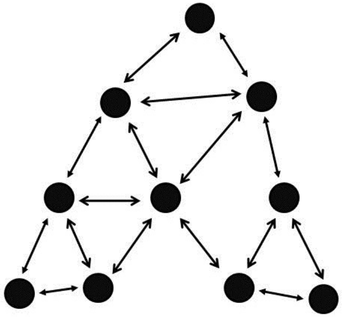
$$x(\infty) = \lim_{k\to\infty} W^k x_0 = \frac{1}{N}\mathbf{1}\mathbf{1}^\top x_0 = \left({w^{(1)}}^\top x_0\right)\mathbf{1} = \frac{1}{N}\left(\sum_{i=1}^N x_{i,0}\right)\mathbf{1}.$$

How to build doubly stochastic matrices?

Undirected Graph $\mathcal{G} = (V, \mathcal{E})$

$\mathcal{N}_i = \{j \in V \,:\, (i,j) \in \mathcal{E}\}$

Degree of node i : $d_i = |\mathcal{N}_i|$



Maximum degree weight

Let $d \geq \max_i d_i$

$$w_{ij} = \begin{cases} \frac{1}{d+1} & \text{if } (i,j) \in \mathcal{E} \text{ and } i \neq j \\ 0 & \text{if } (i,j) \notin \mathcal{E} \\ 1 - \frac{d_i}{d+1} & \text{if } i = j \end{cases}$$

How to build doubly stochastic matrices?

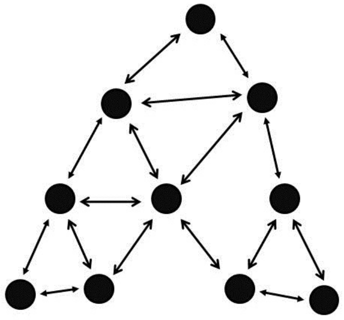
Undirected Graph $\mathcal{G} = (V, \mathcal{E})$

$$\mathcal{N}_i = \{j \in V \; : \; (i,j) \in \mathcal{E}\}$$

Degree of node i : $d_i = |\mathcal{N}_i|$

Metropolis weights

$$w_{ij} = \begin{cases} \frac{1}{1+\max\left\{d_i, d_j\right\}} & \text{if } (i,j) \in \mathcal{E} \text{ and } i \neq j \\ 0 & \text{if } (i,j) \notin \mathcal{E} \\ 1 - \sum_{k=1, k\neq i}^{N} w_{ik} & \text{if } i = j \end{cases}$$



How to build doubly stochastic matrices?

Undirected Graph $\mathcal{G} = (V, \mathcal{E})$

$\mathcal{N}_i = \{j \in V \,:\, (i,j) \in \mathcal{E}\}$

Degree of node i : $d_i = |\mathcal{N}_i|$

Laplacian - based method

Let $\epsilon > 0$ be such that

$$\epsilon < \frac{1}{\max_i d_i}$$

Define

$$W = I - \epsilon L$$

$\Longrightarrow$ W is doubly stochastic, and, if $\mathcal{G}$ is connected, primitive.

To remember...

Adjacency matrix A

$$[A]_{ij} = \begin{cases} 1 & \text{if} \quad (i,j) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases}$$

Degree matrix D

$$D = \text{diag}\{d_i\}$$

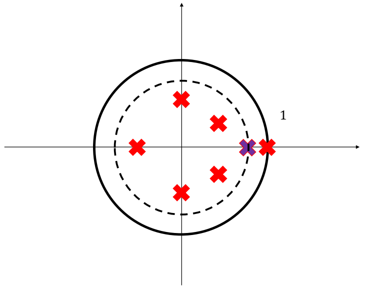
Laplacian matrix L

$$L = D - A$$

Rate of convergence?

W is row stochastic, primitive.

- eigenvalue 1 $(W\mathbf{1} = \mathbf{1})$ is **simple**.
- all the other eigenvalues are strictly inside the unitary circle.



ρ_{ess} : essential spectral radius

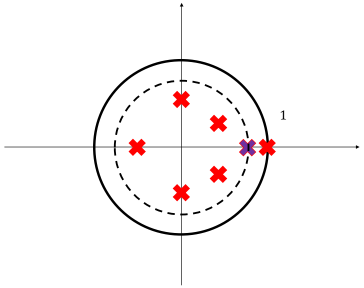
ρ_{ess} : norm of the largest eigenvalue in modulus different from 1

$\rho_{\text{ess}} = \max\{|\lambda| \,:\, \lambda$ eigenvalue of W, $\lambda \neq 1\}$

Rate of convergence?

