Lecture 8- Summaries of Non-Convex Distributed Optimization and Learning and Recent Advances

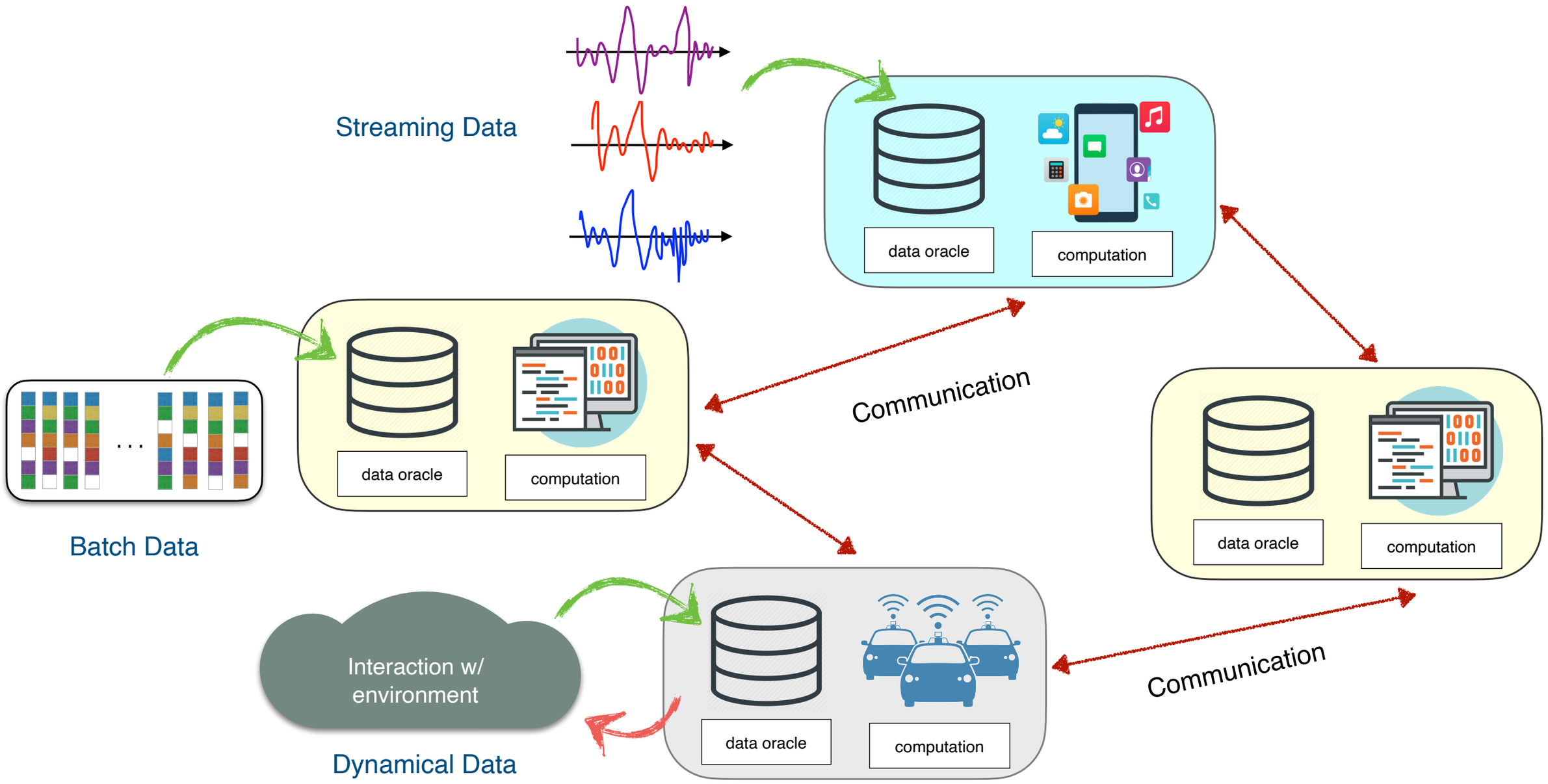
Mingyi Hong

UNIVERSITY OF MINNESOTA

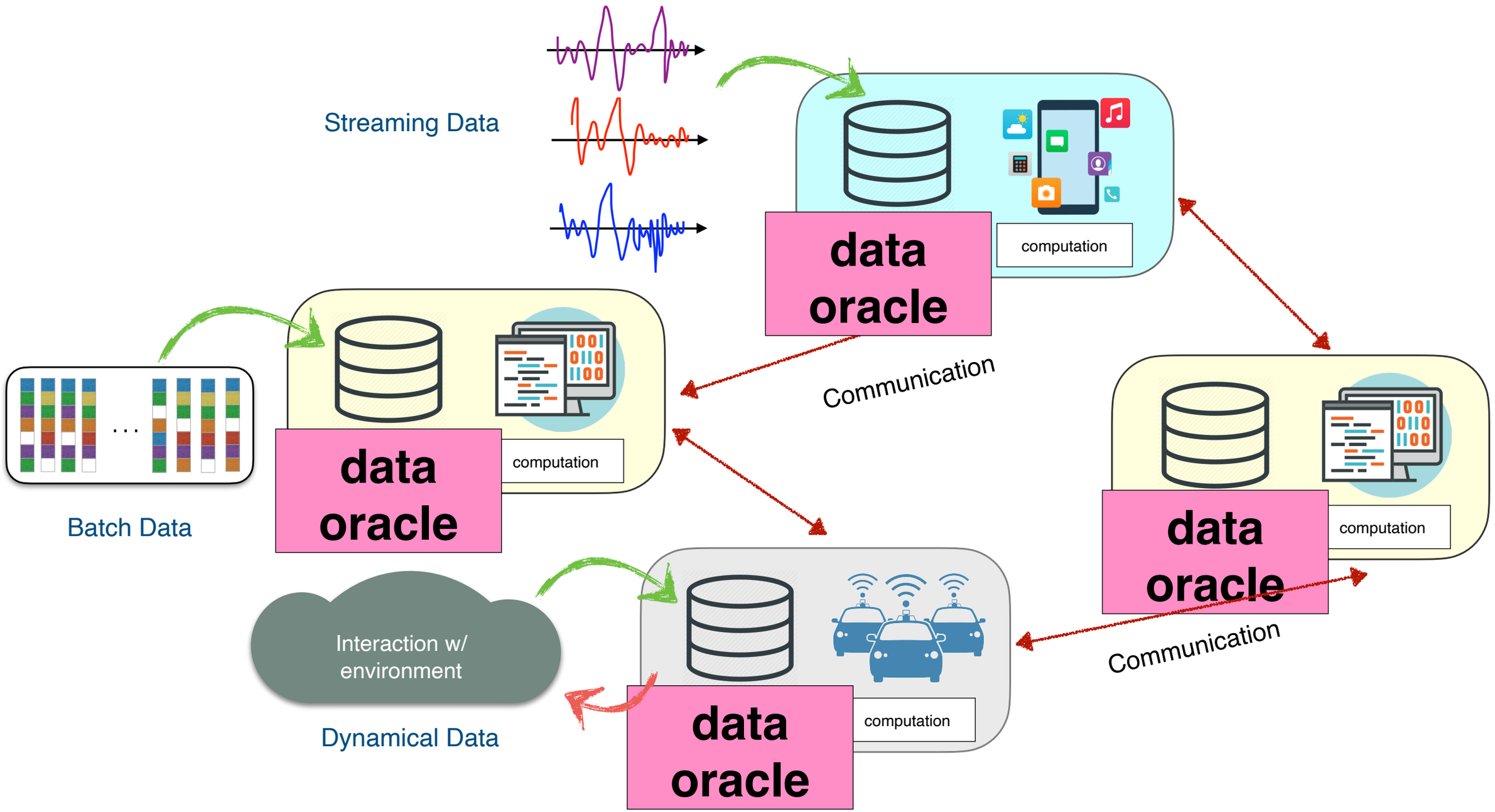
Driven to Discover℠

Introduction

Distributed Data Processing/Optimization

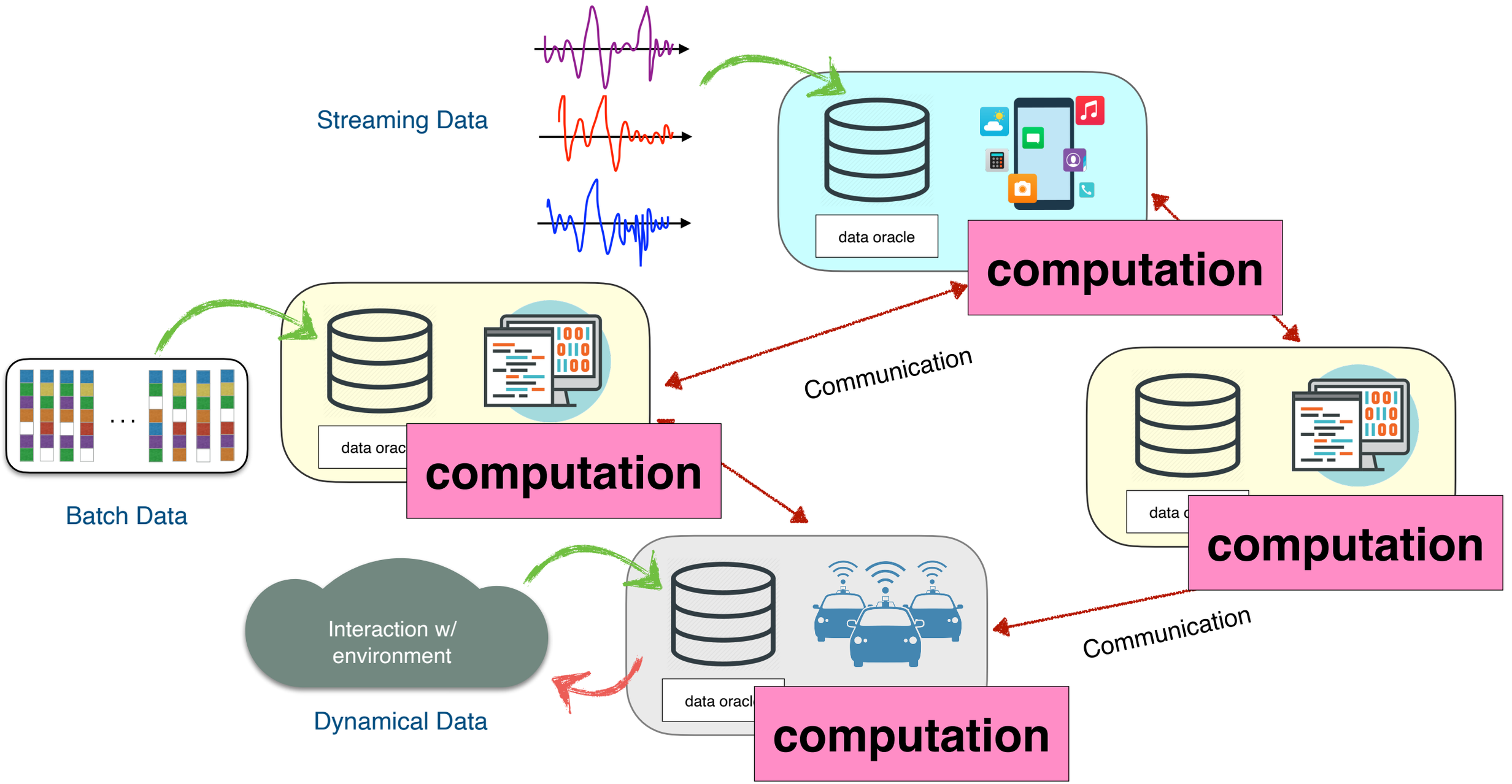


Key Elements: Data Oracle



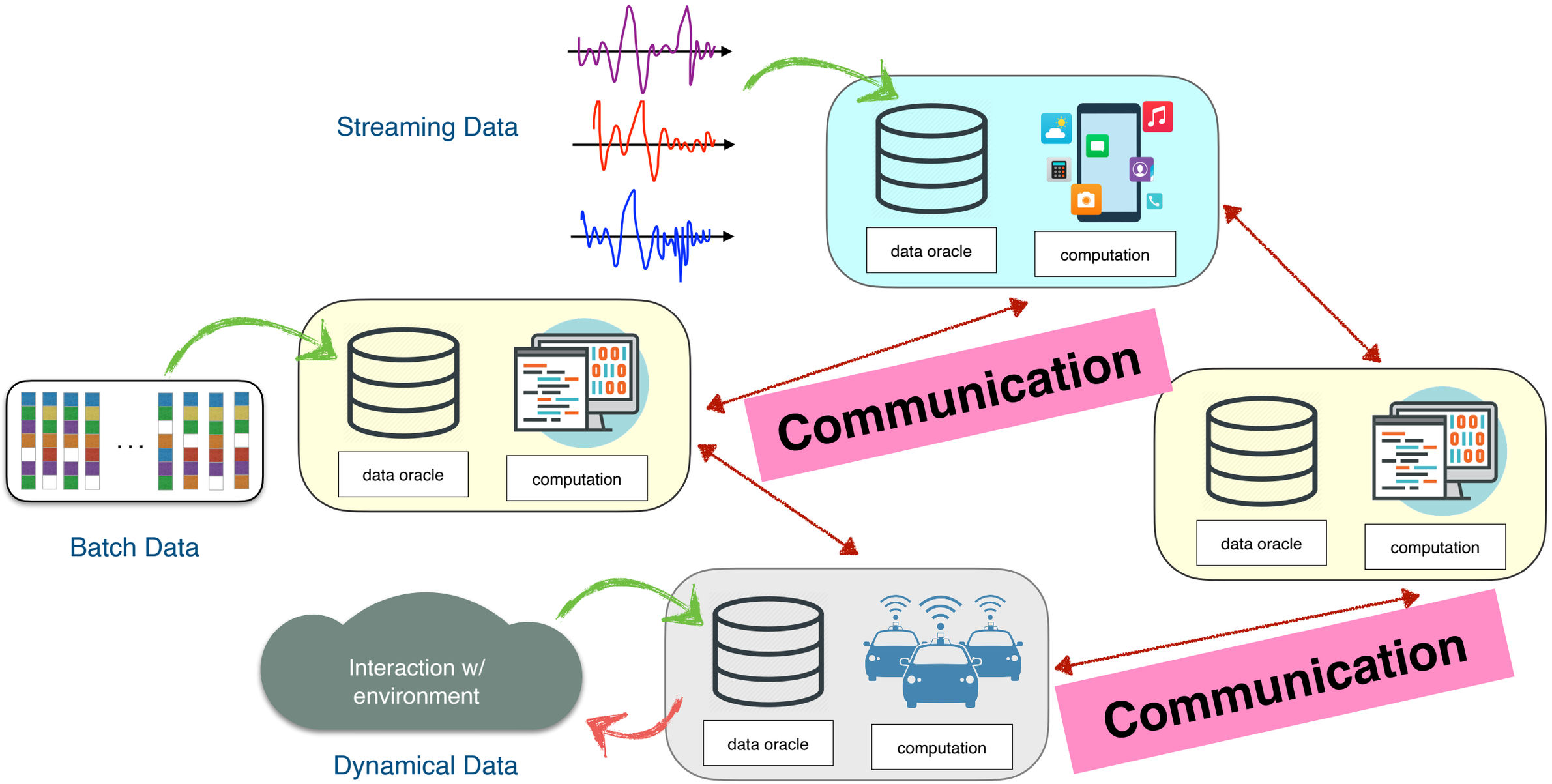
Data Oracle: Describing the data acquisition process.

Key Elements: Computation Models



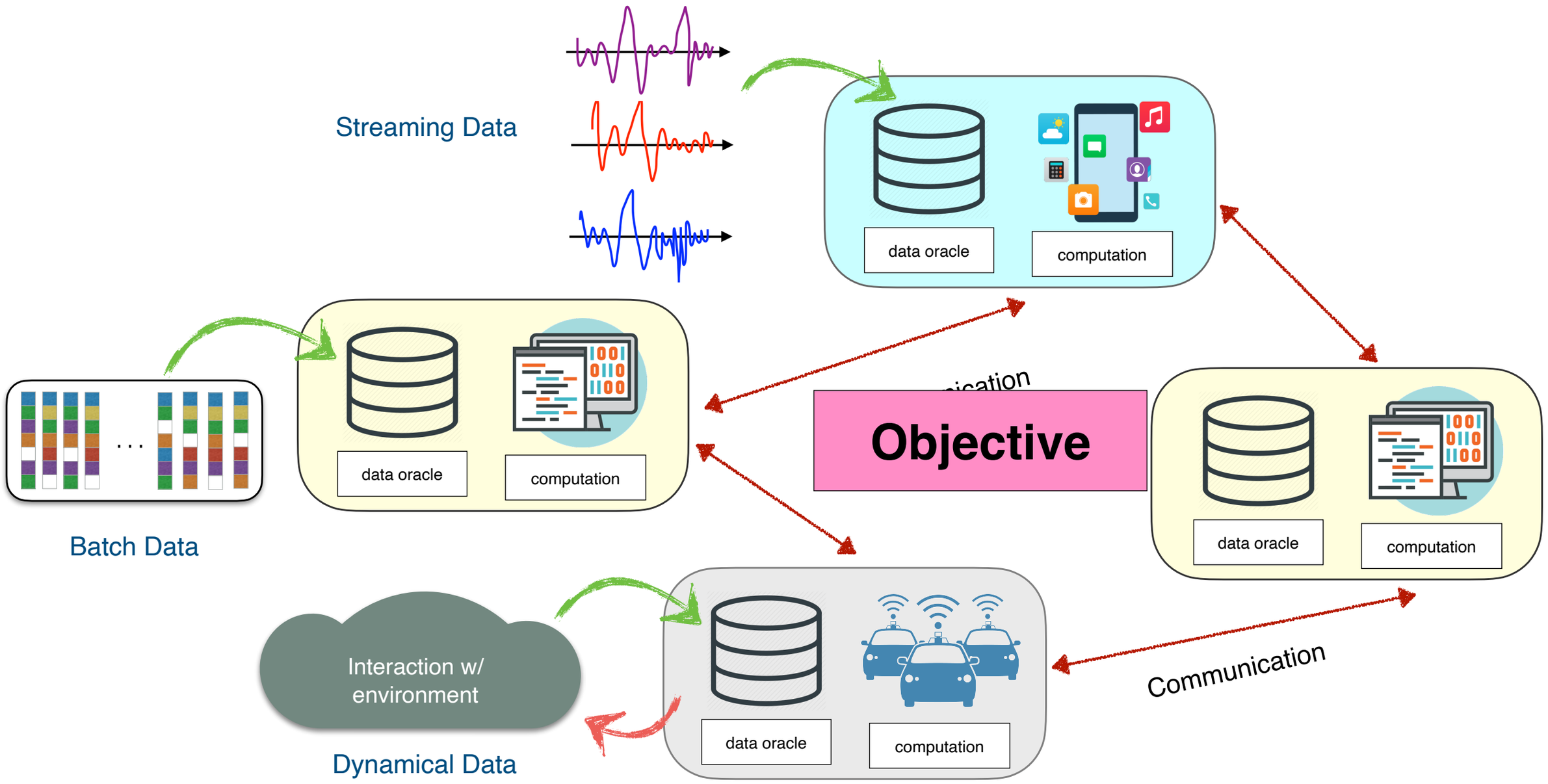
Computation Models: How the data is processed, which local problems to solved, etc…

Key Elements: Communication Models



Communication: What information are exchanged, and how?

