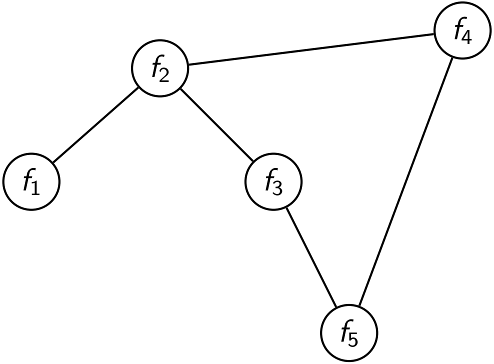
A Canonical Form for First-Order Distributed Optimization Algorithms

Akhil Sundararajan   Bryan Van Scoy   Laurent Lessard

University of Wisconsin–Madison

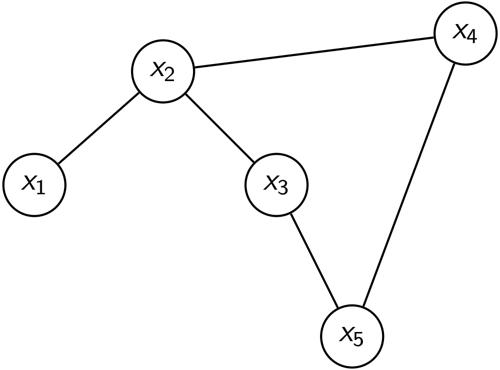
July 12, 2019

Distributed optimization



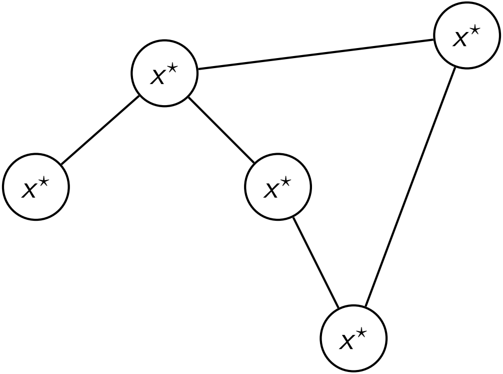
$$x^\star = \arg\min_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} f_i(x)$$

Distributed optimization



$$x^\star = \arg\min_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} f_i(x)$$

Distributed optimization



We must achieve both **consensus** and **optimality**.

DGD [Nedic,Ozdaglar, 2009]

$$x_i^{k+1} = \sum_{j=1}^{n} w_{ij} x_j^k - \alpha_k \nabla f_i(x_i^k)$$

- local state $x_i \in \mathbb{R}^d$ for each agent $i \in \{1, \ldots, n\}$

DGD [Nedic,Ozdaglar, 2009]

$$x_i^{k+1} = \sum_{j=1}^{n} w_{ij} x_j^k - \alpha_k \nabla f_i(x_i^k)$$

- local state $x_i \in \mathbb{R}^d$ for each agent $i \in \{1, \ldots, n\}$
- mixing weights w_{ij}

DGD [Nedic,Ozdaglar, 2009]

$$x_i^{k+1} = \sum_{j=1}^{n} w_{ij} x_j^k - \alpha_k \nabla f_i(x_i^k)$$

- local state $x_i \in \mathbb{R}^d$ for each agent $i \in \{1, \ldots, n\}$
- mixing weights w_{ij}
- converges slowly even for strongly convex f_i

DGD [Nedic,Ozdaglar, 2009]

$$x_i^{k+1} = \sum_{j=1}^{n} w_{ij} x_j^k - \alpha_k \nabla f_i(x_i^k)$$

- local state $x_i \in \mathbb{R}^d$ for each agent $i \in \{1, \ldots, n\}$
- mixing weights w_{ij}
- converges slowly even for strongly convex f_i

Want **linear** (exponential) convergence: $\|x_i^k - x^\star\| = O(\rho^k)$.

- If $n = 1$ (ordinary gradient descent) or
- If f_i are quadratic (average consensus).