W is row stochastic, primitive.

- eigenvalue 1 ($W\mathbf{1} = \mathbf{1}$) is **simple**.
- all the other eigenvalues are strictly inside the unitary circle.



ρ_{ess} : essential spectral radius

ρ_{ess} : norm of the largest eigenvalue in modulus different from 1

$\rho_{\text{ess}} = \max\{|\lambda| \,:\, \lambda \text{ eigenvalue of } W,\ \lambda \neq 1\}$

We have that

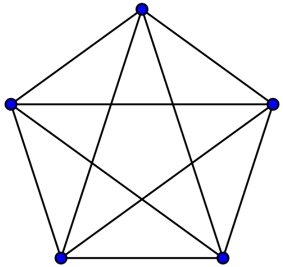
$$\|x_k - \alpha\mathbf{1}\| \leq C\rho_{\text{ess}}^k, \qquad \alpha \text{ consensus value}$$

Rate of convergence : examples

$\rho_{\text{ess}} = \{|\lambda| : \lambda$ eigenvalue of W, $\lambda \neq 1\}$

Complete graph : $\rho_{\text{ess}} = 0$

$$W = \frac{1}{N}\mathbf{1}\mathbf{1}^T = \begin{bmatrix} 1/N & \cdots & 1/N \\ \vdots & & \vdots \\ 1/N & \cdots & 1/N \end{bmatrix} := J$$



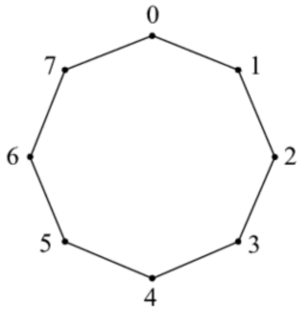
Average Consensus is reached in one step *(dead-beat)*

Rate of convergence : examples

$\rho_{\text{ess}} = \{|\lambda| : \lambda$ eigenvalue of W, $\lambda \neq 1\}$

Circle graph : $\rho_{\text{ess}} = 1 - \frac{C}{N^2}$

$$W = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & 0 & 0 & \cdots & 0 & \frac{1}{3} \\ \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & 0 & \cdots & 0 & 0 \\ 0 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} & \cdots & 0 & 0 \\ \vdots & & & & & & \vdots \\ \frac{1}{3} & 9 & & \cdots & 0 & \frac{1}{3} & \frac{1}{3} \end{bmatrix}$$

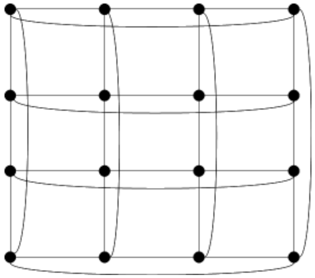


$\lim_{N \to \infty} \rho_{\text{ess}} = 1 \quad \rightarrow \quad$ *the greater the number of agents the slower the algorithm*

Rate of convergence : examples

$\rho_{\text{ess}} = \{|\lambda| : \lambda$ eigenvalue of W, $\lambda \neq 1\}$

2D Toreus graph : $\rho_{\text{ess}} = 1 - \frac{C}{N}$



Again

$$\lim_{N\to\infty} \rho_{\text{ess}} = 1$$

the greater the number of agents the slower the algorithm

Observations.

- A bit better than circle graph
- Similar behavior for 3D toreus and d-dimensional toruses

Rate of convergence : examples

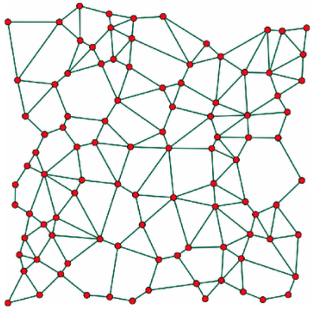
$\rho_{\text{ess}} = \max\{|\lambda| : \lambda \text{ eigenvalue of } W,\ \lambda \neq 1\}$

Random geometric graph.