Key Elements: Overall Objective



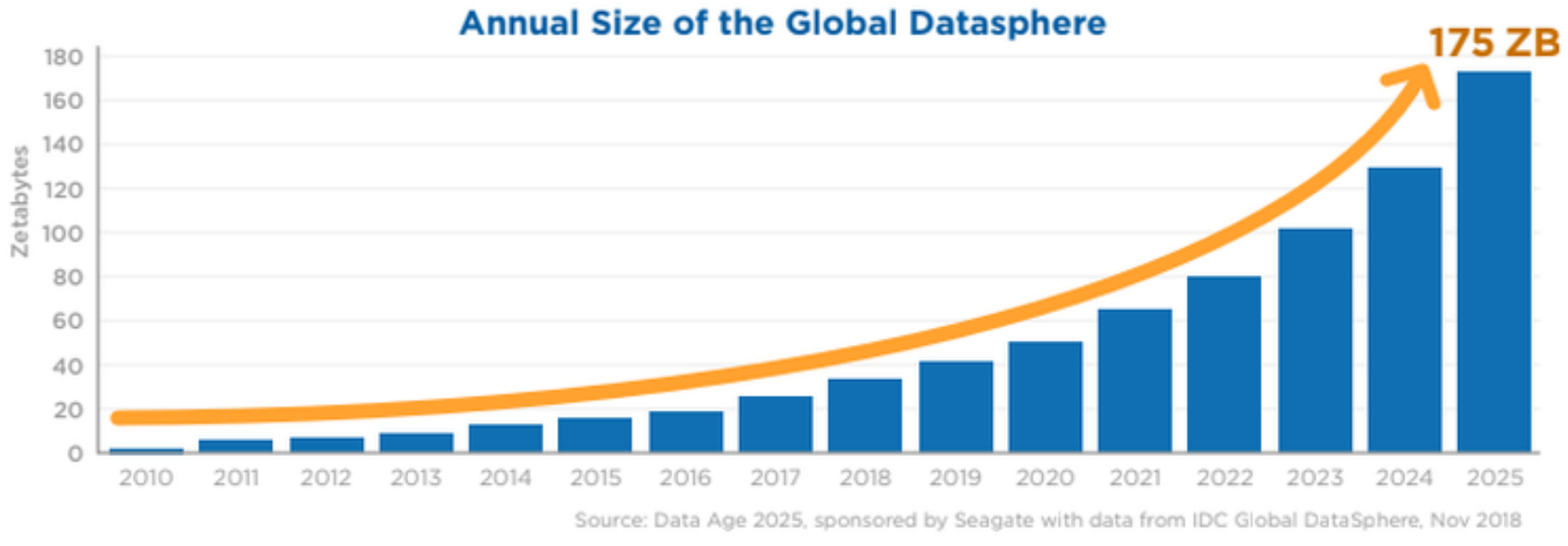
Objective: What kind of overall goal to achieve?

Why Distributed?

Explosive amount of data, collected and located in distributed locations

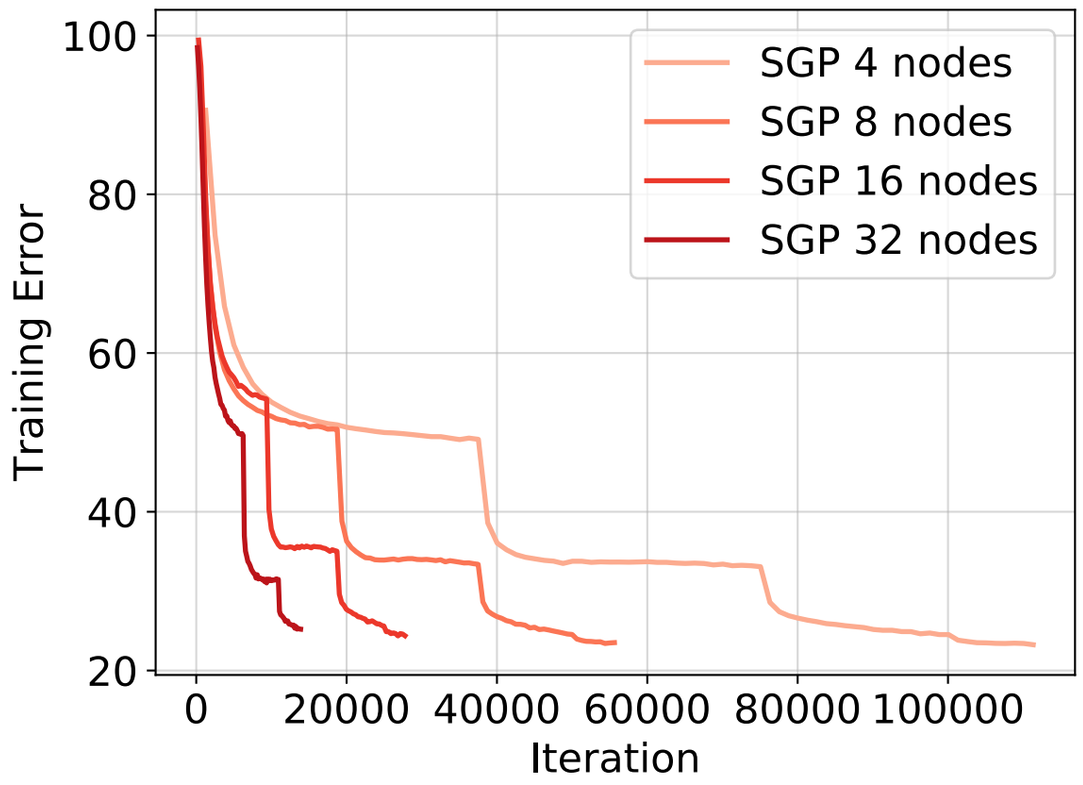
Size of the data has been growing exponentially

Figure 1 – Annual Size of the Global Datasphere

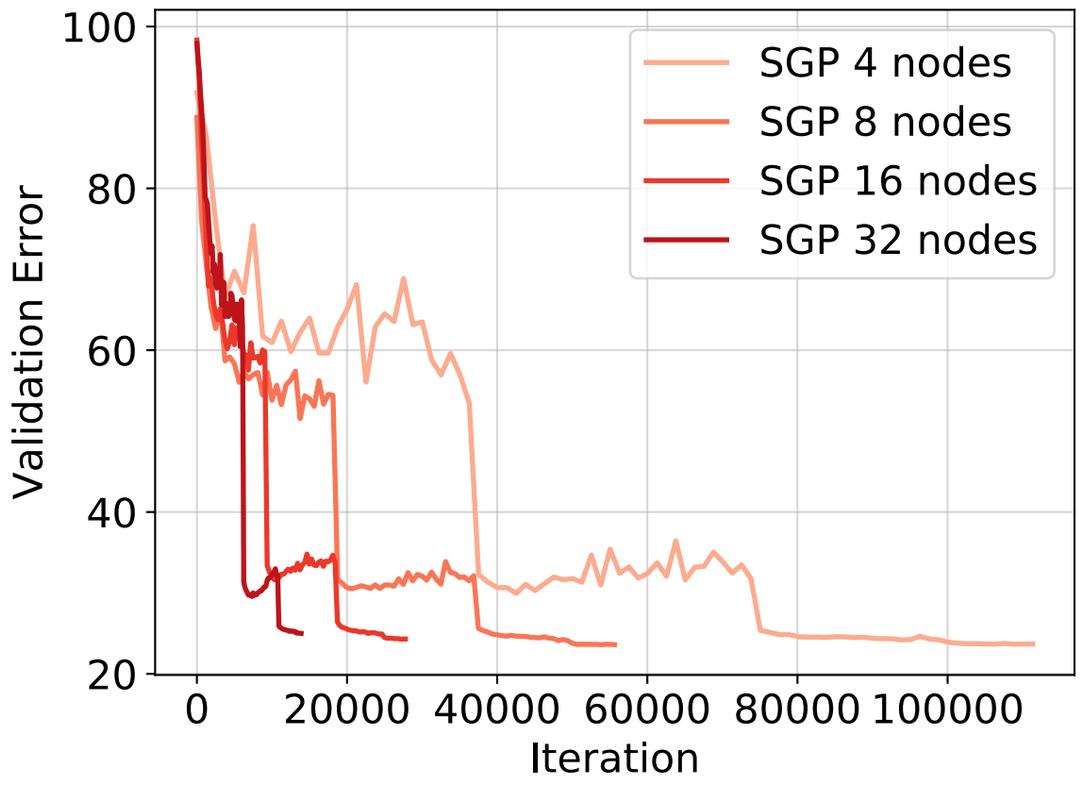


Why Distributed?

Distributed training using multiple machines/GPUs can significantly outperform centralized training [Assran et al 19]



(a) Train

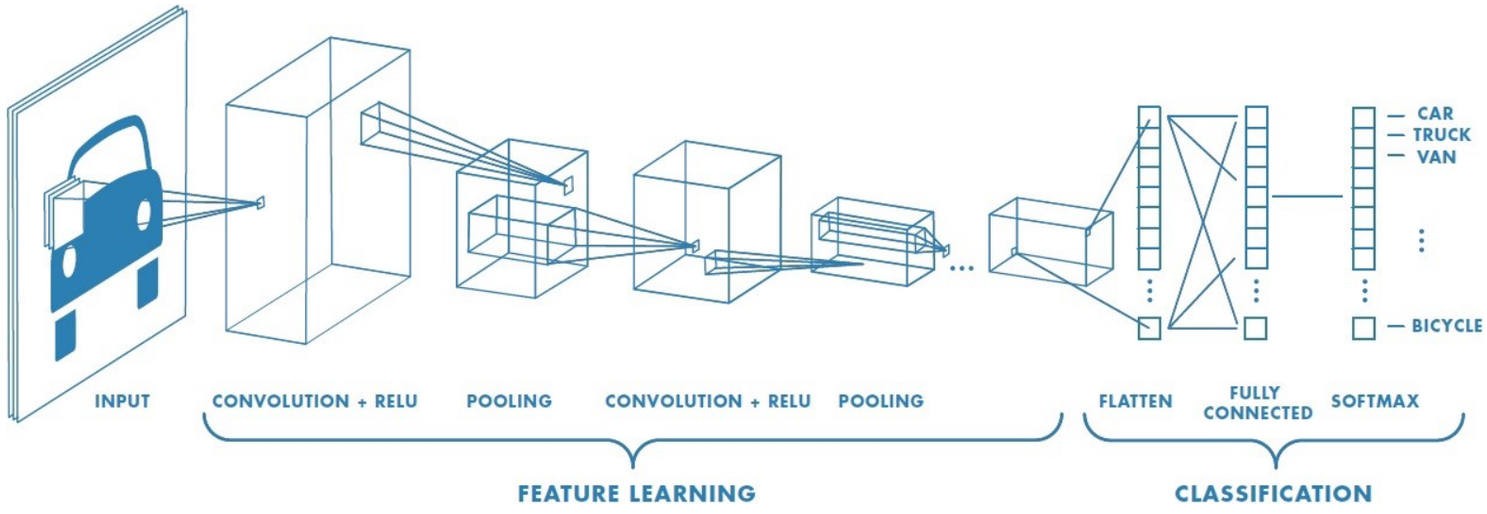


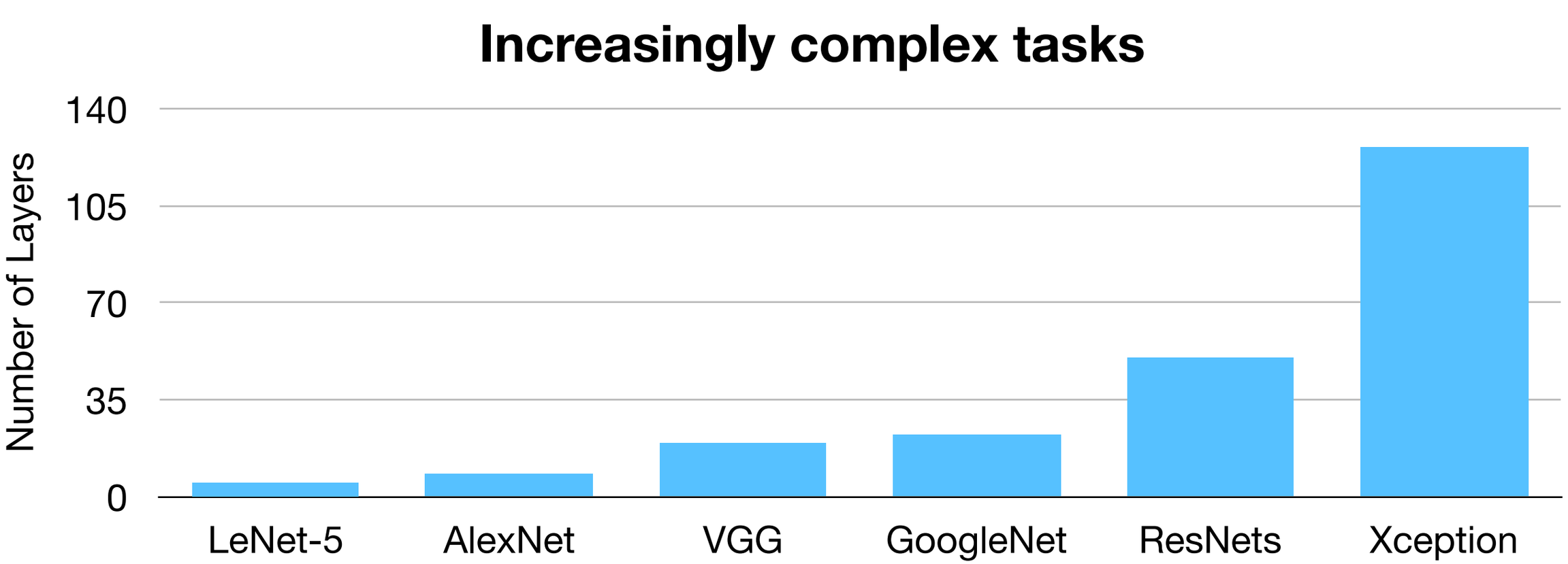
(b) Validation

Distributed Training Improves Training/Testing Accuracy

Why Non-Convexity ?

High-dimensional problems, difficult and non-convex objectives





Source: 10 CNN Architectures @ Towards Data Science

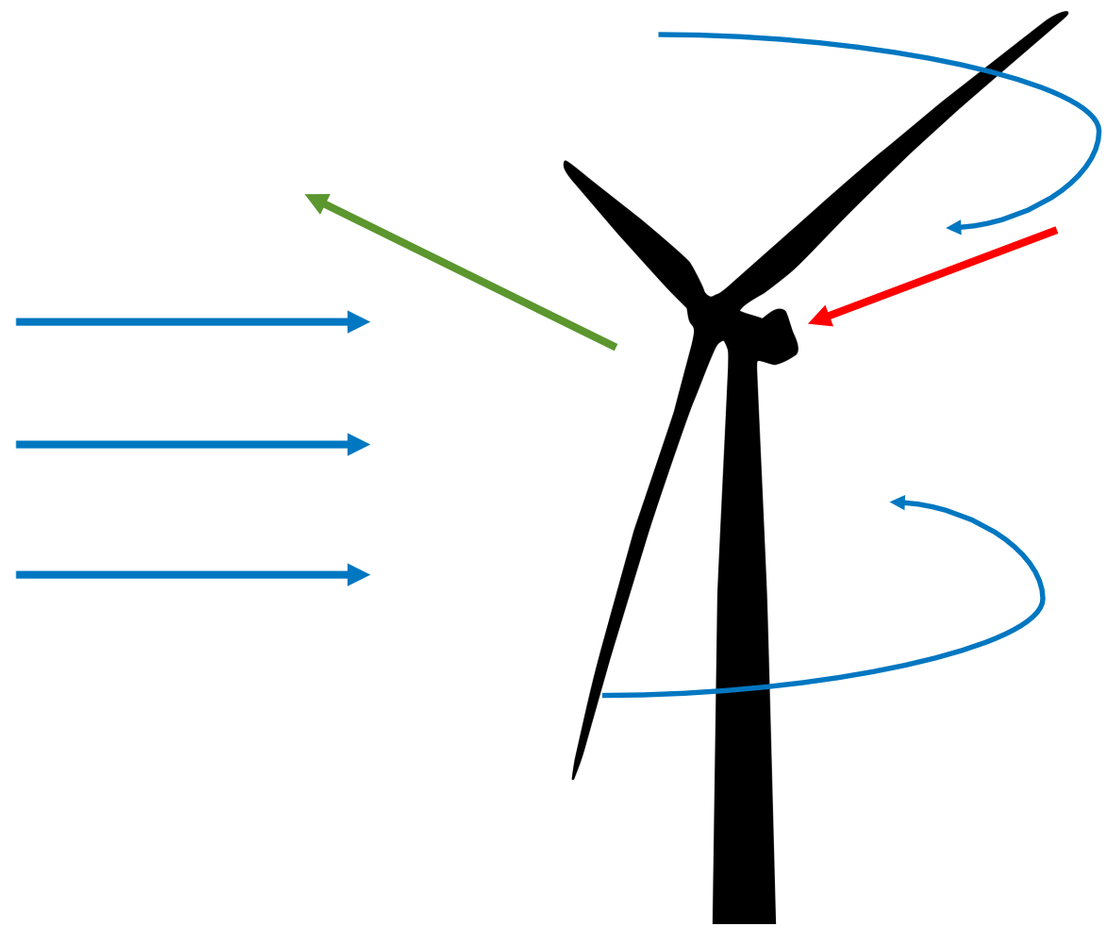
Why Non-Convexity ?