Motivation

Exact Diffusion [Yuan, et al, 2017]

$$x_i^{k+1} = z_i^k - \alpha \nabla f_i(z_i^k)$$
$$z_i^{k+1} = \sum_{j=1}^{n} w_{ij} \left(x_j^{k+1} - x_j^k + z_j^k \right)$$

Motivation

Exact Diffusion [Yuan, et al, 2017]

$$x_i^{k+1} = z_i^k - \alpha \nabla f_i(z_i^k)$$
$$z_i^{k+1} = \sum_{j=1}^{n} w_{ij} \left(x_j^{k+1} - x_j^k + z_j^k\right)$$

NIDS [Li, et al, 2017]

$$x_i^{k+2} = \sum_{j=1}^{n} \widetilde{w}_{ij} \big(2x_j^{k+1} - x_j^k - \alpha \nabla f_j(x_j^{k+1}) + \alpha \nabla f_j(x_j^k)\big)$$

How does one design a distributed algorithm?

Inspirations include:

How does one design a distributed algorithm?

Inspirations include:

- dual decomposition

How does one design a distributed algorithm?

Inspirations include:

- dual decomposition
- discretization of ODEs

How does one design a distributed algorithm?

Inspirations include:

- dual decomposition
- discretization of ODEs
- gradient tracking

How does one design a distributed algorithm?

Inspirations include:

- dual decomposition
- discretization of ODEs
- gradient tracking

Lots of structural variety!

Motivation

Exact Diffusion [Yuan, et al, 2017]

$$x_i^{k+1} = z_i^k - \alpha\,\nabla f_i(z_i^k)$$
$$z_i^{k+1} = \sum_{j=1}^{n} w_{ij}\,(x_j^{k+1} - x_j^k + z_j^k)$$

NIDS [Li, et al, 2017]

$$x_i^{k+2} = \sum_{j=1}^{n} \widetilde{w}_{ij}\big(2x_j^{k+1} - x_j^k - \alpha\nabla f_j(x_j^{k+1}) + \alpha\nabla f_j(x_j^k)\big)$$

Motivation

Exact Diffusion [Yuan, et al, 2017]

$$x_i^{k+1} = z_i^k - \alpha \nabla f_i(z_i^k)$$
$$z_i^{k+1} = \sum_{j=1}^{n} w_{ij} \left(x_j^{k+1} - x_j^k + z_j^k\right)$$

NIDS [Li, et al, 2017]

$$x_i^{k+2} = \sum_{j=1}^{n} \widetilde{w}_{ij}\big(2x_j^{k+1} - x_j^k - \alpha\nabla f_j(x_j^{k+1}) + \alpha\nabla f_j(x_j^k)\big)$$

NIDS and Exact Diffusion are in fact the same!

Algorithm family

from neighbors

$$\begin{bmatrix}\xi_j^k \\ u_j^k\end{bmatrix}$$

Agent i

$$\begin{bmatrix}\xi_i^{k+1} \\ y_i^k\end{bmatrix} = \begin{bmatrix}A_0 & B_0 \\ C_0 & D_0\end{bmatrix}\begin{bmatrix}\xi_i^k \\ u_i^k\end{bmatrix} + \begin{bmatrix}A_1 & B_1 \\ C_1 & D_1\end{bmatrix}\sum_{j=1}^{n} L_{ij}\begin{bmatrix}\xi_j^k \\ u_j^k\end{bmatrix}$$

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

to neighbors

$$\begin{bmatrix}\xi_i^k \\ u_i^k\end{bmatrix}$$

Algorithm family

from neighbors

$$\begin{bmatrix}\xi_j^k\\ u_j^k\end{bmatrix}$$

Agent i

$$\begin{bmatrix}\xi_i^{k+1}\\ y_i^k\end{bmatrix} = \begin{bmatrix}A_0 & B_0\\ C_0 & D_0\end{bmatrix}\begin{bmatrix}\xi_i^k\\ u_i^k\end{bmatrix} + \begin{bmatrix}A_1 & B_1\\ C_1 & D_1\end{bmatrix}\sum_{j=1}^{n} L_{ij}\begin{bmatrix}\xi_j^k\\ u_j^k\end{bmatrix}$$