- Place N nodes within a square of side L
- Connect two nodes if their distance is smaller than R

Behavior similar to that of 2-dimensional toruses

$$\rho_{\text{ess}} \approx 1 - \frac{C}{N}$$

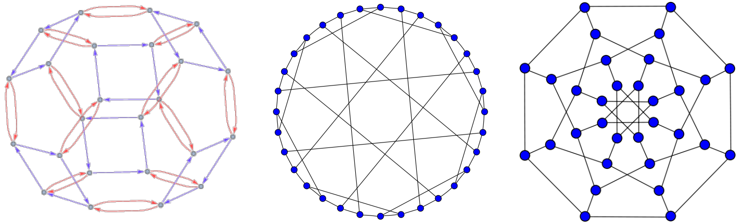


Rate of convergence : examples

$\rho_{\text{ess}} = \{|\lambda| \,:\, \lambda$ eigenvalue of W, $\lambda \neq 1\}$

Cayley graphs : graphs with particular symmetries (e.g., toruses) where each node has the same number of neighbors, say ν

Cayley graphs : $\rho_{\text{ess}} \geq 1 - \frac{C}{N^{\frac{2}{\nu}}}$



Rate of convergence : examples

$\rho_{\text{ess}} = \{|\lambda| \, : \, \lambda$ eigenvalue of W, $\lambda \neq 1\}$

Cayley graphs : graphs with particular symmetries (e.g., toruses) where each node has the same number of neighbors, say ν

Cayley graphs : $\rho_{\text{ess}} \geq 1 - \frac{C}{N^{\frac{2}{\nu}}}$

Questions:

- Is it the symmetry-structure on the graph that prevents achieving good performance?
- Or, is it the fact that each node communicates with a limited number of neighbors?

Rate of convergence : regular graphs

Regular graphs : graphs where each node is connected to the same number of neighbors

- Consider the set of connected regular graphs with degree ν;
- Build the set of corresponding primitive doubly stochastic matrices (Metropolis weights)

$$\Longrightarrow \quad \mathbb{E}\left[\rho_{\text{ess}}(W)\right] \approx \frac{2\sqrt{\nu-1}}{\nu}$$

Remark. As a consequence, we have that, if we fix ν, in the average, ρ_{ess} will stay bounded away from 1, as $N \to \infty$

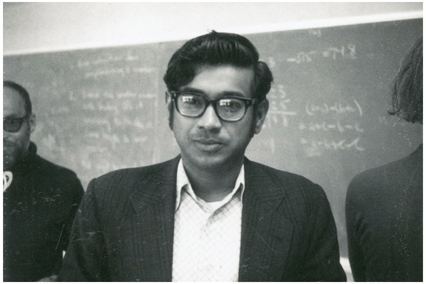
Ramanujan graphs

Ramanujan graphs are those graphs for which we have exactly

$$\rho_{\mathrm{ess}}(W) = \frac{2\sqrt{\nu - 1}}{\nu}$$

There are plenty of Ramanujan graphs but it is not still clear if for any pair (N, ν) there exists a Ramanujan graph with N vertices and degree ν

Srinivasa Ramanujan (1887-1920) was an Indian mathematician particularly known for his contributions number theory.



Adding a slot of memory

Let W be a primitive symmetric doubly stochastic matrix and let

$$x_{t+1} = W x_k,$$

be the corresponding consensus algorithm.

Assume W is built over a family of connected graphs of increasing size such that

$$\rho_{ess}(W_N) = 1 - f(N) \qquad \text{where} \qquad \lim_{N\to\infty} f(N) \to 0$$

Second-order consensus algorithm

$$x_{k+1} = \alpha W x_k + (1-\alpha) x_{k-1} \qquad 1 < \alpha < 2, \qquad \textit{vector form}$$

$$x_{i,k+1} = \alpha \sum_{j=1}^{N} w_{ij} x_{j,k} + (1-\alpha) x_{i,k-1} \qquad \textit{component-wise}$$

If $x_{-1} = 0$ then *mass is preserved*

$$\implies \mathbf{1}^T x_{k+1} = \mathbf{1}^T x_k$$

Adding a slot of memory

Second-order consensus algorithm

$$\begin{bmatrix} x_{k+1} \\ x_k \end{bmatrix} = \begin{bmatrix} \alpha W & (1-\alpha) I \\ I & 0 \end{bmatrix} \begin{bmatrix} x_k \\ x_{k-1} \end{bmatrix}$$

Proposition Given W symmetric, primitive, doubly stochastic, with

$$\rho_{\text{ess}}(W) = 1 - f(N),$$

there exists $\alpha(W)$, $1 < \alpha < 2$, such that the convergence rate of the augmented scheme is

$$\rho_{\text{ess, aug}}(W) = 1 - \sqrt{f(N)}.$$

Observation: Notice that $\sqrt{f(N)} > f(N)$ when $f(N) < 1$.