- Consider a wind turbine optimization problem (e.g., yaw angle) [Annoni et al. 19]

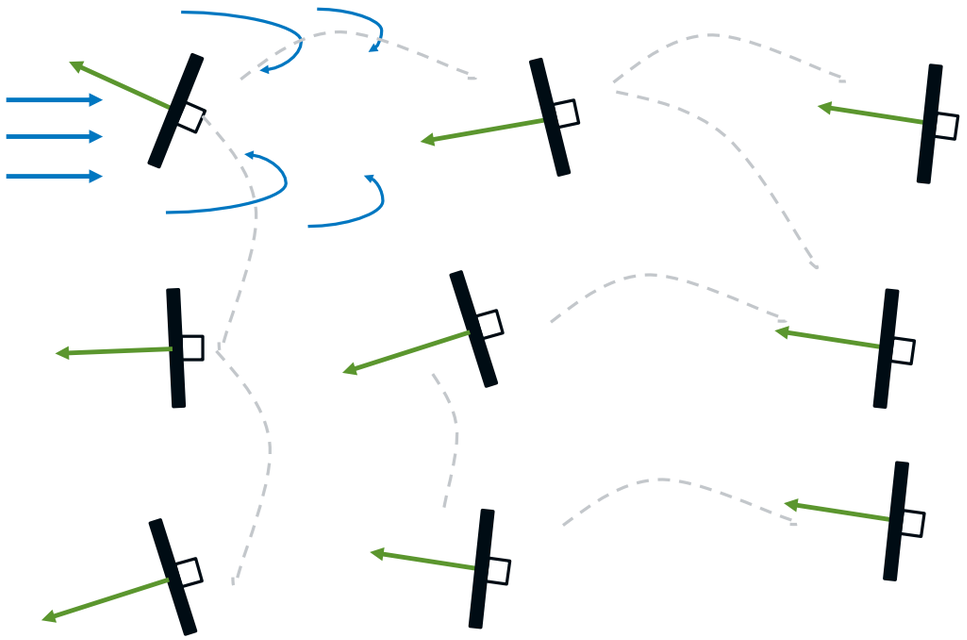
- Traditionally optimized individually

- Input output relationships highly non-trivial

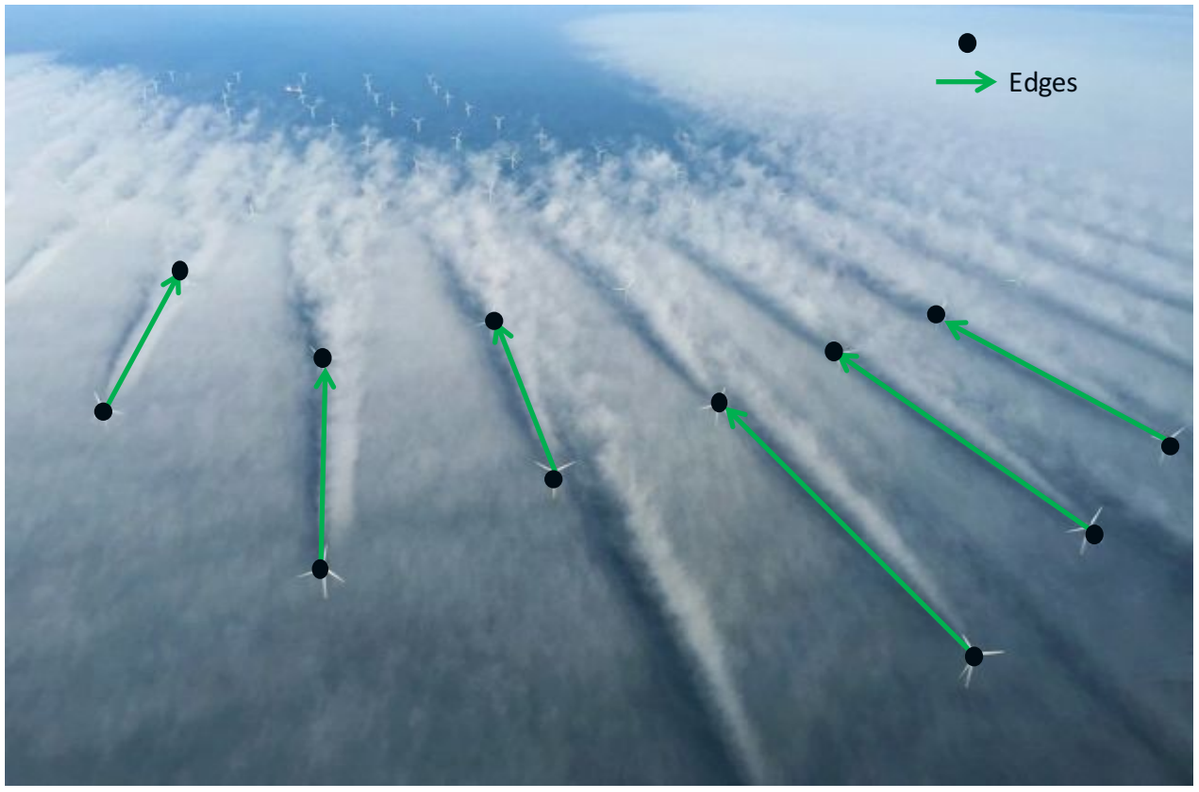
- Typically generated by a simulator



Why Non-Convexity ?

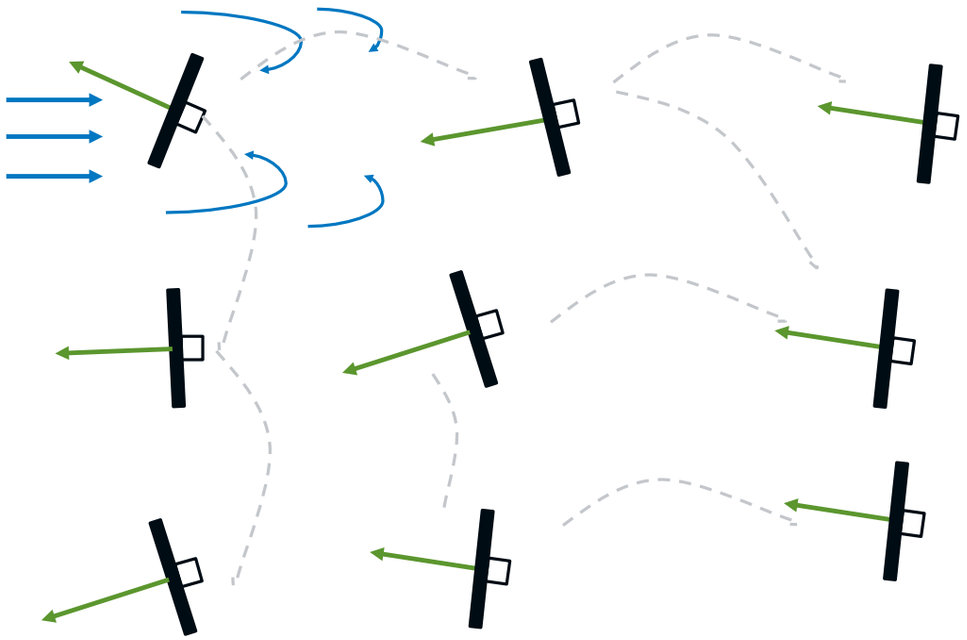


Wind turbines are collocated together

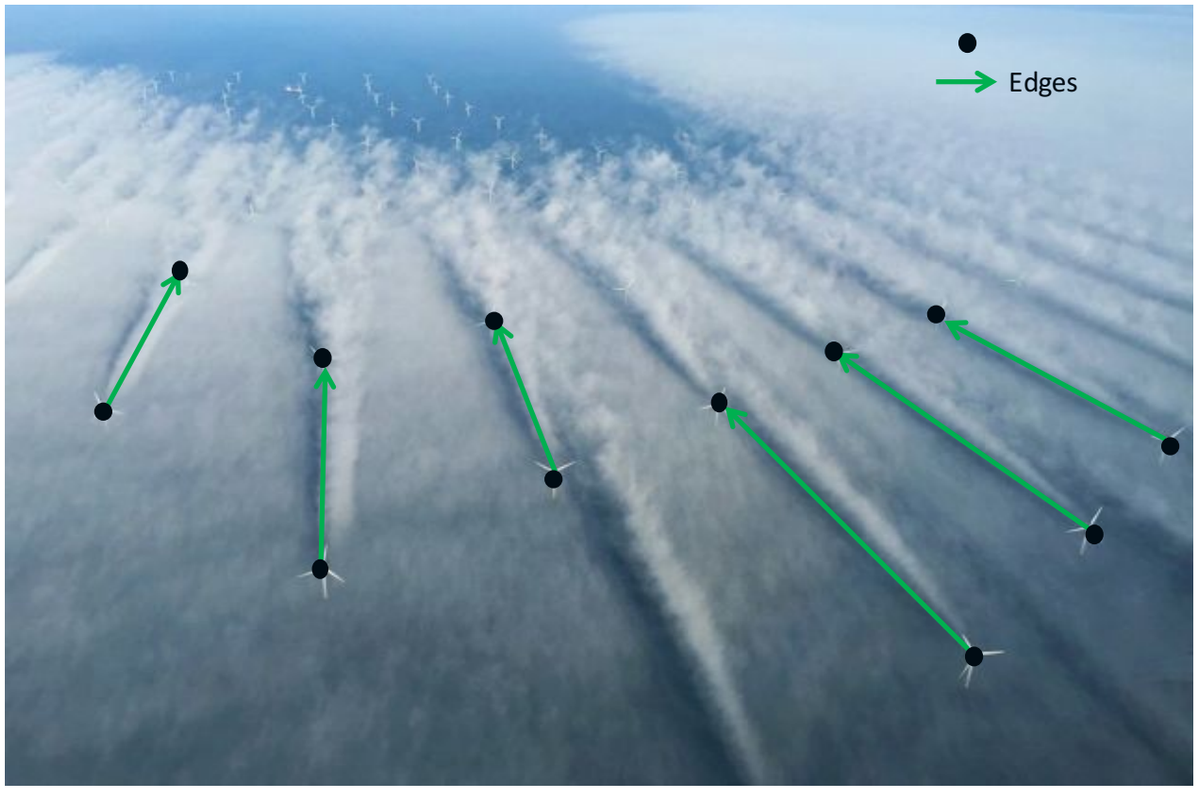


Individual optimization can be highly suboptimal

Why Non-Convexity ?



Wind turbines are collocated together



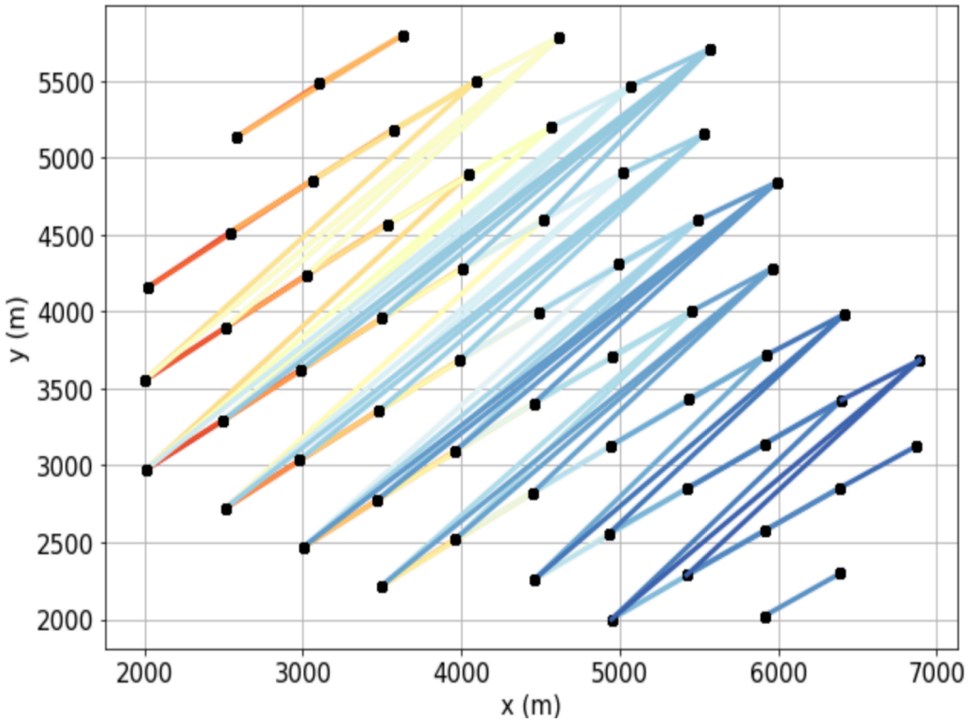
Individual optimization can be highly suboptimal

No coordination between the up- and down stream turbines can cause

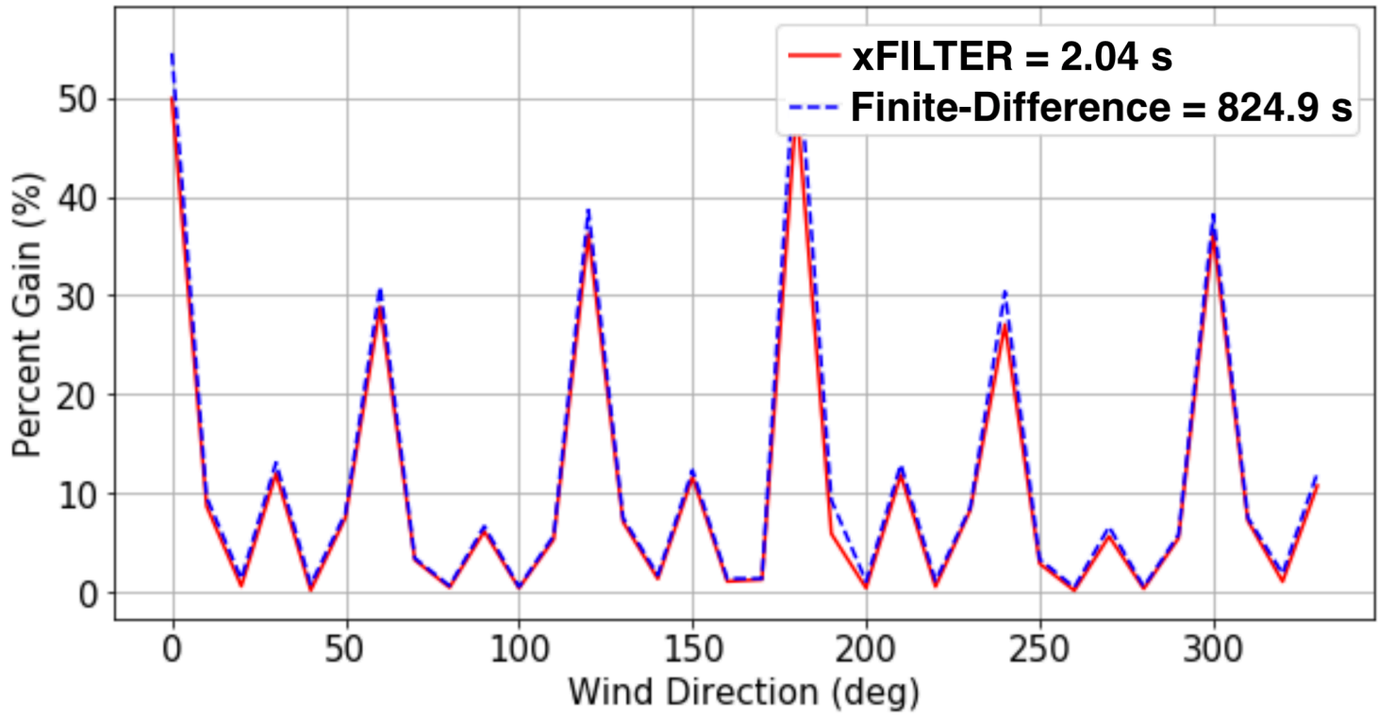
- Lose power - yaw misalignment
- Constantly yawing - noisy wind vane signal

Why Non-Convexity?

Decentralized non-convex optimization significantly improves the practical performance [Annoni et al. 19]



Graph structure of wind farm



100X faster than centralized method

- Princess Amalia Wind Farm [Fleming et al, 2016]
- Simulated 60 NREL's 5MW turbines [Jonkman et al, 2009]

Today's Talk

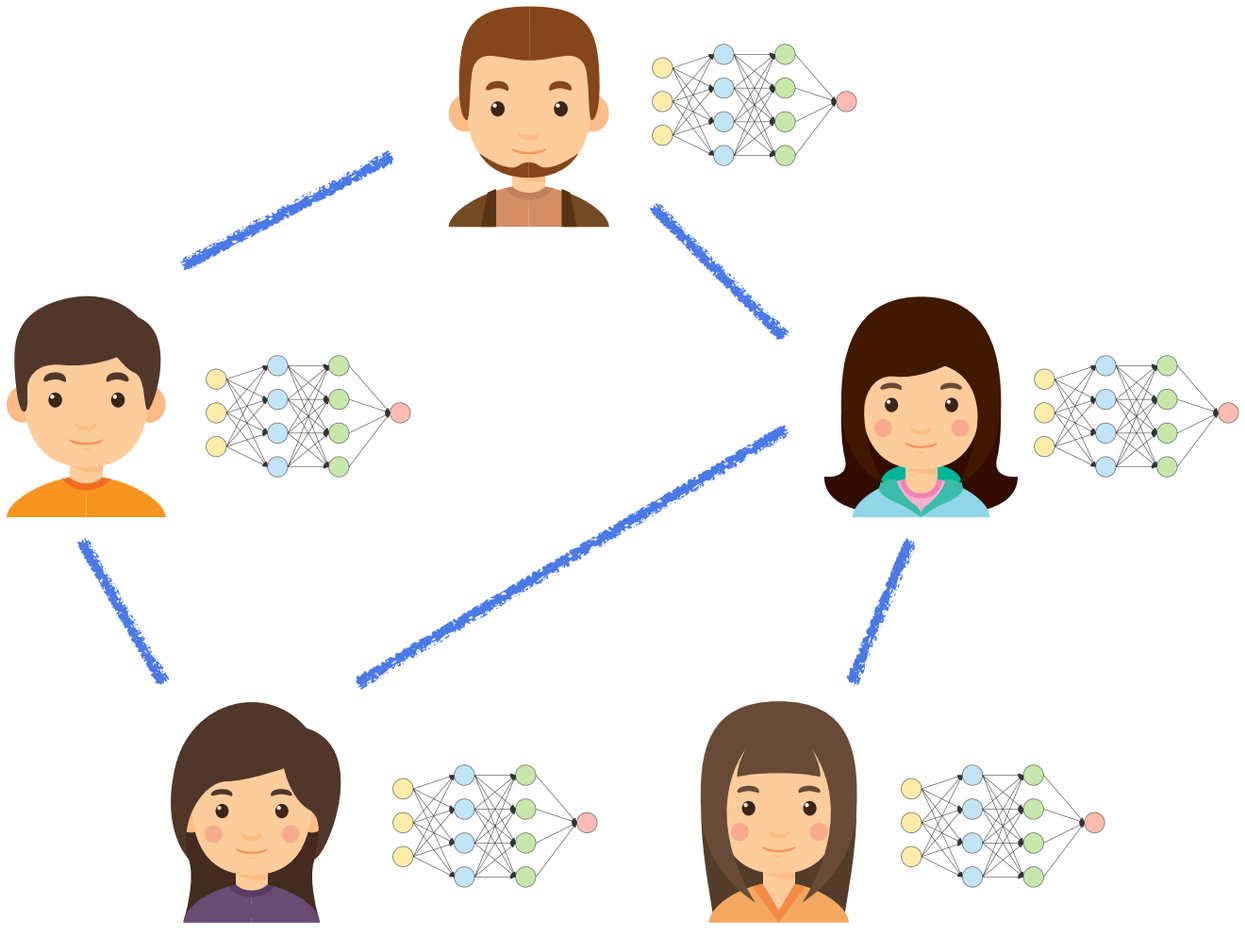
- Provide a high-level survey of recent advances distributed non-convex OPT and learning
- Given particular emphasis on the interaction between
  - Local data acquisition models
  - Local computation models
  - Local communication models
- Emphasizing on achieving certain "optimal" performance