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

to neighbors

$$\begin{bmatrix}\xi_i^k\\ u_i^k\end{bmatrix}$$

- local state ξ_i and gradient u_i are communicated

Algorithm family

from neighbors

$$\begin{bmatrix} \xi_j^k \\ u_j^k \end{bmatrix}$$

Agent i

$$\begin{bmatrix} \xi_i^{k+1} \\ y_i^k \end{bmatrix} = \begin{bmatrix} A_0 & B_0 \\ C_0 & D_0 \end{bmatrix} \begin{bmatrix} \xi_i^k \\ u_i^k \end{bmatrix} + \begin{bmatrix} A_1 & B_1 \\ C_1 & D_1 \end{bmatrix} \sum_{j=1}^{n} L_{ij} \begin{bmatrix} \xi_j^k \\ u_j^k \end{bmatrix}$$

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

to neighbors

$$\begin{bmatrix} \xi_i^k \\ u_i^k \end{bmatrix}$$

- local state ξ_i and gradient u_i are communicated
- local gradient evaluated at y_i

Algorithm family

from neighbors

$$\begin{bmatrix} \xi_j^k \\ u_j^k \end{bmatrix}$$

Agent i

$$\begin{bmatrix} \xi_i^{k+1} \\ y_i^k \end{bmatrix} = \begin{bmatrix} A_0 & B_0 \\ C_0 & D_0 \end{bmatrix} \begin{bmatrix} \xi_i^k \\ u_i^k \end{bmatrix} + \begin{bmatrix} A_1 & B_1 \\ C_1 & D_1 \end{bmatrix} \sum_{j=1}^{n} L_{ij} \begin{bmatrix} \xi_j^k \\ u_j^k \end{bmatrix}$$

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

to neighbors

$$\begin{bmatrix} \xi_i^k \\ u_i^k \end{bmatrix}$$

- local state ξ_i and gradient u_i are communicated
- local gradient evaluated at y_i

Parameterization is pretty general. But...

Algorithm family

from neighbors $\begin{bmatrix} \xi_j^k \\ u_j^k \end{bmatrix}$

Agent i

$$\begin{bmatrix} \xi_i^{k+1} \\ y_i^k \end{bmatrix} = \begin{bmatrix} A_0 & B_0 \\ C_0 & D_0 \end{bmatrix} \begin{bmatrix} \xi_i^k \\ u_i^k \end{bmatrix} + \begin{bmatrix} A_1 & B_1 \\ C_1 & D_1 \end{bmatrix} \sum_{j=1}^{n} L_{ij} \begin{bmatrix} \xi_j^k \\ u_j^k \end{bmatrix}$$

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

to neighbors $\begin{bmatrix} \xi_i^k \\ u_i^k \end{bmatrix}$

- local state ξ_i and gradient u_i are communicated
- local gradient evaluated at y_i

Parameterization is pretty general. But...

- not all choices of $(A_0, B_0, C_0, D_0, A_1, B_1, C_1, D_1)$ are valid

Algorithm family

from neighbors $\begin{bmatrix} \xi_j^k \\ u_j^k \end{bmatrix}$

Agent i

$$\begin{bmatrix} \xi_i^{k+1} \\ y_i^k \end{bmatrix} = \begin{bmatrix} A_0 & B_0 \\ C_0 & D_0 \end{bmatrix} \begin{bmatrix} \xi_i^k \\ u_i^k \end{bmatrix} + \begin{bmatrix} A_1 & B_1 \\ C_1 & D_1 \end{bmatrix} \sum_{j=1}^{n} L_{ij} \begin{bmatrix} \xi_j^k \\ u_j^k \end{bmatrix}$$

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

to neighbors $\begin{bmatrix} \xi_i^k \\ u_i^k \end{bmatrix}$

- local state ξ_i and gradient u_i are communicated
- local gradient evaluated at y_i

Parameterization is pretty general. But...

- not all choices of $(A_0, B_0, C_0, D_0, A_1, B_1, C_1, D_1)$ are valid
- this set is overparameterized

Main result (canonical form)

We can uniquely represent algorithms in this family using five scalars $(\alpha, \zeta_0, \zeta_1, \zeta_2, \zeta_3)$.

Main result (canonical form)

We can uniquely represent algorithms in this family using five scalars $(\alpha, \zeta_0, \zeta_1, \zeta_2, \zeta_3)$.