Adding a slot of memory

Second-order consensus algorithm

$$\begin{bmatrix} x_{k+1} \\ x_k \end{bmatrix} = \begin{bmatrix} \alpha W & (1-\alpha)I \\ I & 0 \end{bmatrix} \begin{bmatrix} x_k \\ x_{k-1} \end{bmatrix}, \qquad 1 < \alpha < 2$$

For future use...

$$\begin{bmatrix} x_{k+1} \\ x_k \end{bmatrix} = \begin{bmatrix} (1+\beta)W & -\beta I \\ I & 0 \end{bmatrix} \begin{bmatrix} x_k \\ x_{k-1} \end{bmatrix}, \qquad 0 < \beta < 1$$

Dynamic average consensus

Consider the set of time-varying signals $\{r_{i,k}\}_{i=1}^{N}$

- signal $r_{i,k}$ is observed by node i;
- $x_{i,k}$ is the internal state of node i;

Goal : to track the time-varying average $\bar{r}_k = \frac{1}{N}\sum_{i=1}^{N} r_{i,k}$, that is,

$$x_{i,k} \longrightarrow \bar{r}_k$$

Algorithm : W doubly stochastic matrix

$$x_{i,k+1} = \sum_{j=1}^{N} w_{ij} x_{j,k} + r_{i,k+1} - r_{i,k}$$

$$x_{k+1} = W x(k) + r_{k+1} - r_k, \qquad r_k = \begin{bmatrix} r_{1,k} & r_{2,k} & \dots & r_{N,k} \end{bmatrix}^T$$

Initialization : $x_0 = r_0 \qquad \Longrightarrow \qquad$ **Mass Preservation** : $\mathbf{1}^T x_k = \mathbf{1}^T r_k$

Outline

- Descent algorithms (Gradient/ Newton-Raphson)
- On Operators (monotone, strongly monotone, Lipschitz continuous, Co-coercive)
- Convex set, convex and strongly convex functions
- Properties of the gradient operator
- Gradient algorithm
- Consensus Optimization over networks
- Elements of Graph Theory
- Consensus algorithms
- Distributed Gradient Descent
- Distributed Gradient Tracking

Distributed gradient descent

- $\mathcal{G} = (V, \mathcal{E})$ undirected
- $f_i : \mathbb{R} \to \mathbb{R}$, local function known only by node i

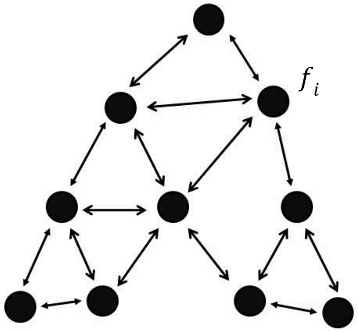
$$f(x) := \sum_{i=1}^{N} f_i(x)$$

- **Goal**:

$$\min_x f(x)$$

- **Gradient algorithm**:

$$x_{k+1} = x_k - \alpha_k \sum_{i=1}^{N} \nabla f_i(x_k)$$



Distributed gradient descent

- $\mathcal{G} = (V, \mathcal{E})$ undirected
- $f_i : \mathbb{R} \to \mathbb{R}$, local function known only by node i
- x_i : local copy of x stored in memory by node i

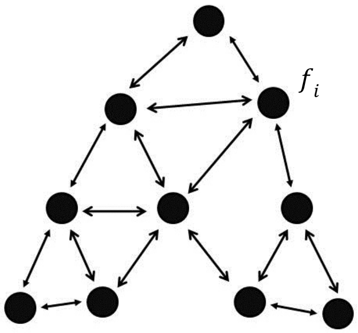
$$\min_{x_1,\ldots,x_N} \sum_{i=1}^{N} f_i(x_i)$$

$$\text{s.t. } x_1 = \ldots = x_N$$

consensus constraint

- $x = [x_1, \ldots, x_N]$
- **Idea ?**

$$x_{k+1} = x_k - \alpha_k \sum_{i=1}^{N} \nabla f_i(x_k) \qquad \mapsto \qquad x_{i,k+1} = x_{i,k} - \alpha_k N \left(\frac{1}{N} \sum_{i=1}^{N} \nabla f_i(x_{i,k}) \right)$$



Distributed gradient descent

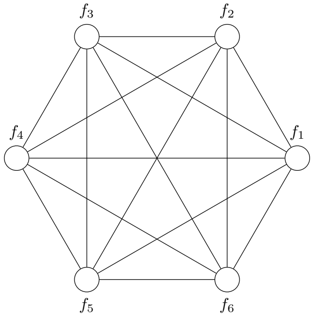
Assumtpion

- the graph is complete
- the local states are initialized all equal, that is,

$$x_{1,0} = \ldots = x_{N,0} = \bar{x}$$

A prototype for **Distributed Gradient Descent** is

$$x_{i,k+1} = x_{i,k} - \alpha_k \, N \left(\frac{1}{N} \sum_{i=1}^{N} \nabla f_i(x_{i,k}) \right)$$