The Problem Setting

Distributed Non-Convex Optimization

Consider a network consists of M agents, who collectively optimize

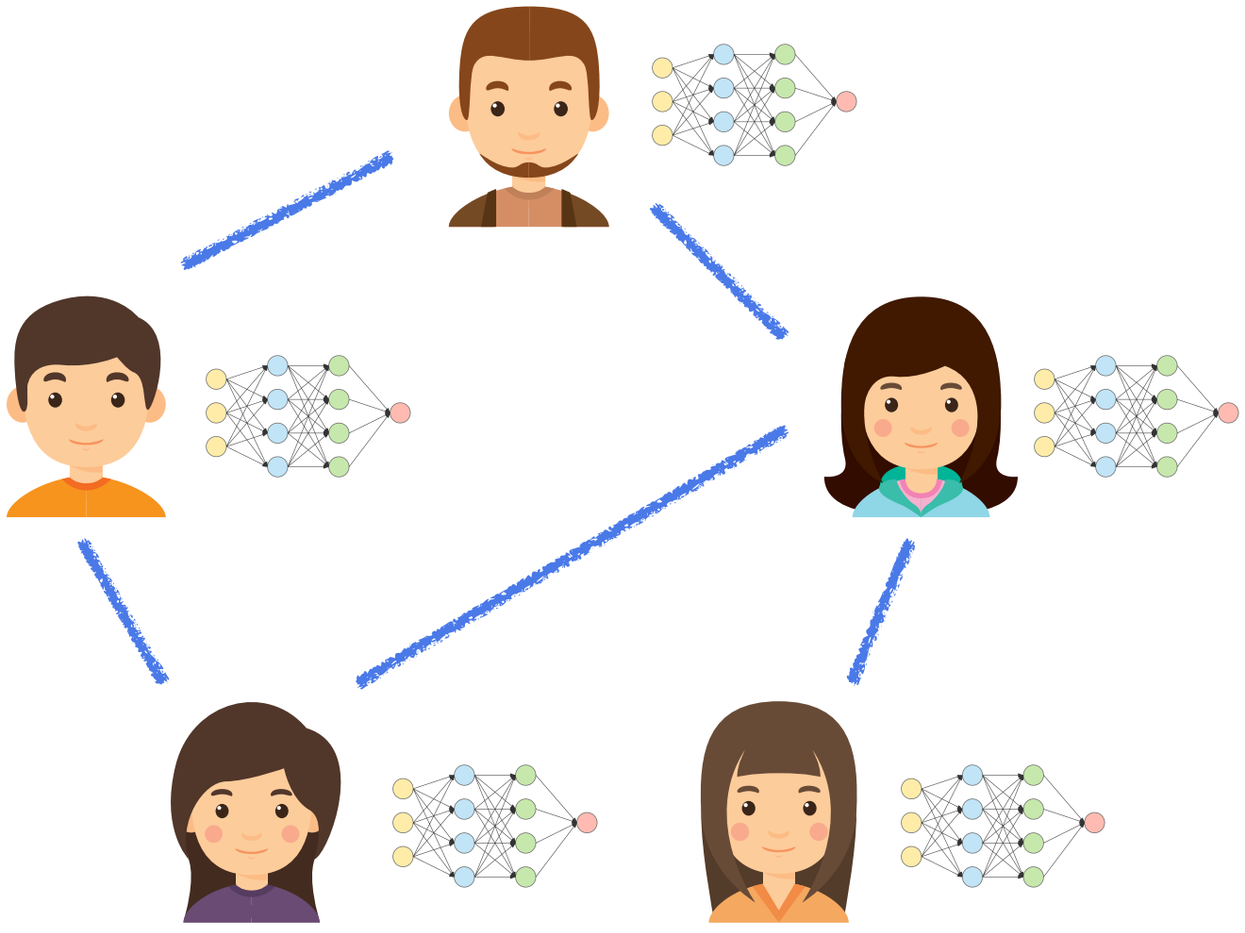
$$(\mathbf{P}) \min_{\mathbf{w}\in\mathbb{R}^P} h(\mathbf{w}) = \frac{1}{M}\sum_{i=1}^{M} f_i(\mathbf{w})$$



Distributed Non-Convex Optimization

Consider a network consists of M agents, who collectively optimize

$$(\mathbf{P}) \min_{\mathbf{w}\in\mathbb{R}^P} h(\mathbf{w}) = \frac{1}{M}\sum_{i=1}^{M} f_i(\mathbf{w})$$

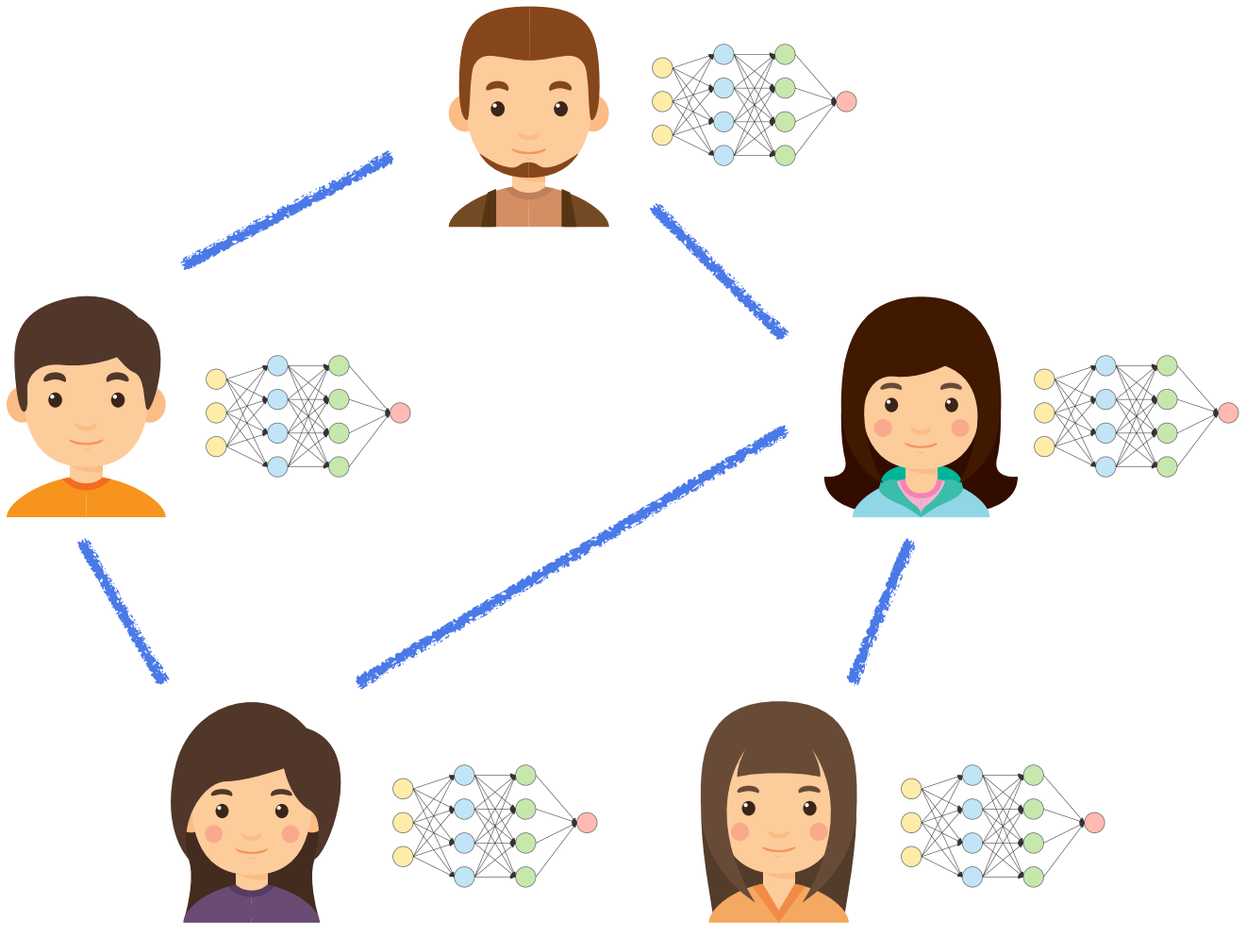


- $f_i(\cdot)$ is only known to agent *i*

Distributed Non-Convex Optimization

Consider a network consists of M agents, who collectively optimize

$$(\mathbf{P}) \min_{\mathbf{w}\in\mathbb{R}^P} h(\mathbf{w}) = \frac{1}{M}\sum_{i=1}^{M} f_i(\mathbf{w})$$

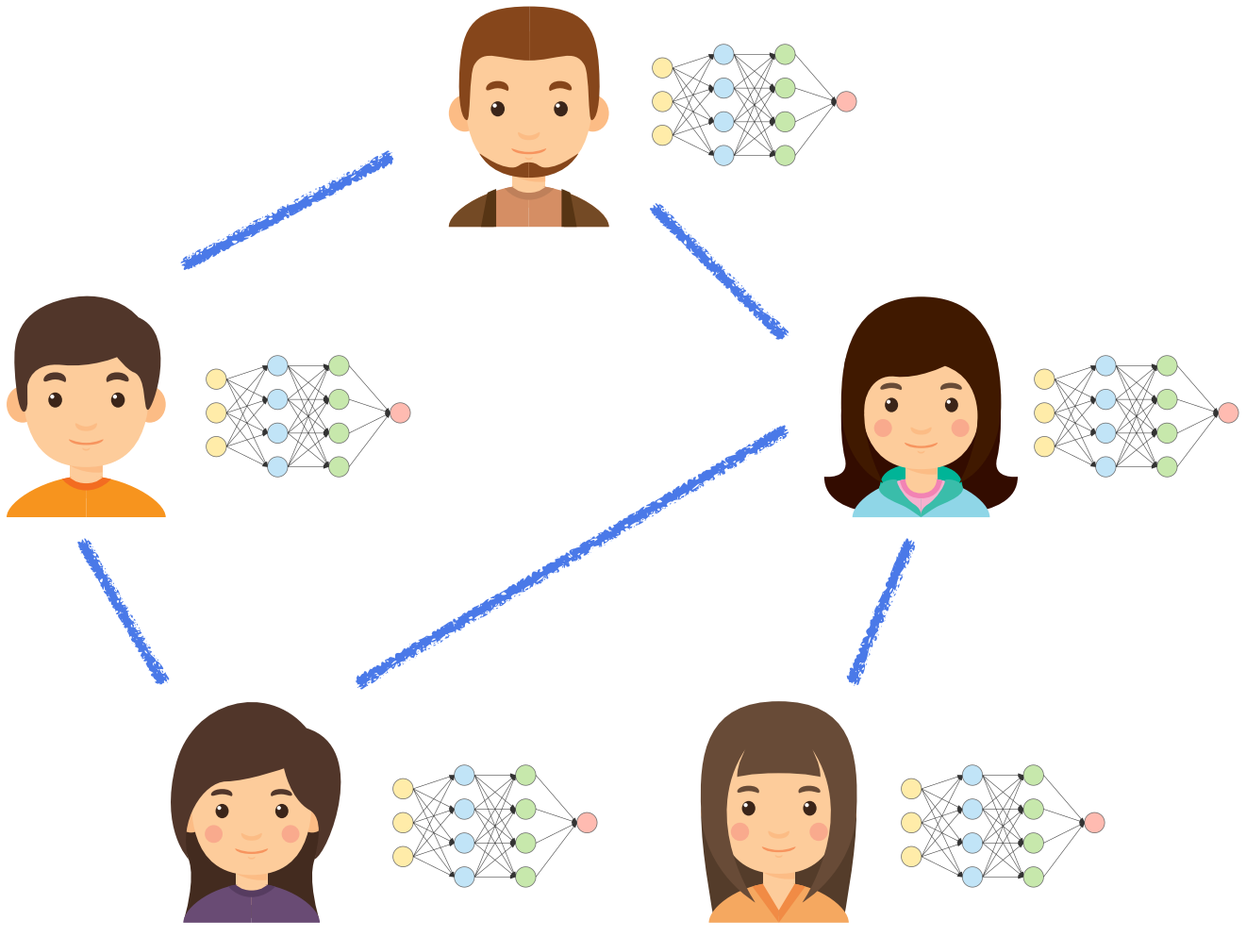


- $f_i(\cdot)$ is only known to agent *i*
- $f_i(\cdot)$ could be non-convex

Distributed Non-Convex Optimization

Consider a network consists of M agents, who collectively optimize

$$(\mathbf{P}) \min_{\mathbf{w}\in\mathbb{R}^P} h(\mathbf{w}) = \frac{1}{M}\sum_{i=1}^{M} f_i(\mathbf{w})$$



- $f_i(\cdot)$ is only known to agent *i*
- $f_i(\cdot)$ could be non-convex
- Agents are connected by an undirected and unweighted graph

$$\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$$

Distributed Non-Convex Optimization

Consider a network consists of M agents, who collectively optimize

$$(\mathbf{P}) \min_{\mathbf{w}\in\mathbb{R}^P} h(\mathbf{w}) = \frac{1}{M}\sum_{i=1}^{M} f_i(\mathbf{w})$$

Introduce M local variables and reformulate to the consensus problem

$$(\mathbf{Q}) \qquad \min_{\mathbf{x}\in\mathbb{R}^{MP}} \; f(\mathbf{x}) = \frac{1}{M}\sum_{i=1}^{M} f_i(\mathbf{x}_i), \quad \text{s.t. } \mathbf{A}\mathbf{x} = 0.$$

Distributed Non-Convex Optimization

Consider a network consists of M agents, who collectively optimize

$$(\mathbf{P}) \min_{\mathbf{w}\in\mathbb{R}^P} h(\mathbf{w}) = \frac{1}{M}\sum_{i=1}^{M} f_i(\mathbf{w})$$

Introduce M local variables and reformulate to the consensus problem

$$(\mathbf{Q}) \quad \min_{\mathbf{x}\in\mathbb{R}^{MP}} f(\mathbf{x}) = \frac{1}{M}\sum_{i=1}^{M} f_i(\mathbf{x}_i), \quad \text{s.t. } \mathbf{A}\mathbf{x} = 0.$$

- For simplicity assume local variables are scalars

Distributed Non-Convex Optimization

Consider a network consists of M agents, who collectively optimize

$$(\mathbf{P}) \min_{\mathbf{w}\in\mathbb{R}^P} h(\mathbf{w}) = \frac{1}{M}\sum_{i=1}^{M} f_i(\mathbf{w})$$

Introduce M local variables and reformulate to the consensus problem

$$(\mathbf{Q}) \quad \min_{\mathbf{x}\in\mathbb{R}^{MP}} f(\mathbf{x}) = \frac{1}{M}\sum_{i=1}^{M} f_i(\mathbf{x}_i), \quad \text{s.t. } \mathbf{A}\mathbf{x} = 0.$$

- For simplicity assume local variables are scalars
- Let $\mathbf{x} = [\mathbf{x}_1, \cdots, \mathbf{x}_M]^T$ collect all the variables

Distributed Non-Convex Optimization

Consider a network consists of M agents, who collectively optimize

$$(\mathbf{P}) \min_{\mathbf{w}\in\mathbb{R}^P} h(\mathbf{w}) = \frac{1}{M}\sum_{i=1}^{M} f_i(\mathbf{w})$$

Introduce M local variables and reformulate to the consensus problem

$$(\mathbf{Q}) \quad \min_{\mathbf{x}\in\mathbb{R}^{MP}} f(\mathbf{x}) = \frac{1}{M}\sum_{i=1}^{M} f_i(\mathbf{x}_i), \quad \text{s.t. } \mathbf{A}\mathbf{x} = 0.$$

- For simplicity assume local variables are scalars
- Let $\mathbf{x} = [\mathbf{x}_1, \cdots, \mathbf{x}_M]^T$ collect all the variables
- Incidence matrix $\mathbf{A} \in \mathbb{R}^{E\times M}$ encodes the network structure

Literature: Decentralized Convex OPT

- **Data model**: all local data is available at the same time
- **Computation model**: local gradients $\{\nabla f_i(x_i)\}$ can be computed
- **Communication model:** each node talks to its neighbors

Literature: Decentralized Convex OPT

Many algorithms are available for decentralized convex optimization

1. Decentralized averaging [Tsitsiklis 84]
2. Decentralized gradient descent (DGD) [Nedić-Ozdaglar 09ab]
3. The Alternating Direction Method of Multiplier (ADMM) based methods [Boyd et al 11] [Mota et al 12] [Giannakis et al 16] [Ma et al 18][Ma et al 19]
4. The Distributed Dual Averaging based methods [Duchi et al 12]
5. EXTRA algorithm [Shi et al 14]
6. Analysis on improving the convergence rates [Olshevsky-Tsitsiklis 09] [Scaman et al 17] [Qu et al 17] [Uribe et al 18][Xu et al 19]
7. Analysis on incorporating the variance reduction type techniques [Mokhtari-Ribeiro, 16] [Shen et al 18] [Yuan et al 18] [Hendrikx et al 19] [Wang et al 19] [Xin et al 19] [Li et al 19]
8. ...

Decentralized Gradient Descent (DGD)

$$\mathbf{x}^{r+1} = \mathbf{W}\mathbf{x}^{r} - \alpha \nabla f(\mathbf{x}^{r})$$

[1] Nedic, Angelia, and Asuman Ozdaglar. "Distributed subgradient methods for multi-agent optimization." *IEEE Transactions on Automatic Control* 54, no. 1 (2009): 48.
[2] Zeng, Jinshan, and Wotao Yin. "On nonconvex decentralized gradient descent." *IEEE Transactions on signal processing* 66, no. 11 (2018): 2834-2848.

Decentralized Gradient Descent (DGD)

$$\mathbf{x}^{r+1} = \mathbf{W}\mathbf{x}^r - \alpha \nabla f(\mathbf{x}^r)$$

$$\mathbf{x}_i^{r+1} = \sum_{j \in \mathcal{N}_i} \mathbf{W}_{ij}\mathbf{x}_j^r - \alpha \nabla f_i(\mathbf{x}_i^r)$$

[1] Nedic, Angelia, and Asuman Ozdaglar. "Distributed subgradient methods for multi-agent optimization." *IEEE Transactions on Automatic Control* 54, no. 1 (2009): 48.
[2] Zeng, Jinshan, and Wotao Yin. "On nonconvex decentralized gradient descent." *IEEE Transactions on signal processing* 66, no. 11 (2018): 2834-2848.

Decentralized Gradient Descent (DGD)

$$\mathbf{x}^{r+1} = \mathbf{W}\mathbf{x}^r - \alpha \nabla f(\mathbf{x}^r)$$

$$\mathbf{x}_i^{r+1} = \sum_{j \in \mathcal{N}_i} \mathbf{W}_{ij}\mathbf{x}_j^r - \alpha \nabla f_i(\mathbf{x}_i^r)$$

Consensus Step

[1] Nedic, Angelia, and Asuman Ozdaglar. "Distributed subgradient methods for multi-agent optimization." *IEEE Transactions on Automatic Control* 54, no. 1 (2009): 48.
[2] Zeng, Jinshan, and Wotao Yin. "On nonconvex decentralized gradient descent." *IEEE Transactions on signal processing* 66, no. 11 (2018): 2834-2848.

Decentralized Gradient Descent (DGD)

$$\mathbf{x}^{r+1} = \mathbf{W}\mathbf{x}^r - \alpha \nabla f(\mathbf{x}^r)$$

$$\mathbf{x}_i^{r+1} = \sum_{j \in \mathcal{N}_i} \mathbf{W}_{ij}\mathbf{x}_j^r - \alpha \nabla f_i(\mathbf{x}_i^r)$$

Consensus Step

Gradient Step

[1] Nedic, Angelia, and Asuman Ozdaglar. "Distributed subgradient methods for multi-agent optimization." *IEEE Transactions on Automatic Control* 54, no. 1 (2009): 48.
[2] Zeng, Jinshan, and Wotao Yin. "On nonconvex decentralized gradient descent." *IEEE Transactions on signal processing* 66, no. 11 (2018): 2834-2848.

Decentralized Gradient Descent (DGD)

$$\mathbf{x}^{r+1} = \mathbf{W}\mathbf{x}^r - \alpha \nabla f(\mathbf{x}^r)$$

$$\mathbf{x}_i^{r+1} = \sum_{j \in \mathcal{N}_i} \mathbf{W}_{ij}\mathbf{x}_j^r - \alpha \nabla f_i(\mathbf{x}_i^r)$$

Consensus Step

Gradient Step

- Where $\mathbf{W}\mathbf{1} = \mathbf{1},\ -\mathbf{I} \preceq \mathbf{W} \preceq \mathbf{I}$
- Constant stepsize converges to a neighborhood of OPT

[1] Nedic, Angelia, and Asuman Ozdaglar. "Distributed subgradient methods for multi-agent optimization." *IEEE Transactions on Automatic Control* 54, no. 1 (2009): 48.
[2] Zeng, Jinshan, and Wotao Yin. "On nonconvex decentralized gradient descent." *IEEE Transactions on signal processing* 66, no. 11 (2018): 2834-2848.