With $\xi_i^k := \begin{bmatrix} x_i^k \\ w_i^k \end{bmatrix}$, Agent i performs:

$$\begin{bmatrix} x_i^{k+1} \\ w_i^{k+1} \\ y_i^k \end{bmatrix} = \left[\begin{array}{cc|c} 1 & \zeta_0 & -\alpha \\ 0 & 1 & 0 \\ \hline 1 & 0 & 0 \end{array}\right] \begin{bmatrix} x_i^k \\ w_i^k \\ u_i^k \end{bmatrix} + \left[\begin{array}{cc|c} -\zeta_1 & -\zeta_2 & 0 \\ -1 & 0 & 0 \\ \hline -\zeta_3 & 0 & 0 \end{array}\right] \sum_{j=1}^n L_{ij} \begin{bmatrix} x_j^k \\ w_j^k \\ u_j^k \end{bmatrix}$$

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

Implementation

Initialize: x_i^0 arbitrary, and $w_i^0 = 0$

Implementation

Initialize: x_i^0 arbitrary, and $w_i^0 = 0$

Communicate: $v_{1i}^k = \sum_{j=1}^n L_{ij}\, x_j^k \qquad v_{2i}^k = \sum_{j=1}^n L_{ij}\, w_j^k$

Implementation

Initialize: x_i^0 arbitrary, and $w_i^0 = 0$

Communicate: $v_{1i}^k = \sum_{j=1}^n L_{ij}\, x_j^k \qquad v_{2i}^k = \sum_{j=1}^n L_{ij}\, w_j^k$

Compute gradient: $y_i^k = x_i^k - \zeta_3\, v_{1i}^k \qquad u_i^k = \nabla f_i(y_i^k)$

Implementation

Initialize: x_i^0 arbitrary, and $w_i^0 = 0$

Communicate: $v_{1i}^k = \sum_{j=1}^n L_{ij}\, x_j^k \qquad v_{2i}^k = \sum_{j=1}^n L_{ij}\, w_j^k$

Compute gradient: $y_i^k = x_i^k - \zeta_3\, v_{1i}^k \qquad u_i^k = \nabla f_i(y_i^k)$

Update state:

$$
\begin{aligned}
x_i^{k+1} &= x_i^k + \zeta_0\, w_i^k - \alpha\, u_i^k - \zeta_1 v_{1i}^k + \zeta_2 v_{2i}^k \\
w_i^{k+1} &= w_i^k - v_{1i}^k
\end{aligned}
$$

Implementation

Initialize: x_i^0 arbitrary, and $w_i^0 = 0$

Communicate: $v_{1i}^k = \sum_{j=1}^n L_{ij}\, x_j^k \qquad v_{2i}^k = \sum_{j=1}^n L_{ij}\, w_j^k$

Compute gradient: $y_i^k = x_i^k - \zeta_3\, v_{1i}^k \qquad u_i^k = \nabla f_i(y_i^k)$

Update state:

$$\begin{aligned} x_i^{k+1} &= x_i^k + \zeta_0\, w_i^k - \alpha\, u_i^k - \zeta_1 v_{1i}^k + \zeta_2 v_{2i}^k \\ w_i^{k+1} &= w_i^k - v_{1i}^k \end{aligned}$$

Existing algorithms

		α	ζ_0	ζ_1	ζ_2	ζ_3
Shi, et. al, 2015	EXTRA	α	$\frac{1}{2}$	1	0	0
Yuan, et. al, 2017	Exact Diffusion	α	$\frac{1}{2}$	1	0	$\frac{1}{2}$
Li, et. al, 2017	NIDS	α	$\frac{1}{2}$	1	0	$\frac{1}{2}$
Qu, Li, 2018	DIGing	α	0	2	1	0
Xu, 2018	AsynDGM	α	0	2	1	1
Jakovetić, 2019	$(\mathcal{B} = \beta I)$	α	$\alpha\beta$	2	1	0
Jakovetić, 2019	$(\mathcal{B} = \beta W)$	α	$\alpha\beta$	2	$1 - \alpha\beta$	0