Distributed gradient descent

Assumtpion

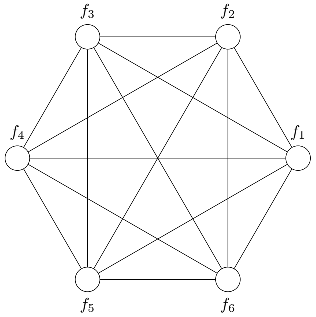
- the graph is complete
- the local states are initialized all equal, that is,

$$x_{1,0} = \ldots = x_{N,0} = \bar{x}$$

A prototype for **Distributed Gradient Descent** is

$$x_{i,k+1} = x_{i,k} - \alpha_k \, N \left(\frac{1}{N} \sum_{i=1}^{N} \nabla f_i(x_{i,k}) \right)$$

can be computed in one step since the graph is complete



Distributed gradient descent

Let $x_k = [x_{1,k}, \ldots, x_{N,k}]^\top$ then

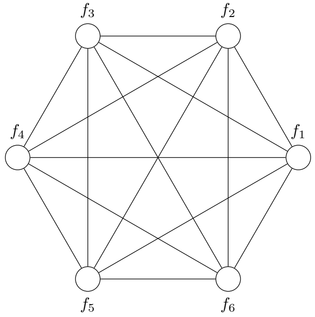
$$x_{k+1} = x_k - \alpha_k W \nabla f(x_k)$$

where

- $x_k = [x_{1,k}, \ldots, x_{N,k}]$
- $\nabla f(x_k) = [\nabla f_1(x_{1,k}), \ldots \nabla f_N(x_{N,k})]^\top$
- $W = \frac{1}{N}\mathbf{1}\mathbf{1}^T = \begin{bmatrix} 1/N & \cdots & 1/N \\ \vdots & & \vdots \\ 1/N & \cdots & 1/N \end{bmatrix} := J$

We have

$$x_{k+1} = x_k - \alpha_k \, N \left(\frac{1}{N} \mathbf{1}^\top \nabla f(x_k) \right) \mathbf{1}$$



Distributed gradient descent

Let $\beta_k = \frac{1}{N}\mathbf{1}^\top \nabla f(x_k)$ then

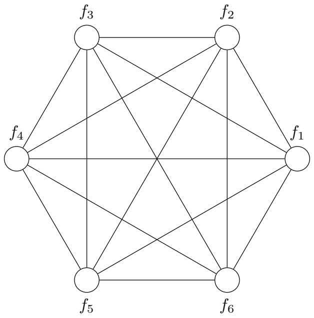
$$\begin{bmatrix} x_{1,k+1} \\ \vdots \\ x_{N,k+1} \end{bmatrix} = \begin{bmatrix} x_{1,k} \\ \vdots \\ x_{N,k} \end{bmatrix} - \alpha N \begin{bmatrix} \beta_k \\ \vdots \\ \beta_k \end{bmatrix}$$

Since $x_{i,0} = \bar{x}$ for all i then

$$x_{1,k} = \ldots = x_{N,k}, \qquad \forall\, k$$

and

$$x_{1,k} = \ldots = x_{N,k} \to x_*$$



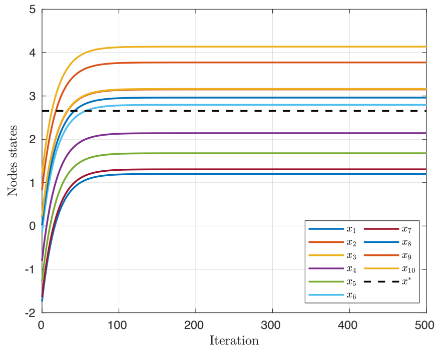
owards a distributed gradient descent algorithm

Question: What about if the initial conditions are not exactly the same?

$$\begin{bmatrix} x_{1,k+1} \\ \vdots \\ x_{N,k+1} \end{bmatrix} = \begin{bmatrix} x_{1,k} \\ \vdots \\ x_{N,k} \end{bmatrix} - \alpha N \begin{bmatrix} \beta_k \\ \vdots \\ \beta_k \end{bmatrix}$$

Trajectories are parallel

$$x_{k+1} = x_k - \alpha\, \beta_k \mathbf{1}$$



Towards a distributed gradient descent algorithm

Idea: average consensus also over the states

$$\begin{aligned} x_{k+1} &= Jx_k - \alpha J \nabla f(x_k), \qquad J = \frac{1}{N}\mathbf{1}\mathbf{1}^\top \\ &= J\left(x_k - \alpha \nabla f(x_k)\right) \end{aligned}$$

Then, again,

$$x_{1,k} = \ldots = x_{N,k}, \qquad \forall\, k \geq 0$$

Towards a distributed gradient descent algorithm

Other question : what about if the graph is not complete?