EXact firsT-ordeR Algorithm (EXTRA)

$$\mathbf{x}^1 = \mathbf{W}\mathbf{x}^0 - \alpha\nabla f(\mathbf{x}^0)$$

$$\mathbf{x}^{r+2} = (\mathbf{W}+\mathbf{I})\mathbf{x}^{r+1} - \widetilde{\mathbf{W}}\mathbf{x}^r - \alpha\left(\nabla f(\mathbf{x}^{r+1}) - \nabla f(\mathbf{x}^r)\right)$$

Consensus **Diminishing**

EXTRA as Corrected DGD

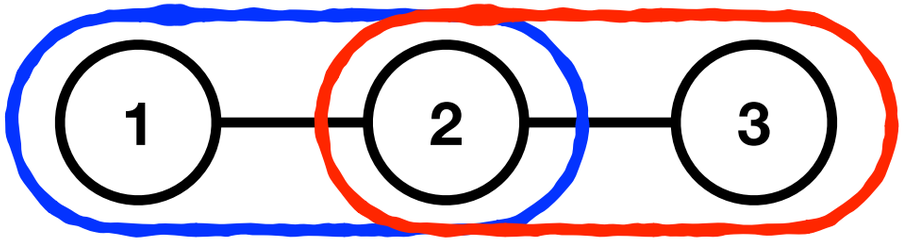
$$\mathbf{x}^{r+1} = \mathbf{W}\mathbf{x}^r - \alpha\nabla f(\mathbf{x}^r) + \sum_{t=0}^{r-1}(\mathbf{W}-\widetilde{\mathbf{W}})\mathbf{x}^t$$

DGD **Correction**

Alternating Direction Method of Multipliers (ADMM)

$$\min_{\mathbf{x},\mathbf{z}} \quad f(\mathbf{x}) + g(\mathbf{z}), \quad \text{s.t. } \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = 0.$$

matrix A, B encodes the network structure



$$\mathbf{x}_1 = \mathbf{z}_1 \quad \mathbf{x}_2 = \mathbf{z}_2$$

$$\mathbf{x}_2 = \mathbf{z}_1 \quad \mathbf{x}_3 = \mathbf{z}_2$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \end{bmatrix} + \begin{bmatrix} -1 & 0 \\ -1 & 0 \\ 0 & -1 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} \mathbf{z}_1 \\ \mathbf{z}_2 \end{bmatrix} = 0$$

A **x** **B** **z**

Alternating Direction Method of Multipliers (ADMM)

$$\min_{\mathbf{x},\mathbf{z}} \quad f(\mathbf{x}) + g(\mathbf{z}), \quad \text{s.t. } \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} = 0.$$

Augmented Lagrangian

$$L(\mathbf{x}, \mathbf{z}, \lambda) = f(\mathbf{x}) + \langle \lambda, \mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z} \rangle + \frac{c}{2} \|\mathbf{A}\mathbf{x} + \mathbf{B}\mathbf{z}\|^2$$

Alternating Updates

$$\mathbf{x} \text{ update: } \nabla f(\mathbf{x}^{r+1}) + \mathbf{A}^T \lambda^r + c\mathbf{A}^T(\mathbf{A}\mathbf{x}^{r+1} + \mathbf{B}\mathbf{z}^r) = 0$$

$$\mathbf{z} \text{ update: } \mathbf{B}^T \lambda^r + c\mathbf{B}^T(\mathbf{A}\mathbf{x}^{r+1} + \mathbf{B}\mathbf{z}^{r+1}) = 0$$

$$\lambda \text{ update: } \lambda^{r+1} - \lambda^r - c(\mathbf{A}\mathbf{x}^{r+1} + \mathbf{B}\mathbf{z}^{r+1}) = 0$$

Can we simply apply these existing methods to non-convex problems?

DGD work for Non-Convex Problems?

Let us apply the DGD algorithm (with $\mathbf{W} = 1/2 \times [1\ 1; 1\ 1]$, and a *constant* stepsize α) to a two-node problem **(P)**, where

$$f_1 = (\mathbf{x}_1)^2,\ f_2 = -(\mathbf{x}_2)^2.$$

The iteration becomes

$$\mathbf{x}^{r+1} = \mathbf{V}\mathbf{x}^r,\ \text{ where }\ \mathbf{V} := \begin{bmatrix} 1/2 - 2\alpha & 1/2 \\ 1/2 & 1/2 + 2\alpha \end{bmatrix}.$$

- Non-convex objective, but this problem just seeks to find average
- $|\mathbf{x}_1 - \mathbf{x}_2|$ produced by DGD diverges regardless of the choice of α

Literature: Decentralized Non-Convex OPT

- Literature in decentralized non-convex optimization is relatively scant
- Early works focused on getting a convergent algorithm

Literature: Decentralized Non-Convex OPT

- Literature in decentralized non-convex optimization is relatively scant
- Early works focused on getting a convergent algorithm
  - Dual subgradient method [Zhu et al., 13]
    - Limitation: Relaxes the consensus, no global rate analysis

Literature: Decentralized Non-Convex OPT

- Literature in decentralized non-convex optimization is relatively scant
- Early works focused on getting a convergent algorithm
    - Dual subgradient method [Zhu et al., 13]
        - Limitation: Relaxes the consensus, no global rate analysis
    - ADMM [H. -Luo-Razaviyayn 14]
        - Limitation: Only deals with networks with a central controller (parameter server setting)

Literature: Decentralized Non-Convex OPT

- Literature in decentralized non-convex optimization is relatively scant
- Early works focused on getting a convergent algorithm
  - Dual subgradient method [Zhu et al., 13]
    - Limitation: Relaxes the consensus, no global rate analysis
  - ADMM [H. -Luo-Razaviyayn 14]
    - Limitation: Only deals with networks with a central controller (parameter server setting)
  - NEXT [Di Lorezo-Scutari, 16]
    - Asymptotically converges to stationary solutions
    - Can deal with non-smooth/constrained problems

Literature: Decentralized Non-Convex OPT

- Recent works focused on analyzing convergence rates
  - Primal-Dual Based Methods [H. et al., 16]
    - Global sublinear rate O($1/\epsilon$) for any connected network
  - Gradient-Tracking based Methods [Sun-Scutari 18]
    - Global sublinear rate O($1/\epsilon$), for occasionally connected network
  - DGD Method [Zeng-Yin, 2019] [Jiang et al 18]
    - Uses diminishing step-size, relatively slow rate O($1/\epsilon^2$)

Primal-Dual Based Algorithms

Consider the consensus problem

$$(\mathbf{Q}) \qquad \min_{\mathbf{x}\in\mathbb{R}^{MP}} \; f(\mathbf{x}) = \frac{1}{M}\sum_{i=1}^{M} f_i(\mathbf{x}_i), \quad \text{s.t. } \mathbf{A}\mathbf{x} = 0.$$

[1] Hong, Mingyi, Davood Hajinezhad, and Ming-Min Zhao. "Prox-PDA: The proximal primal-dual algorithm for fast distributed nonconvex optimization and learning over networks." In *ICML* 2017.

Primal-Dual Based Algorithms

Consider the consensus problem

$$(\mathbf{Q}) \qquad \min_{\mathbf{x}\in\mathbb{R}^{MP}} \; f(\mathbf{x}) = \frac{1}{M}\sum_{i=1}^{M} f_i(\mathbf{x}_i), \quad \text{s.t. } \mathbf{A}\mathbf{x} = 0.$$

Augmented Lagrangian

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \langle \lambda, \mathbf{A}\mathbf{x}\rangle + \frac{\rho}{2}\|\mathbf{A}\mathbf{x}\|^2$$

[1] Hong, Mingyi, Davood Hajinezhad, and Ming-Min Zhao. "Prox-PDA: The proximal primal-dual algorithm for fast distributed nonconvex optimization and learning over networks." In *ICML* 2017.

Primal-Dual Based Algorithms

Consider the consensus problem

$$(\mathbf{Q}) \qquad \min_{\mathbf{x}\in\mathbb{R}^{MP}} \; f(\mathbf{x}) = \frac{1}{M}\sum_{i=1}^{M} f_i(\mathbf{x}_i), \quad \text{s.t. } \mathbf{A}\mathbf{x} = 0.$$

Augmented Lagrangian

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \langle \lambda, \mathbf{A}\mathbf{x}\rangle + \frac{\rho}{2}\|\mathbf{A}\mathbf{x}\|^2$$

Algorithm Updates [H.-Hajnezhad-Zhao 16]

$$\mathbf{x}^{r+1} = \arg\min_{\mathbf{x}} \underbrace{\langle \nabla f(\mathbf{x}^r) + \mathbf{A}^T\lambda^r + \rho\mathbf{A}^T\mathbf{A}\mathbf{x}^r, \mathbf{x} - \mathbf{x}^r\rangle}_{\text{linearizes the entire AL function.}} + \underbrace{\frac{\beta}{2}\|\mathbf{x} - \mathbf{x}^r\|^2 + \frac{\rho\lambda_{\max}(\mathbf{A}^T\mathbf{A})}{2}\|\mathbf{x} - \mathbf{x}^r\|^2}_{\text{proximal terms}}$$

$$\lambda^{r+1} = \lambda^r + \rho\mathbf{A}\mathbf{x}^{r+1}$$

[1] Hong, Mingyi, Davood Hajinezhad, and Ming-Min Zhao. "Prox-PDA: The proximal primal-dual algorithm for fast distributed nonconvex optimization and learning over networks." In *ICML* 2017.

Primal-Dual Based Algorithms

- **Comment**: This algorithm is slightly different than the one we discussed in Lecture 6:
  - Because here the gradient of f is used in the update;
  - So the primal step is reminiscent to a GD step
  - While in Lecture 6, we need to solve a convex optimization problem

Primal-Dual Based Algorithms

- After some manipulation, we have the following equivalent form

$$\mathbf{x}^{r+1} = (\mathbf{I} - \alpha \mathbf{A}^T \mathbf{A})(2\mathbf{x}^r - \mathbf{x}^{r-1}) - \alpha(\nabla f(\mathbf{x}^r) - \nabla f(\mathbf{x}^{r-1}))$$

- Updates depends on the previous two iterations

Relationship to other algorithms

- The primal-dual iteration is also equivalent to EXTRA [Shi et al., 14]

$$\mathbf{x}^{r+1} = (\mathbf{I}+\mathbf{W})\mathbf{x}^r - \frac{1}{2}(\mathbf{I}+\mathbf{W})\mathbf{x}^{r-1} - \alpha(\nabla f(\mathbf{x}^r) - \nabla f(\mathbf{x}^{r-1}))$$

if we choose $\mathbf{W} = \mathbf{I} - \alpha\mathbf{A}^T\mathbf{A}$

- Further, Setting $\lambda \equiv 0$ the primal-dual algorithm becomes

$$\mathbf{x}^{r+1} = \underbrace{(\mathbf{I} - \alpha\mathbf{A}^T\mathbf{A})}_{:=\mathbf{W}}\mathbf{x}^r - \alpha\nabla f(\mathbf{x}^r)$$

- Becomes DGD algorithm; no dual update —-> slow convergence

Decentralized Gradient Tracking (GT)

Introduce an auxiliary variable to track the global information [Sun-Scutari 18]

$$\mathbf{x}^{r+1} = \mathbf{W}\mathbf{x}^r - \alpha\mathbf{y}^r$$

$$\mathbf{y}^{r+1} = \mathbf{W}\mathbf{y}^r + \nabla f(\mathbf{x}^{r+1}) - \nabla f(\mathbf{x}^r)$$

[1] Zhu, Minghui, and Sonia Martínez. "Discrete-time dynamic average consensus." *Automatica* 46, no. 2 (2010): 322-329.
[2] Nedic, Angelia, Alex Olshevsky, and Wei Shi. "Achieving geometric convergence for distributed optimization over time-varying graphs." *SIAM Journal on Optimization* 27, no. 4 (2017): 2597-2633.
[3] Di Lorenzo, Paolo, and Gesualdo Scutari. "Next: In-network nonconvex optimization." *IEEE Transactions on Signal and Information Processing over Networks* 2, no. 2 (2016): 120-136.

Decentralized Gradient Tracking (GT)

Introduce an auxiliary variable to track the global information [Sun-Scutari 18]

$$\mathbf{x}^{r+1} = \mathbf{W}\mathbf{x}^r - \alpha\mathbf{y}^r$$
$$\mathbf{y}^{r+1} = \mathbf{W}\mathbf{y}^r + \nabla f(\mathbf{x}^{r+1}) - \nabla f(\mathbf{x}^r)$$

Global Gradient Estimation

[1] Zhu, Minghui, and Sonia Martínez. "Discrete-time dynamic average consensus." *Automatica* 46, no. 2 (2010): 322-329.
[2] Nedic, Angelia, Alex Olshevsky, and Wei Shi. "Achieving geometric convergence for distributed optimization over time-varying graphs." *SIAM Journal on Optimization* 27, no. 4 (2017): 2597-2633.
[3] Di Lorenzo, Paolo, and Gesualdo Scutari. "Next: In-network nonconvex optimization." *IEEE Transactions on Signal and Information Processing over Networks* 2, no. 2 (2016): 120-136.

Decentralized Gradient Tracking (GT)

Introduce an auxiliary variable to track the global information [Sun-Scutari 18]

$$\mathbf{x}^{r+1} = \mathbf{W}\mathbf{x}^r - \alpha\mathbf{y}^r$$

$$\mathbf{y}^{r+1} = \mathbf{W}\mathbf{y}^r + \nabla f(\mathbf{x}^{r+1}) - \nabla f(\mathbf{x}^r)$$

[1] Zhu, Minghui, and Sonia Martínez. "Discrete-time dynamic average consensus." *Automatica* 46, no. 2 (2010): 322-329.
[2] Nedic, Angelia, Alex Olshevsky, and Wei Shi. "Achieving geometric convergence for distributed optimization over time-varying graphs." *SIAM Journal on Optimization* 27, no. 4 (2017): 2597-2633.
[3] Di Lorenzo, Paolo, and Gesualdo Scutari. "Next: In-network nonconvex optimization." *IEEE Transactions on Signal and Information Processing over Networks* 2, no. 2 (2016): 120-136.

Decentralized Gradient Tracking (GT)

Introduce an auxiliary variable to track the global information [Sun-Scutari 18]

$$\mathbf{x}^{r+1} = \mathbf{W}\mathbf{x}^r - \alpha\mathbf{y}^r$$

$$\mathbf{y}^{r+1} = \mathbf{W}\mathbf{y}^r + \nabla f(\mathbf{x}^{r+1}) - \nabla f(\mathbf{x}^r)$$

Track the global information

[1] Zhu, Minghui, and Sonia Martínez. "Discrete-time dynamic average consensus." *Automatica* 46, no. 2 (2010): 322-329.
[2] Nedic, Angelia, Alex Olshevsky, and Wei Shi. "Achieving geometric convergence for distributed optimization over time-varying graphs." *SIAM Journal on Optimization* 27, no. 4 (2017): 2597-2633.
[3] Di Lorenzo, Paolo, and Gesualdo Scutari. "Next: In-network nonconvex optimization." *IEEE Transactions on Signal and Information Processing over Networks* 2, no. 2 (2016): 120-136.

Decentralized Gradient Tracking (GT)

Introduce an auxiliary variable to track the global information [Sun-Scutari 18]

$$\mathbf{x}^{r+1} = \mathbf{W}\mathbf{x}^r - \alpha\mathbf{y}^r$$

$$\mathbf{y}^{r+1} = \mathbf{W}\mathbf{y}^r + \nabla f(\mathbf{x}^{r+1}) - \nabla f(\mathbf{x}^r)$$

Track the global information

Update the new local information

[1] Zhu, Minghui, and Sonia Martínez. "Discrete-time dynamic average consensus." *Automatica* 46, no. 2 (2010): 322-329.
[2] Nedic, Angelia, Alex Olshevsky, and Wei Shi. "Achieving geometric convergence for distributed optimization over time-varying graphs." *SIAM Journal on Optimization* 27, no. 4 (2017): 2597-2633.
[3] Di Lorenzo, Paolo, and Gesualdo Scutari. "Next: In-network nonconvex optimization." *IEEE Transactions on Signal and Information Processing over Networks* 2, no. 2 (2016): 120-136.

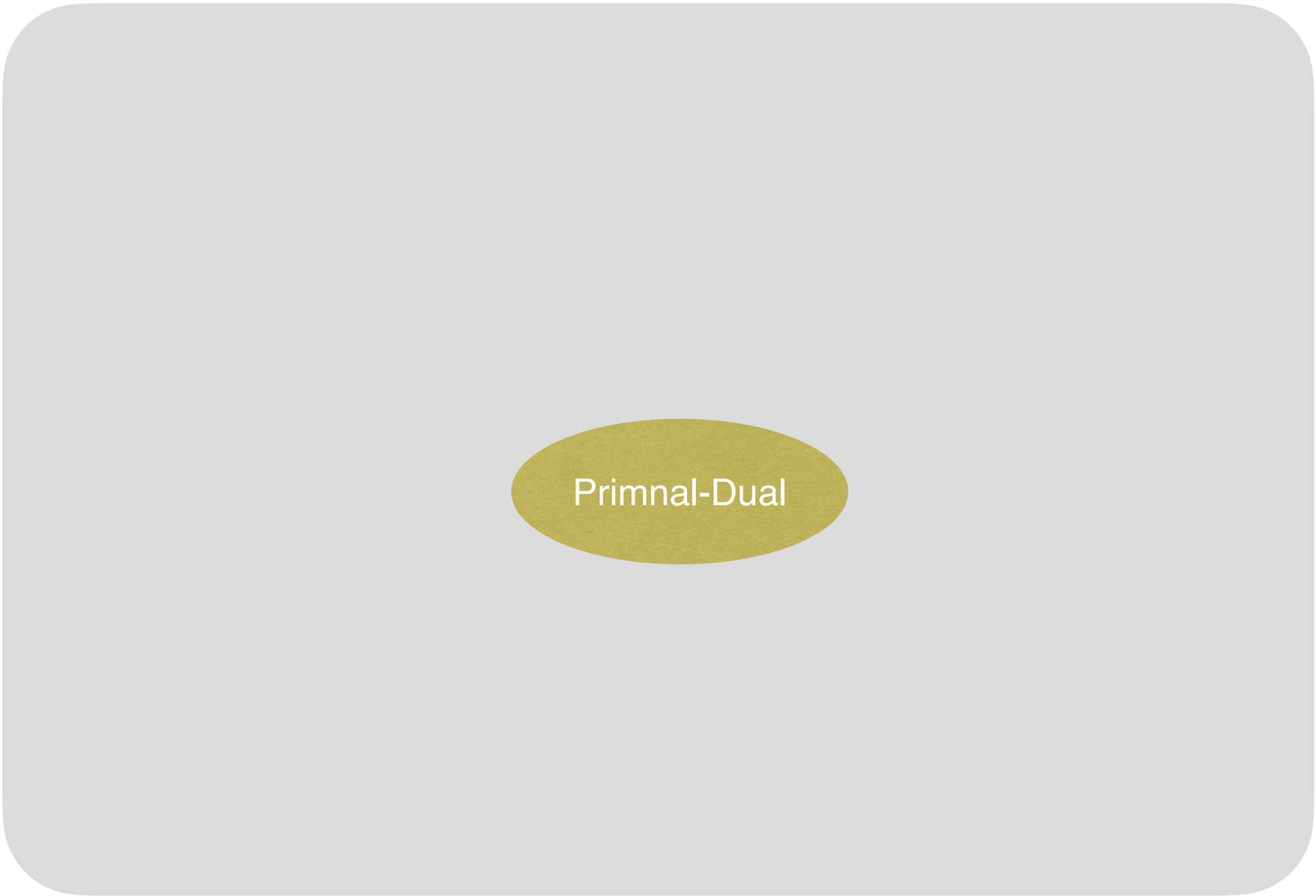
Decentralized GT

- After some manipulation, we have the following equivalent form

$$\mathbf{x}^{r+1} = 2\mathbf{W}\mathbf{x}^r - \mathbf{W}^2\mathbf{x}^{r-1} - \alpha(\nabla f(\mathbf{x}^r) - \nabla f(\mathbf{x}^{r-1}))$$

- When certain conditions on W is satisfied, this algorithm is equivalent to EXTRA