Existing algorithms

		α	ζ_0	ζ_1	ζ_2	ζ_3
Shi, et. al, 2015	EXTRA	α	$\frac{1}{2}$	1	0	0
Yuan, et. al, 2017	Exact Diffusion	α	$\frac{1}{2}$	1	0	$\frac{1}{2}$
Li, et. al, 2017	NIDS	α	$\frac{1}{2}$	1	0	$\frac{1}{2}$
Qu, Li, 2018	DIGing	α	0	2	1	0
Xu, 2018	AsynDGM	α	0	2	1	1
Jakovetić, 2019	$(\mathcal{B} = \beta I)$	α	$\alpha\beta$	2	1	0
Jakovetić, 2019	$(\mathcal{B} = \beta W)$	α	$\alpha\beta$	2	$1 - \alpha\beta$	0

Existing algorithms

		α	ζ_0	ζ_1	ζ_2	ζ_3
Shi, et. al, 2015	EXTRA	α	$\frac{1}{2}$	1	0	0
Yuan, et. al, 2017	Exact Diffusion	α	$\frac{1}{2}$	1	0	$\frac{1}{2}$
Li, et. al, 2017	NIDS	α	$\frac{1}{2}$	1	0	$\frac{1}{2}$
Qu, Li, 2018	DIGing	α	0	2	1	0
Xu, 2018	AsynDGM	α	0	2	1	1
Jakovetić, 2019	$(\mathcal{B} = \beta I)$	α	$\alpha\beta$	2	1	0
Jakovetić, 2019	$(\mathcal{B} = \beta W)$	α	$\alpha\beta$	2	$1 - \alpha\beta$	0

Properties of canonical form

Given a distributed algorithm that converges to a solution $x^\star$,
it can be put into canonical form in a unique way.

Properties of canonical form

Given a distributed algorithm that converges to a solution $x^\star$,
it can be put into canonical form in a unique way.

Given an algorithm in canonical form,
it has a fixed point $x^\star$ that is a solution
(but doesn't necessarily converge to it).

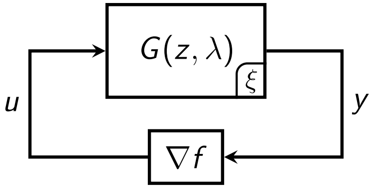
Properties of canonical form

Given a distributed algorithm that converges to a solution $x^\star$,
it can be put into canonical form in a unique way.

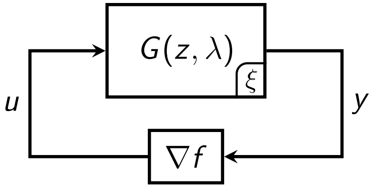
Given an algorithm in canonical form,
it has a fixed point $x^\star$ that is a solution
(but doesn't necessarily converge to it).

These results are independent of assumptions on local functions!

Multidimensional transfer function interpretation



Multidimensional transfer function interpretation



- encode constraints on fixed points in the frequency domain
- simplify $G(z, \lambda)$ according to the following:

1. $G(z, 0)$ has a pole at $z = 1$ and is marginally stable.
2. $G(z, \lambda)$ has a zero at $z = 1$ and is strictly stable for $\lambda > 0$.

Impossibility result

At least **two states** are required for a time-invariant distributed algorithm to achieve both consensus and optimality.

Impossibility result

At least **two states** are required for a time-invariant distributed algorithm to achieve both consensus and optimality.

- Explains why DGD requires a diminishing stepsize

What's next?

Conventional approach:

What's next?

Conventional approach:

- come up with a design

What's next?

Conventional approach:

- come up with a design
- prove something about it

What's next?

Conventional approach:

- come up with a design
- prove something about it
- repeat

What's next?

Conventional approach:

- come up with a design
- prove something about it
- repeat

With a canonical form:

- prove something about canonical form
- holds over broad class of algorithms

Algorithm design

- universal analysis framework

Algorithm design

- universal analysis framework
- worst-case linear rate guarantees

Algorithm design

- universal analysis framework
- worst-case linear rate guarantees
- optimal algorithm design

Algorithm design

- universal analysis framework
- worst-case linear rate guarantees
- optimal algorithm design
- algorithm robust to time-varying graphs

Algorithm design

- universal analysis framework
- worst-case linear rate guarantees
- optimal algorithm design
- algorithm robust to time-varying graphs

- check arXiv on Monday!

Thanks!