Idea : we use a doubly stochastic matrix built over the graph (*we substitute J with W*)

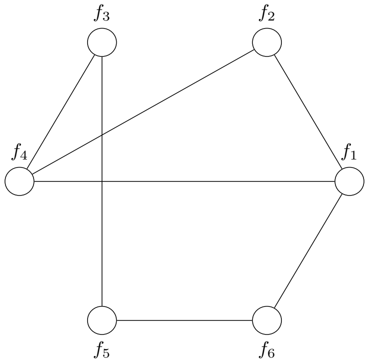
$$\begin{aligned} x_{k+1} &= W x_k - \alpha_k W \nabla f(x_k) \\ &= W\left(x_k - \alpha_k \nabla f(x_k)\right) \end{aligned}$$

Or, alternatively

$$x_{k+1} = W x_k - \alpha_k \nabla f(x_k)$$

consensus only on the states and not on the gradients - privacy reasons

$$x_{i,k+1} = \sum_{j \in \mathcal{N}_i} w_{ij} x_{j,k} - \alpha_k \nabla f_i(x_{i,k}) \qquad component - wise$$

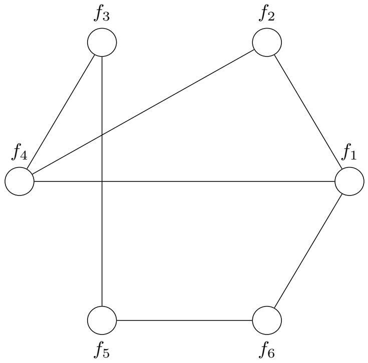


Distributed gradient descent algorithm

Question: Is this algorithm converging?

$$x_{i,k+1} = \sum_{j \in \mathcal{N}_i} w_{ij} x_{j,k} - \alpha_k \nabla f_i(x_{i,k})$$

Assumption : α_k is constant, that is, $\alpha_k = \alpha$ for all k (*step-size constant*)



Distributed gradient descent algorithm

Question: Is this algorithm converging?

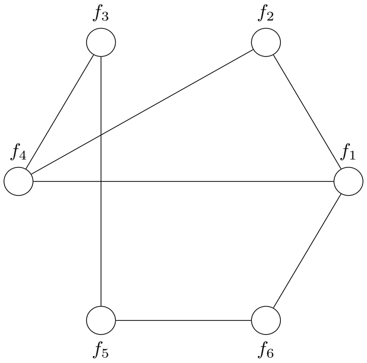
$$x_{i,k+1} = \sum_{j \in \mathcal{N}_i} w_{ij} x_{j,k} - \alpha_k \nabla f_i(x_{i,k})$$

Assumption : α_k is constant, that is, $\alpha_k = \alpha$ for all k (*step-size constant*)

Observation : x^* is not a fixed point
Indeed

$$x^* = \sum_{j \in \mathcal{N}_i} w_{ij} x^* - \alpha_k \nabla f_i(x^*)\,?$$

$x^* = x^* - \alpha_k \nabla f_i(x^*)\,?$ No! In general $0 \neq \nabla f_i(x^*)$



Distributed gradient descent algorithm

Question: Is this algorithm converging?

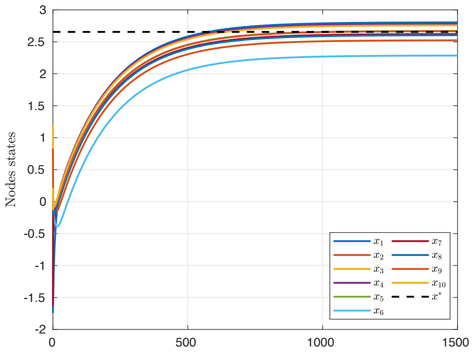
$$x_{i,k+1} = \sum_{j \in \mathcal{N}_i} w_{ij} x_{j,k} - \alpha_k \nabla f_i(x_{i,k})$$

Assumption : α_k is constant, that is, $\alpha_k = \alpha$ for all k (*step-size constant*)

Result : If α satisfies

$$\alpha < \frac{1 + \lambda_{\text{min}}(W)}{L}$$

then $x(t)$ converges to a neighborhood of $x_* \mathbf{1}$ but, in general, not $x_* \mathbf{1}$ itself.



Distributed gradient descent algorithm

Question:Why doesn't this algorithm reach x_*?