Relations Between Algorithms



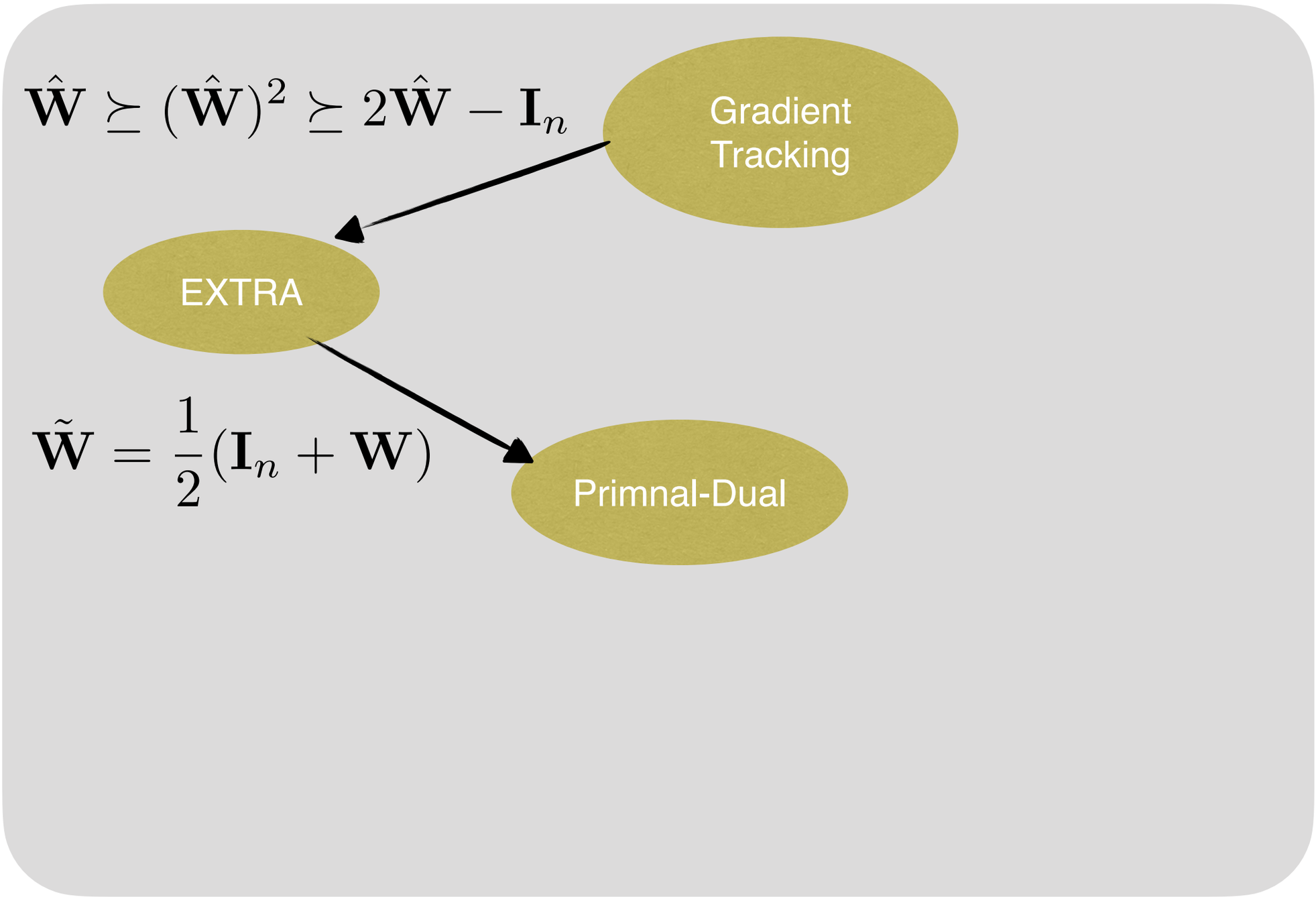
Relations Between Algorithms

EXTRA

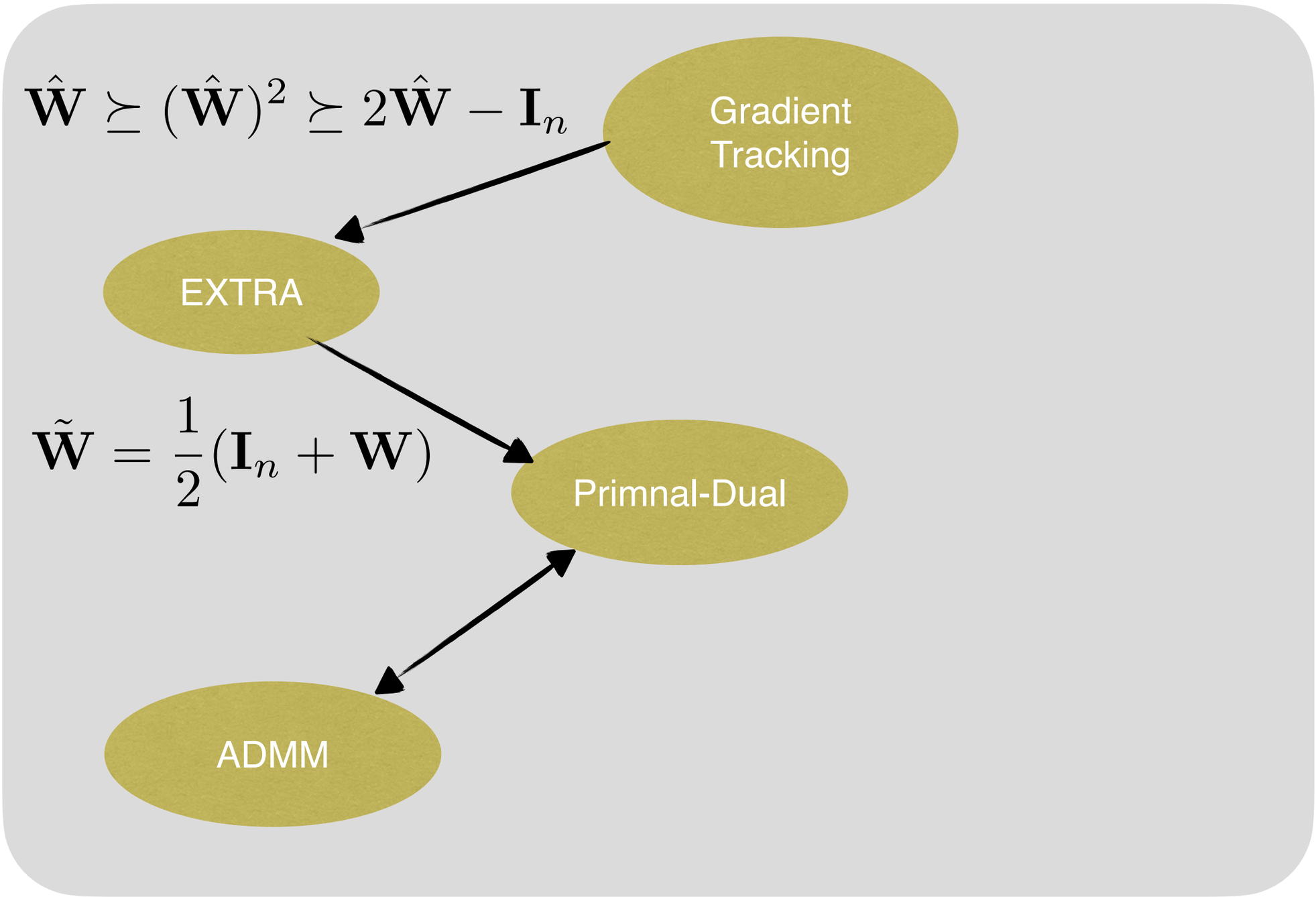
$$\tilde{\mathbf{W}} = \frac{1}{2}(\mathbf{I}_n + \mathbf{W})$$

Primnal-Dual

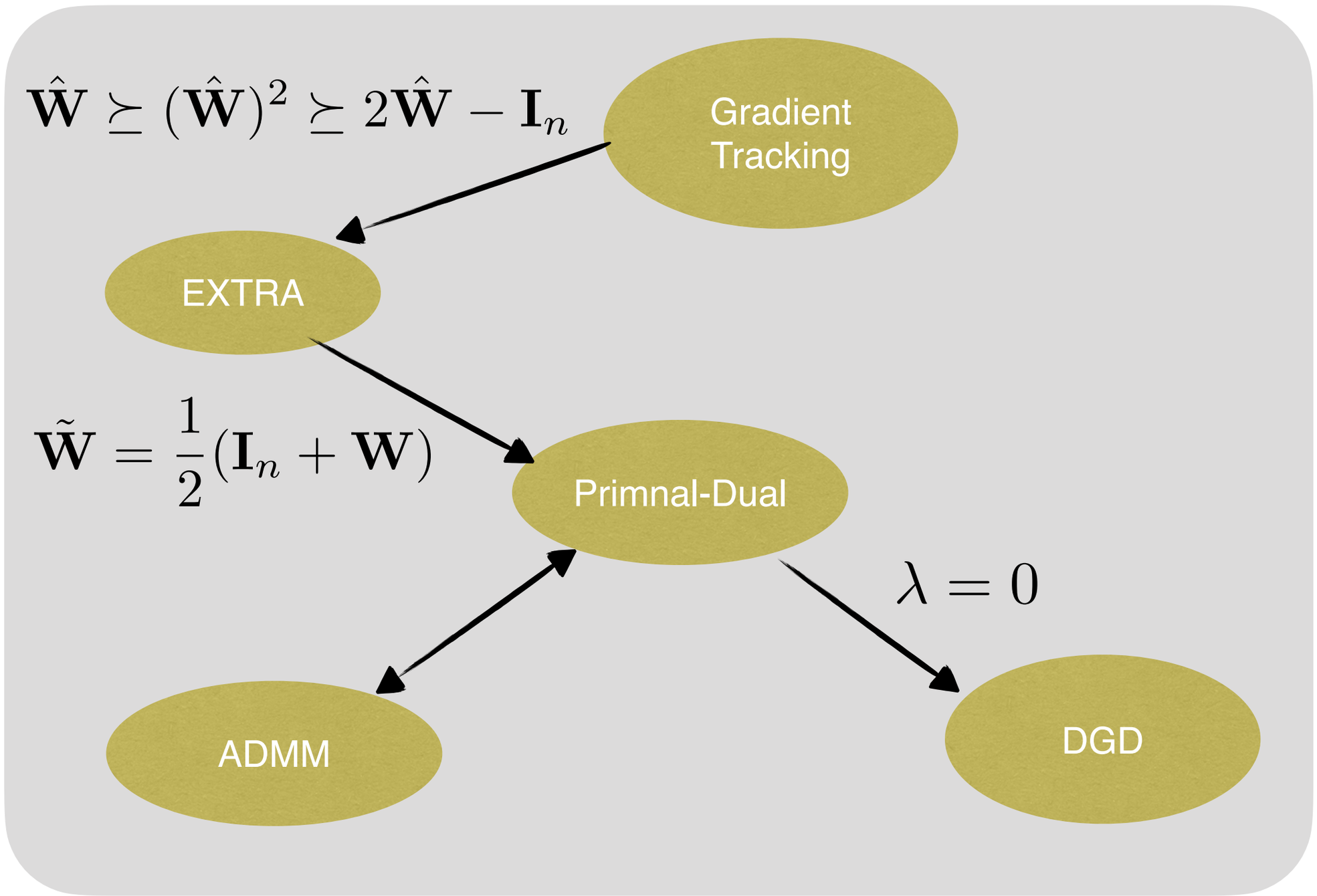
Relations Between Algorithms



Relations Between Algorithms



Relations Between Algorithms



Challenge

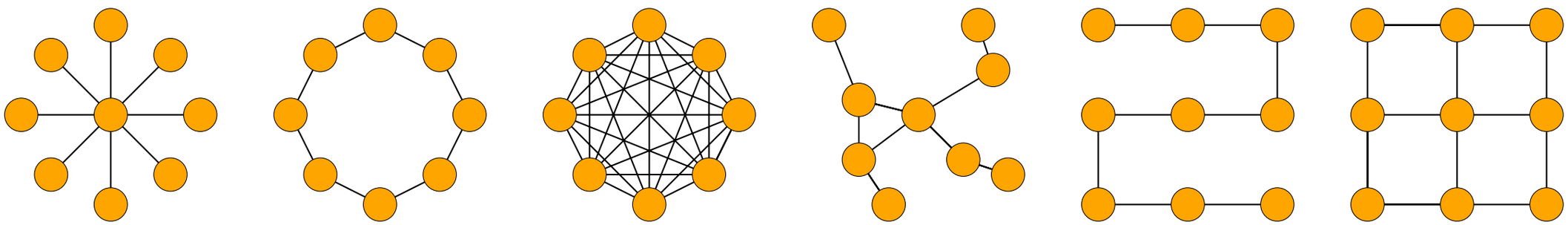
GT/Primal-Dual Methods converges “sublinearly”

— Requires O($1/\epsilon$) “iteration” to compute some ϵ - solutions

[H.-Hajnezhad-Zhao 16 16][Sun-Scutari 18]

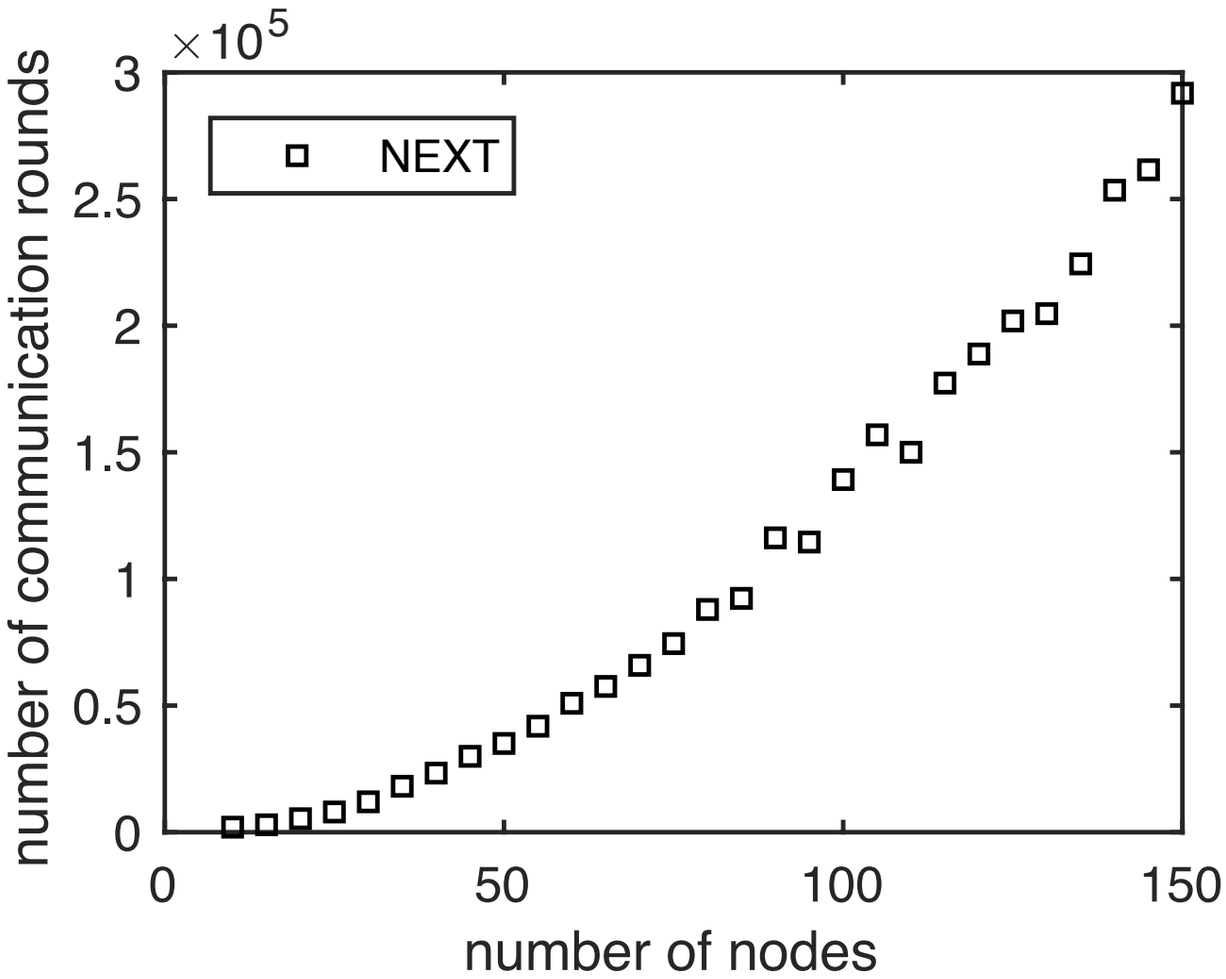
— Big O hides many factors

Many important factors are ignored for decentralized optimization



Network dependency can be very bad

Challenge



Communication rounds required to reach e-10 on decentralized binary classification problems over path graph

Convergence rate could slow down very quickly for large networks

“Rate Optimal” Algorithms?

Can we identify “rate optimal” algorithms?

- What is the best possible network dependency achievable by any decentralized algorithms?

- And how to achieve such optimal dependency?

“Rate Optimal” Algorithms?

Given a network structure, what is the best possible network dependency achievable by any decentralized algorithms? And how to achieve those dependency?

Key components (specifics needed):

★ For which kind of problems?

★ Over which kind of networks?

★ Using what kind of algorithms?

★ To achieve what kind of solution quality?

Lower Complexity Bounds

Problem Class

Problem Class $\mathcal{P}_L^M$: Consider the following decentralized problem

$$\min_{\mathbf{x}\in\mathbb{R}^{MP}} \quad f(\mathbf{x}) = \frac{1}{M}\sum_{i=1}^{M} f_i(\mathbf{x}_i), \quad \text{s.t.} \quad \mathbf{x}_i = \mathbf{x}_k, \quad \forall (i,k) \in \mathcal{E}.$$

A1. The objective is a sum of M functions (one for each node);

A2. Each component function $f_i(\mathbf{x})$'s has Lipschitz gradient:

$$\|\nabla f_i(\mathbf{x}_i) - \nabla f_i(\mathbf{z}_i)\| \leq L_i \|\mathbf{x}_i - \mathbf{z}_i\|, \ \forall \ \mathbf{x}_i, \mathbf{z}_i \in \mathbb{R}^P, \ \forall \ i.$$

A3. The function $f(\mathbf{x})$ is lower bounded:

$$\min_{x\in\mathbb{R}^{P\times M}} f(\mathbf{x}) \geq \underline{f}$$

Network Class

- **Network Class** $\mathcal{N}_M$: Undirected and unweighted graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with $|\mathcal{V}| = M$ vertices and $|\mathcal{E}| = E$ edges; each node has degree d_i;

Network Class

- **Network Class** $\mathcal{N}_M$: Undirected and unweighted graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ with $|\mathcal{V}| = M$ vertices and $|\mathcal{E}| = E$ edges; each node has degree d_i;

- Some quantities related to graph

  – Normalized Graph Laplacian (NL)

$$\mathcal{L} := P^{-1/2} A^T A P^{-1/2}, \text{ with } P = \text{diag}[d_1, \cdots, d_M]$$

$$[\mathcal{L}]_{ij} = \begin{cases} 1 & \text{if } i = j \\ -\frac{1}{\sqrt{d_i d_j}} & \text{if } (ij) \in \mathcal{E}, i \neq j \\ 0 & \text{otherwise.} \end{cases}$$

  – The eigengap ($\underline{\lambda}_{\min}$: smallest non-zero eigenvalue)

$$\xi(\mathcal{L}) = \underline{\lambda}_{\min}(\mathcal{L}) / \lambda_{\max}(\mathcal{L}) \leq 1.$$

Network Class

- The eigengap of NL for a number of special graphs

  1) **Complete Graph:** $\xi(\mathcal{L}) = 1$;

  2) **Star Graph:** $\xi(\mathcal{L}) = 1/2$;

  3) **Path Graph:** $\xi(\mathcal{L}) = \mathcal{O}(1/M^2)$.

  4) **Grid Graph:** $\xi(\mathcal{L}) = \mathcal{O}(1/M)$.

  5) **Random Geometric Graph:** Place the nodes uniformly in $[0,1]^2$ and connect any two nodes separated by a distance less than a radius $R \in (0,1)$. With high probability $\xi(\mathcal{L}) = \mathcal{O}\left(\frac{\log(M)}{M}\right)$.

Algorithm Class

Algorithm Class

Algorithm Class $\mathcal{A}$: distributed, first-order algorithm

Algorithm Class

Algorithm Class $\mathcal{A}$: distributed, first-order algorithm

- One round of message exchanges among all nodes, and one round of local updates per iteration

Algorithm Class

Algorithm Class $\mathcal{A}$: distributed, first-order algorithm

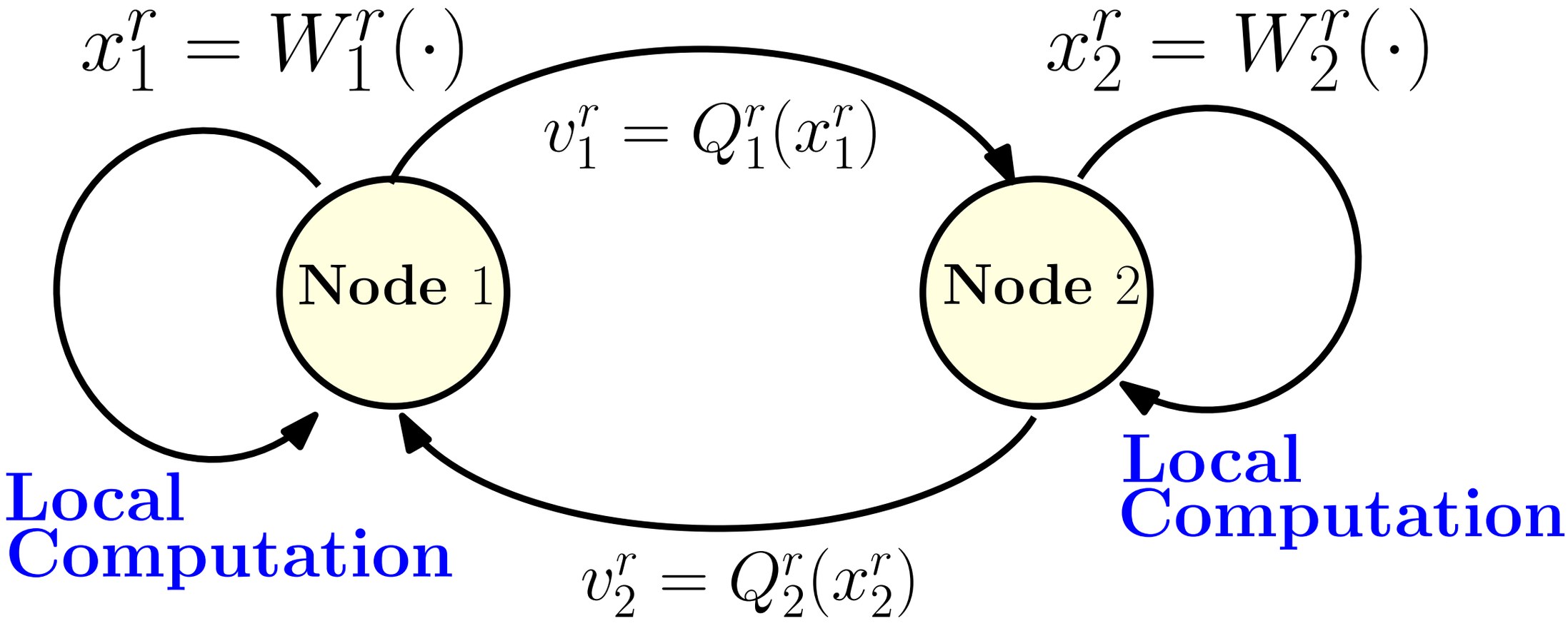
- One round of message exchanges among all nodes, and one round of local updates per iteration

- The output of node i is a linear combination of the historical output and gradients of its neighboring set

$$\mathbf{v}_i^r = \underbrace{Q_i^r(\mathbf{x}_i^r)}_{\text{communication step}}, \ \mathbf{x}_i^{r+1} \in \underbrace{W_i^r\left(\{\{\mathbf{v}_j^r\}_{j\in\mathcal{N}_i}, \nabla f_i(\mathbf{x}_i^t), \mathbf{x}_i^t\}_{t=1}^r\right)}_{\text{computation step}}. \quad (1)$$

$\mathcal{N}_i := i \cup \{j\}_{j\sim i}$ denotes the neighbor set for node $i \in \mathcal{E}$; Q_i^r, W_i^r are some linear operators

Algorithm Class



Solution Quality

- We say that $\{\mathbf{x}_i\}$ is an ϵ-solution if the following holds

$$h_T^* := \min_{r\in[T]} \underbrace{\left\| \frac{1}{M}\sum_{i=1}^{M} \nabla f_i(\mathbf{x}_i) \right\|^2}_{\text{gradient size}} + \underbrace{\frac{1}{M^2}\sum_{i,j} \sqrt{L_i L_j}\|\mathbf{x}_i - \mathbf{x}_j\|^2}_{\text{weighted average consensus error}} \leq \epsilon.$$

- Interpret T as the first time that the "optimality gap" reaches below ϵ

Lower Complexity Bound

Claim 1. [Sun-H. 18] Let $L_i = U, \forall\ I$. There exists a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ in $\in \mathcal{N}_M$, and a problem in class $\mathcal{P}_L^M$, such that for any algorithm in $\mathcal{A}$, the total number of iterations required to reach a precision $h_T^* \leq \epsilon$ is lower bounded by

$$T \geq \Omega\left(\frac{1}{\sqrt{\xi(\mathcal{L})}}\frac{1}{\epsilon} \times U \times (f(0) - \underline{f})\right)$$

Lower Complexity Bound

Claim 1. [Sun-H. 18] Let $L_i = U, \forall\ I$. There exists a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ in $\in \mathcal{N}_M$, and a problem in class $\mathcal{P}_L^M$, such that for any algorithm in $\mathcal{A}$, the total number of iterations required to reach a precision $h_T^* \leq \epsilon$ is lower bounded by

$$T \geq \Omega\left(\frac{1}{\sqrt{\xi(\mathcal{L})}}\frac{1}{\epsilon} \times U \times (f(0) - \underline{f})\right)$$

Remark. This bound is $1/\sqrt{\xi(\mathcal{L})}$ times worse than the centralized GD (lower bound proving in [Carmon et al 17])

Lower Complexity Bound

Claim 1. [Sun-H. 18] Let $L_i = U, \forall\ I$. There exists a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ in $\in \mathcal{N}_M$, and a problem in class $\mathcal{P}_L^M$, such that for any algorithm in $\mathcal{A}$, the total number of iterations required to reach a precision $h_T^* \leq \epsilon$ is lower bounded by

$$T \geq \Omega\left(\frac{1}{\sqrt{\xi(\mathcal{L})}}\frac{1}{\epsilon} \times U \times (f(0) - \underline{f})\right)$$

Remark. This bound is $1/\sqrt{\xi(\mathcal{L})}$ times worse than the centralized GD (lower bound proving in [Carmon et al 17])

Remark. For path graph, the lower bound is in the order of

$$\Omega\left(M \times \frac{1}{\epsilon}\right)$$

Lower Complexity Bound

Claim 1. [Sun-H. 18] Let $L_i = U, \forall\ I$. There exists a graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ in $\in \mathcal{N}_M$, and a problem in class $\mathcal{P}_L^M$, such that for any algorithm in $\mathcal{A}$, the total number of iterations required to reach a precision $h_T^* \leq \epsilon$ is lower bounded by

$$T \geq \Omega\left(\frac{1}{\sqrt{\xi(\mathcal{L})}}\frac{1}{\epsilon} \times U \times (f(0) - \underline{f})\right)$$

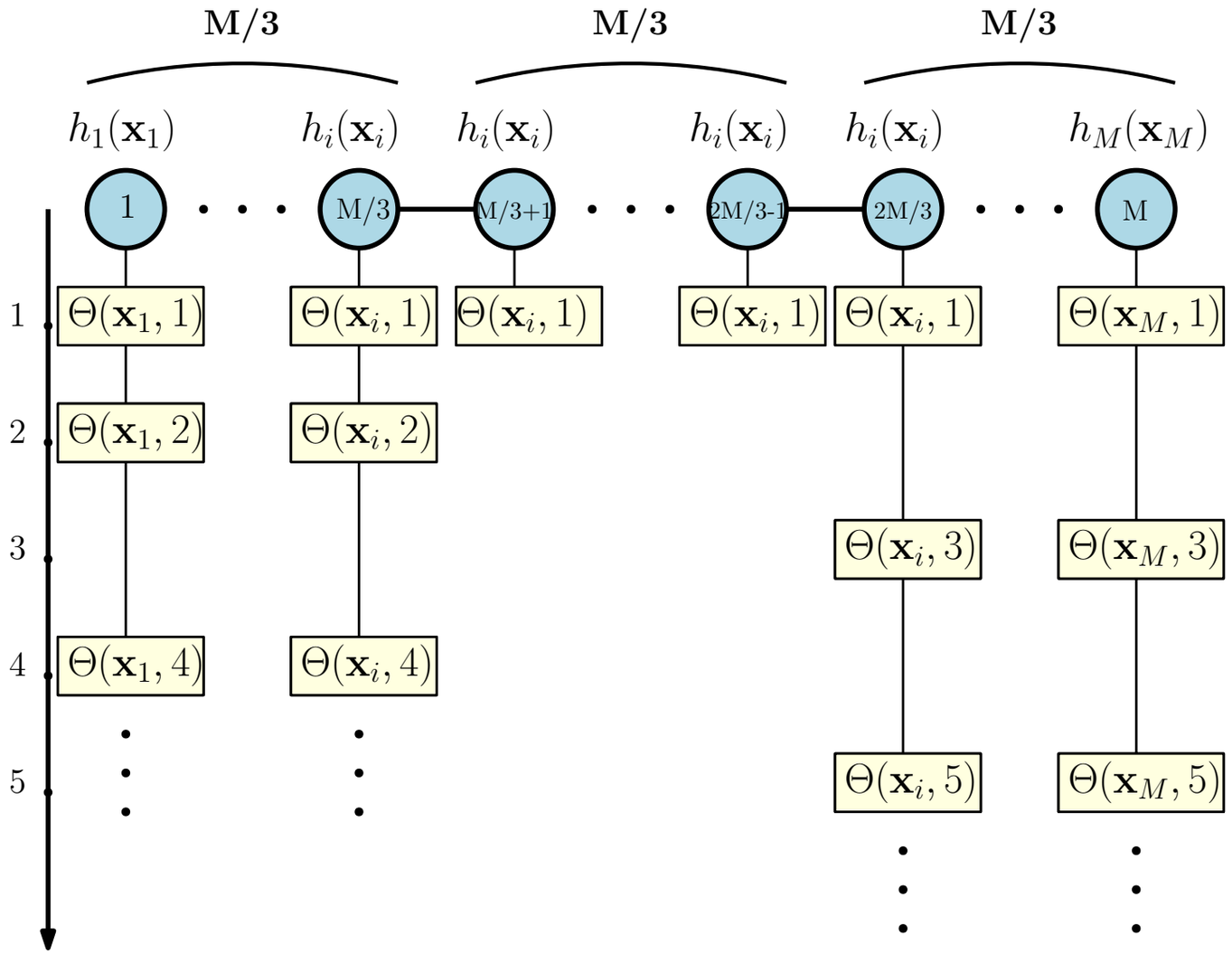
Remark. This bound is $1/\sqrt{\xi(\mathcal{L})}$ times worse than the centralized GD (lower bound proving in [Carmon et al 17])

Remark. For path graph, the lower bound is in the order of

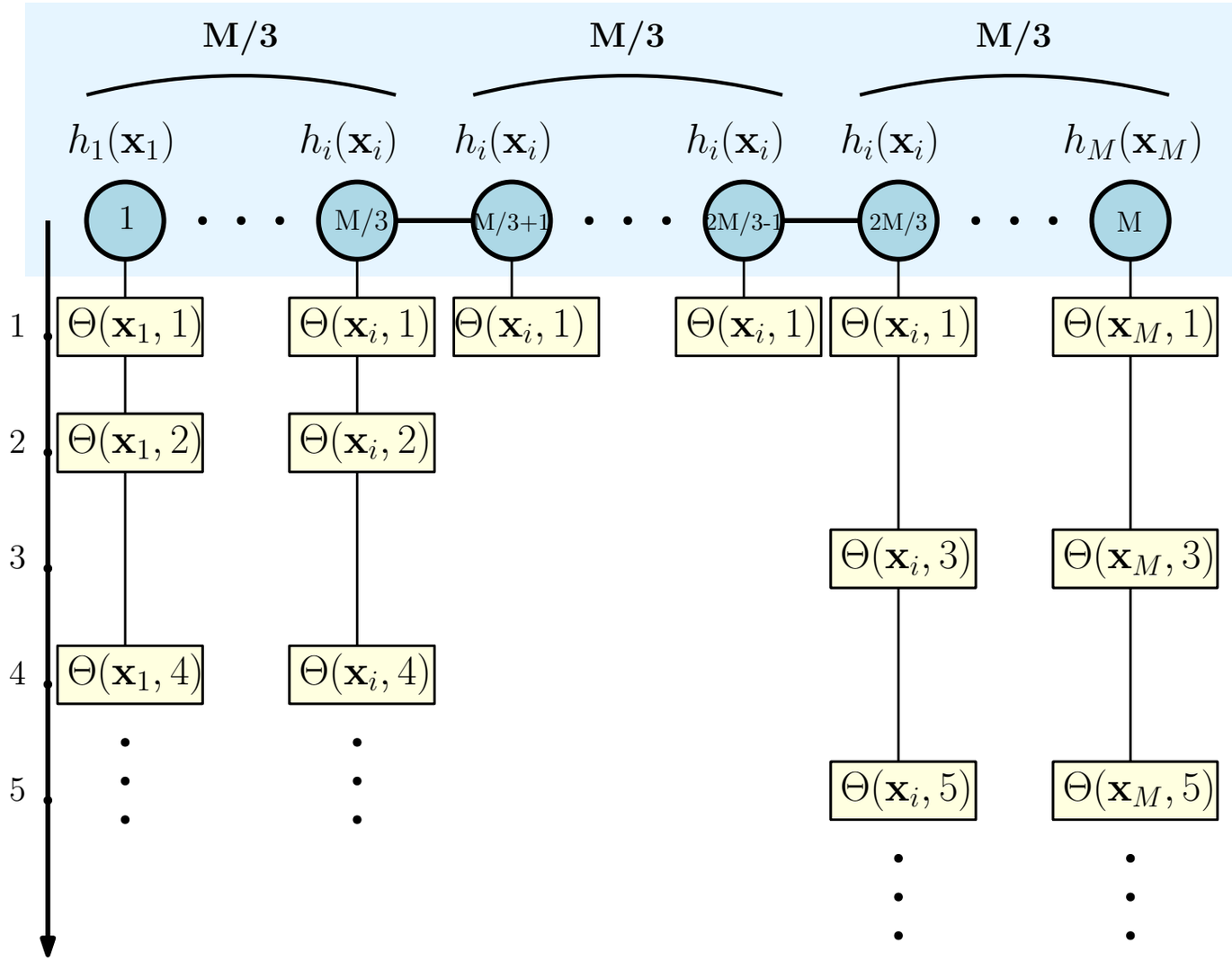
$$\Omega\left(M \times \frac{1}{\epsilon}\right)$$

Remark. Computation complexity lower bound is $\Omega\left(\frac{1}{\epsilon}\right)$, same as the centralized lower bound proving in [Carmon et al 17]