- we have already seen that $x_* \mathbf{1}$ is not a fixed point for the updating rule
- *anti-consensus push* behavior; if $x_{i,0} = \bar{x}$ then

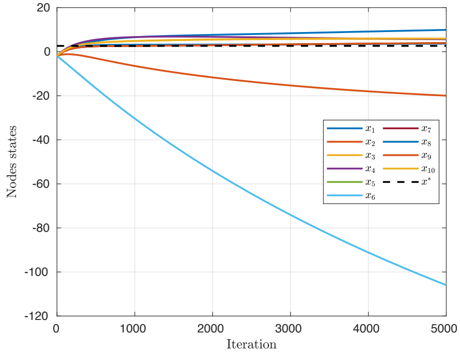
$$\begin{aligned} x_{i,1} &= \sum_{j \in \mathcal{N}_i} w_{ij} \bar{x} - \alpha \nabla f_i(\bar{x}) \\ &= \bar{x} - \alpha \nabla f_i(\bar{x}) \end{aligned}$$

It holds

$$x_{i,1} = x_{j,1} \qquad \Leftrightarrow \qquad \nabla f_i(\bar{x}) = \nabla f_j(\bar{x})$$

- the DGD algorithm solves the following regularized problem

$$\min_{x_1, \dots, x_N} \; \alpha \sum_{i=1}^{N} f_i(x_i) + \frac{1}{2} x^\top (I - W) x$$



Distributed gradient descent algorithm

Question:time-varying step size α_k?

$$x_{i,k+1} = \sum_{j \in \mathcal{N}_i} w_{ij} x_{j,k} - \alpha_k \nabla f_i(x_{i,k})$$

Result : let $\{\alpha_k\}_{k=0}^{\infty}$, $\alpha_k > 0$, be such that

$$\sum_{k=0}^{\infty} \alpha_k = \infty \qquad \text{and} \qquad \sum_{k=0}^{\infty} \alpha_k^2 < \infty$$

then

$$\lim_{k \to \infty} x_k = x_* \mathbf{1}$$

Distributed gradient descent algorithm

Question:time-varying step size α_k?

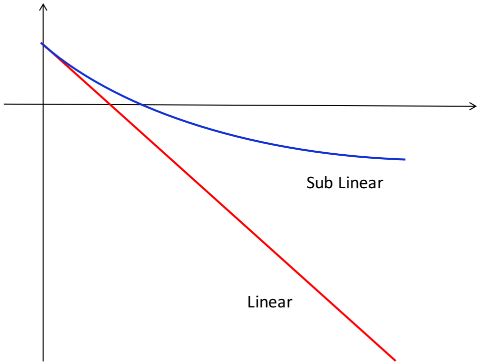
$$x_{i,k+1} = \sum_{j\in\mathcal{N}_i} w_{ij}x_{j,k} - \alpha_k \nabla f_i(x_{i,k})$$

Result : let $\{\alpha_k\}_{k=0}^{\infty}$, $\alpha_k > 0$, be such that

$$\sum_{k=0}^{\infty} \alpha_k = \infty \qquad \text{and} \qquad \sum_{k=0}^{\infty} \alpha_k^2 < \infty$$

then

$$\lim_{k\to\infty} x_k = x_* \mathbf{1}$$



Remark : sub-linear rate

How to obtain linear rate? : Gradient tracking, Distributed ADMM

Outline

- Descent algorithms (Gradient/ Newton-Raphson)
- On Operators (monotone, strongly monotone, Lipschitz continuous, Co-coercive)
- Convex set, convex and strongly convex functions
- Properties of the gradient operator
- Gradient algorithm
- Consensus Optimization over networks
- Elements of Graph Theory
- Consensus algorithms
- Distributed Gradient Descent
- Distributed Gradient Tracking

From DGD to distributed gradient tracking

Distributed Gradient Descent:

$$x_{i,k+1} = \sum_{j \in \mathcal{N}_i} w_{ij} x_{j,k} - \alpha_k \nabla f_i(x_{i,k})$$

Idea : Replace $\nabla f_i(x_{i,k})$ with a tracker of the average of the local gradients $\frac{1}{N}\sum_{i=1}^N \nabla f_i(x_{i,k})$
(*from local to dynamic consensus*)

$$x_{i,k+1} = \sum_{j \in \mathcal{N}_i} w_{ij} x_{j,k} - \alpha_k s_{i,k}$$

$$s_{i,k} \text{ : tracker of } \frac{1}{N}\sum_{i=1}^N \nabla f_i(x_{i,k})$$

Dynamic average consensus (tracking of time-varying signals)

$$x_{i,k+1} = \sum_{j=1}^N w_{ij} x_{j,k} + r_{i,k+1} - r_{i,k}$$

$$x_{k+1} = W x_k + r_{k+1} - r_k$$

$$x_{i,0} = r_{i,0} \quad \Rightarrow \quad \mathbf{1}^\top x_0 = \mathbf{1}^\top r_0$$

mass preservation

$$x_i \text{ : tracker of } \frac{1}{N}\sum_{i=1}^N r_{i,k}$$

In our case

$$x_{i,k} \to s_{i,k}$$
$$r_{i,k} \to \nabla f_i(x_{i,k})$$

Distributed gradient tracking

Distributed Gradient Tracking:

$$x_{i,k+1} = \sum_{j \in \mathcal{N}_i} w_{ij} x_{j,k} - \alpha \nabla f_i(x_{i,k})$$

$$s_{i,k+1} = \sum_{j \in \mathcal{N}_i} w_{ij} s_{j,k} + \nabla f_i(x_{i,k+1}) - \nabla f_i(x_{i,k})$$

$$s_{i,0} = \nabla f_i(x_{i,0}) \qquad \Rightarrow$$

$$\sum_{i=1}^{N} s_{i,k} = \sum_{i=0}^{N} \nabla f_i(x_{i,k})$$

mass preservation

Dynamic average consensus (tracking of time-varying signals)

$$x_{i,k+1} = \sum_{j=1}^{N} W_{ij} x_{j,k} + r_{i,k+1} - r_{i,k}$$

$$x_{k+1} = W x(k) + r_{k+1} - r_k$$

$$x_{i,0} = r_{i,0} \quad \Rightarrow \quad \mathbf{1}^\top x_0 = \mathbf{1}^\top r_0$$

mass preservation

$$x_i : \text{ tracker of } \frac{1}{N} \sum_{i=1}^{N} r_{i,k}$$

In our case

$$x_{i,k} \to s_{i,k}$$
$$r_{i,k} \to \nabla f_i(x_{i,k})$$

Distributed gradient tracking

Distributed Gradient Tracking:

$$
\begin{aligned}
x_{k+1} &= W x_k - \alpha \nabla f(x_k) \\
s_{k+1} &= W s_k + \nabla f(x_{k+1}) - \nabla f(x_k)
\end{aligned}
$$

Proposition. Assume

- for all i, f_i is L-smooth and μ-strongly convex;
- W doubly stochastic and primitive;
- $s_0 = \nabla f_(x_0)$ and x_0 arbitrary;
- α constant sufficiently small.

Then $x_k \to \mathbf{1}x_\star$ linearly, i.e., there exists $0 < \rho < 1$ and $C > 0$ such that

$$\|x_k - \mathbf{1}x_\star\| \leq C\rho^k.$$

Distributed gradient tracking

Distributed Gradient Tracking: How to prove convergence?

- singular perturbation/ time-scale separation;
- small gain theorem;
- algebraic analysis/ matrix stability.

Distributed gradient tracking

Distributed Gradient Tracking: How to prove convergence?

- singular perturbation/ time-scale separation;
- small gain theorem;
- algebraic analysis/ matrix stability.

Let

$$x_{\text{ave},k} = \frac{1}{N}\mathbf{1}^T x_k, \qquad s_{\text{ave},k} = \frac{1}{N}\mathbf{1}^T s_k, \qquad \nabla f_{\text{ave},k} = \frac{1}{N}\mathbf{1}^T \nabla f(x_k)$$

then

$$\begin{bmatrix} \|s_{k+1} - s_{\text{ave},k+1}\mathbf{1}\| \\ \|x_{k+1} - x_{\text{ave},k+1}\mathbf{1}\| \\ \sqrt{N}\|x_{\text{ave},k+1} - x_*\| \end{bmatrix} \leq \begin{bmatrix} \sigma + L\alpha & L(\alpha L + 2) & \alpha L^2 \\ \alpha & \sigma & 0 \\ 0 & \alpha\beta & \lambda \end{bmatrix} \begin{bmatrix} \|s_k - s_{\text{ave},k}\mathbf{1}\| \\ \|x_k - x_{\text{ave},k}\mathbf{1}\| \\ \sqrt{N}\|x_{\text{ave},k} - x_*\| \end{bmatrix}$$

where $\lambda = \max\{|1-\mu\alpha|, |1-L\alpha|\}$ and where σ depends on spectral properties of W

It is possible to prove that there exists $\bar{\alpha} > 0$ such that for $0 < \alpha < \bar{\bar{\alpha}}$ the above matrix is Schur stable.