Proof sketch (1): Path Graph

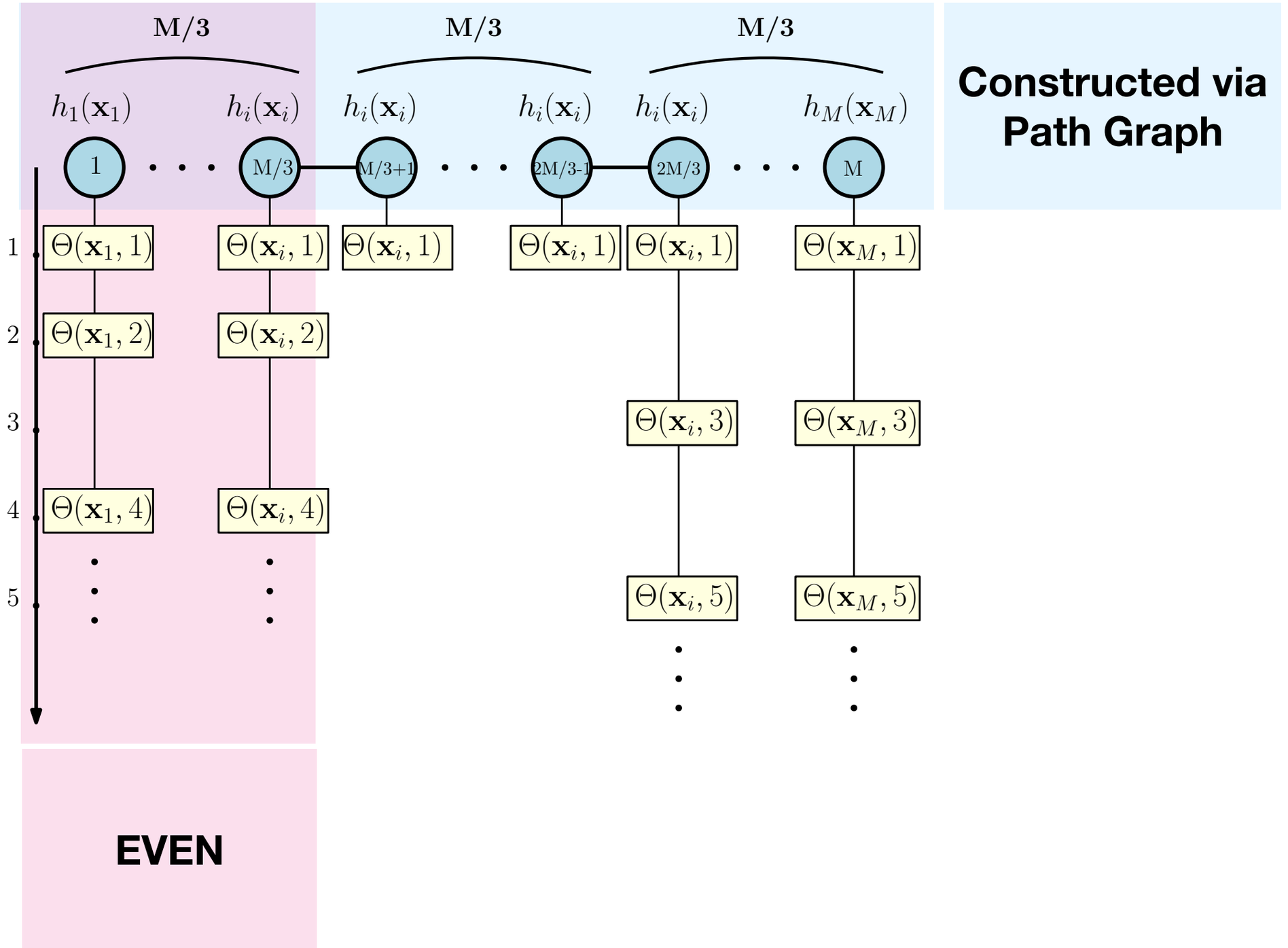


Proof sketch (1): Path Graph

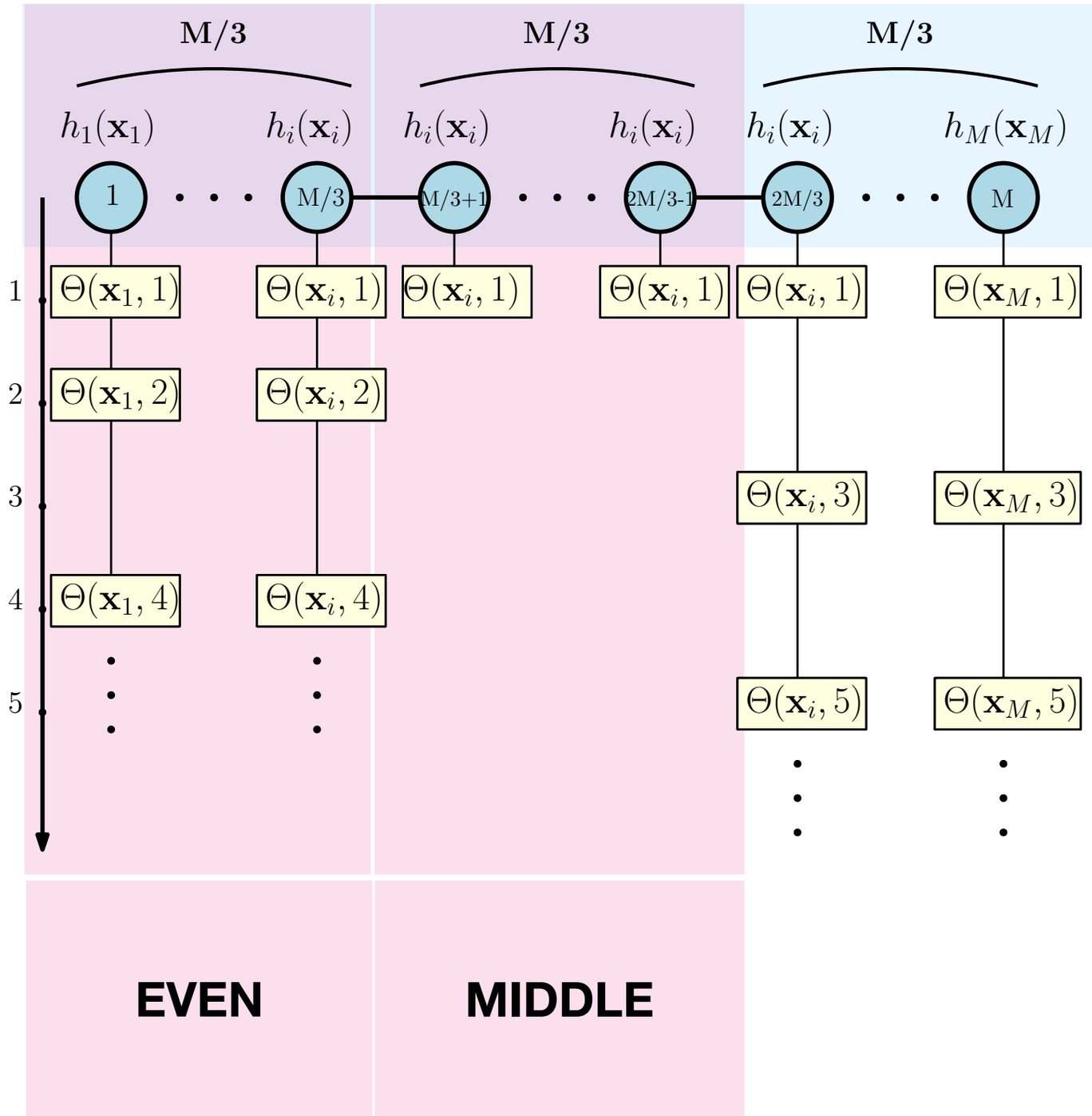


Constructed via Path Graph

Proof sketch (1): Path Graph

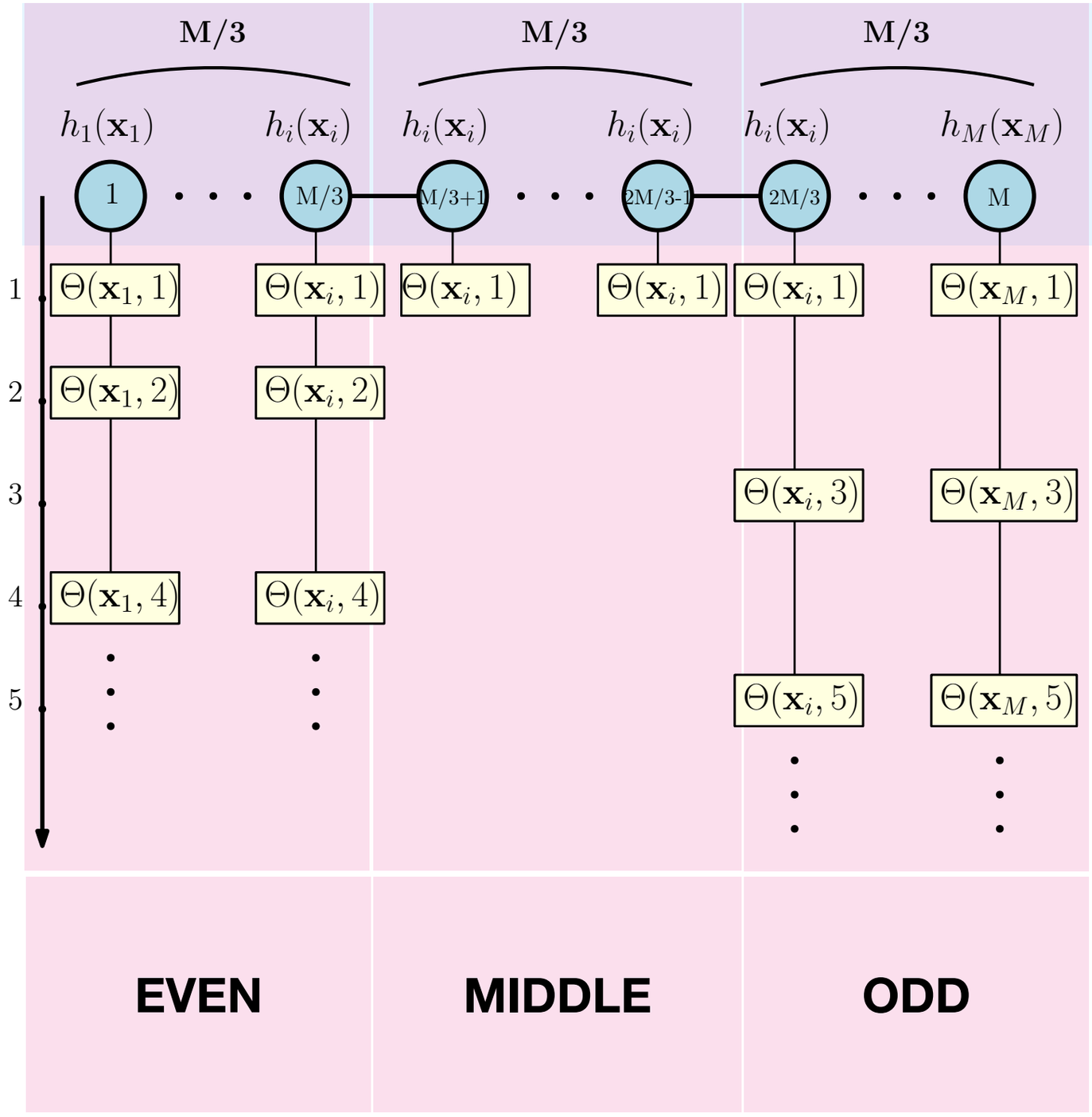


Proof sketch (1): Path Graph



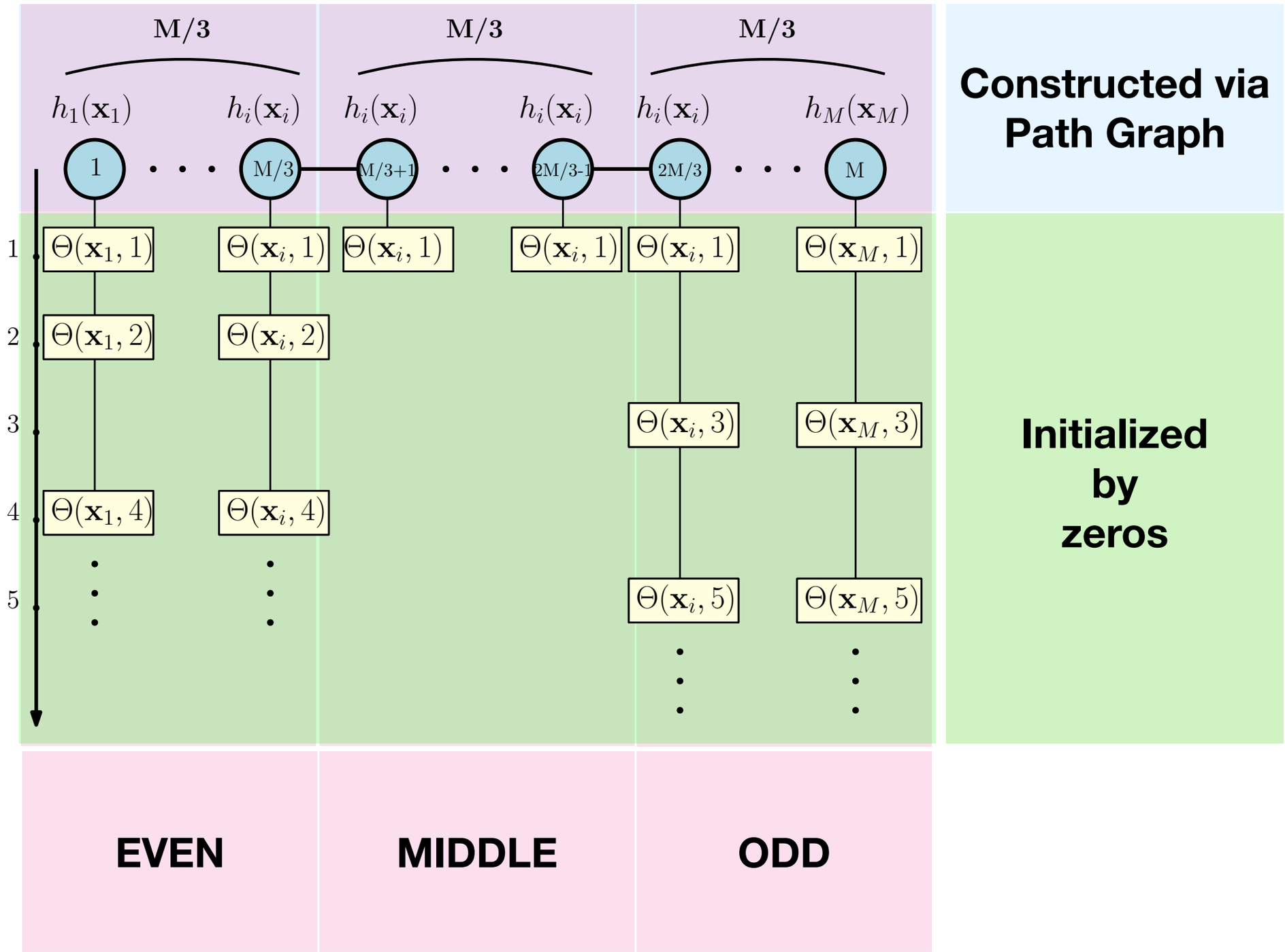
Constructed via Path Graph

Proof sketch (1): Path Graph

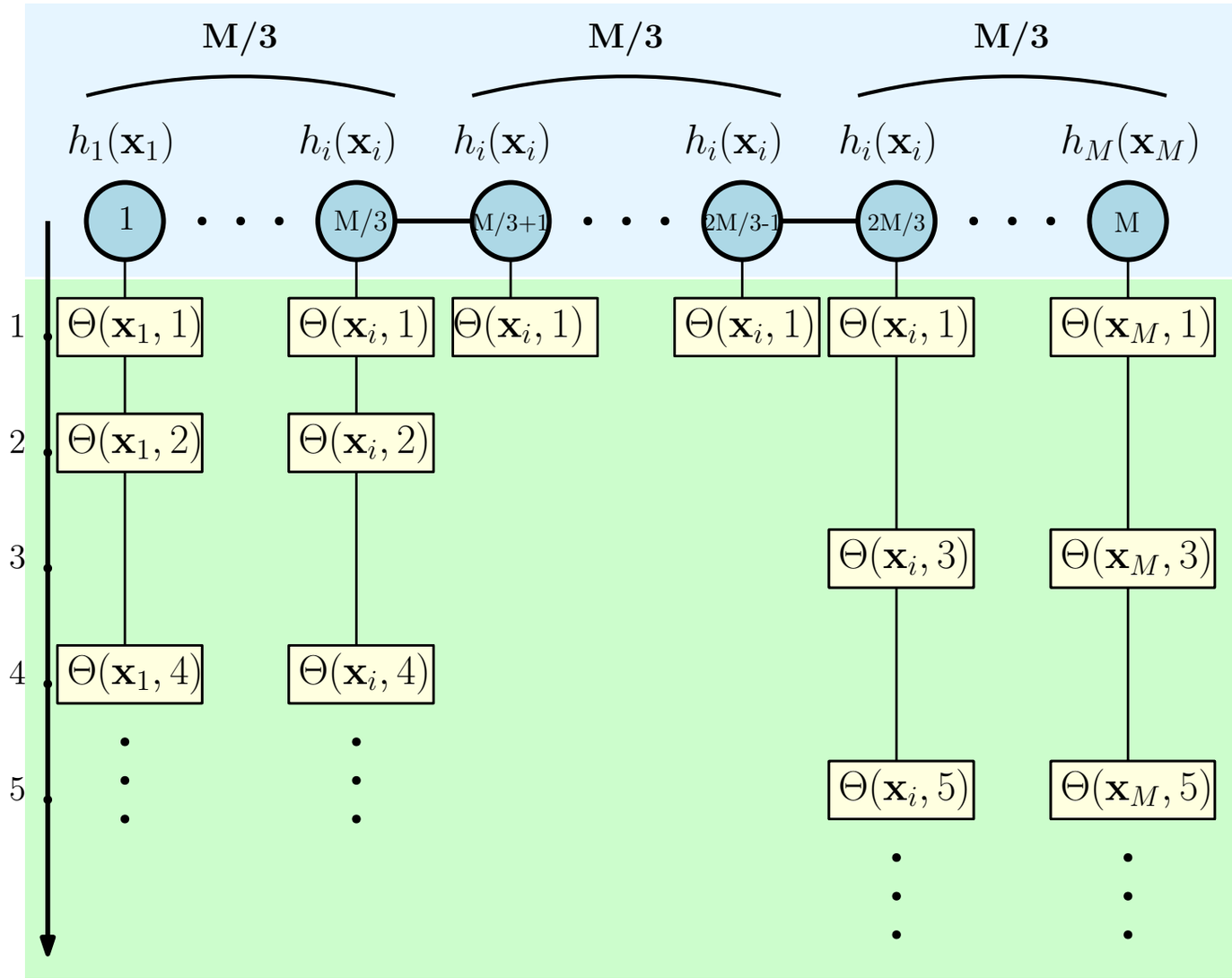


Constructed via Path Graph

Proof sketch (1): Path Graph



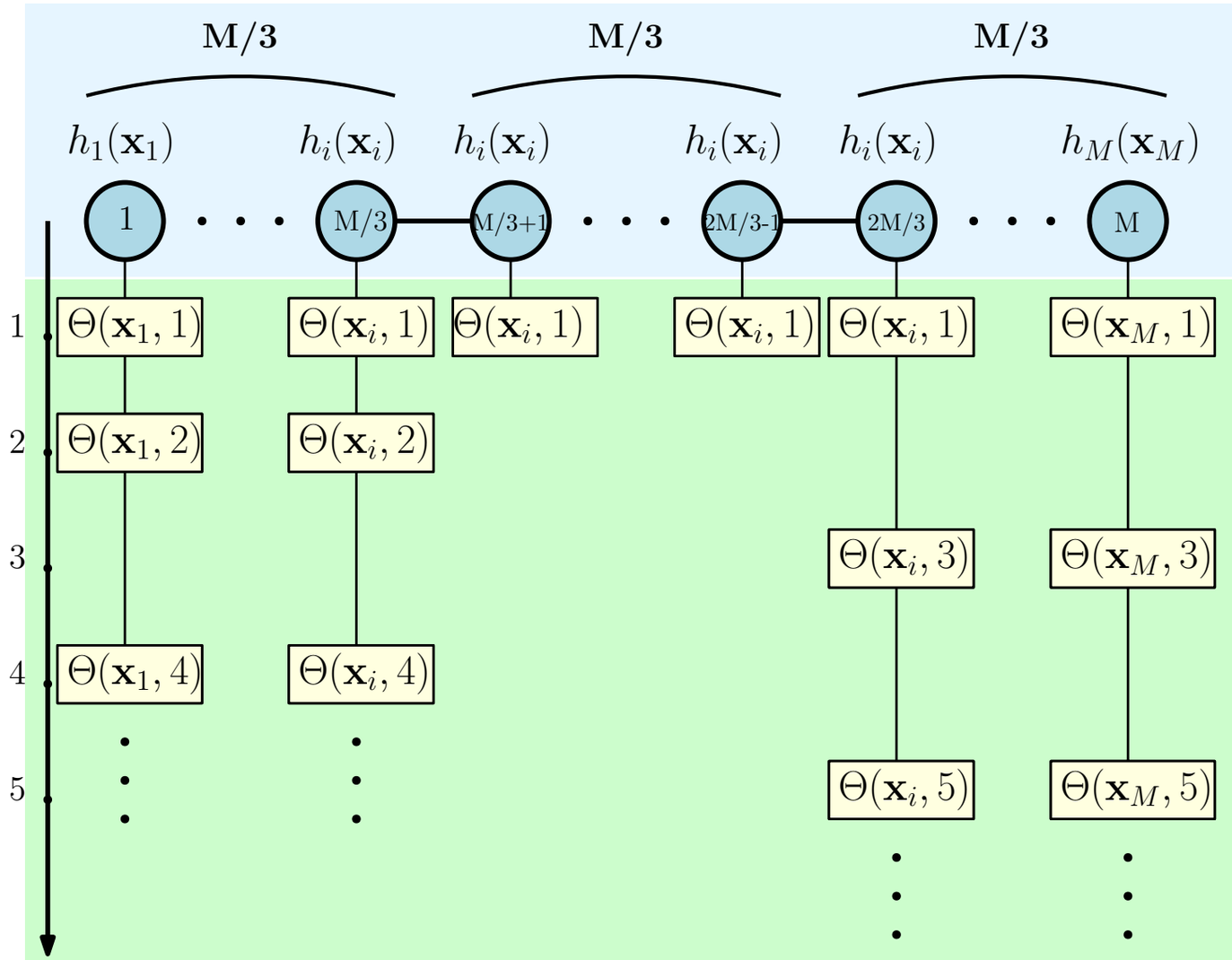
Proof sketch (1): Path Graph



Constructed via Path Graph

Initialized by zeros

Proof sketch (1): Path Graph



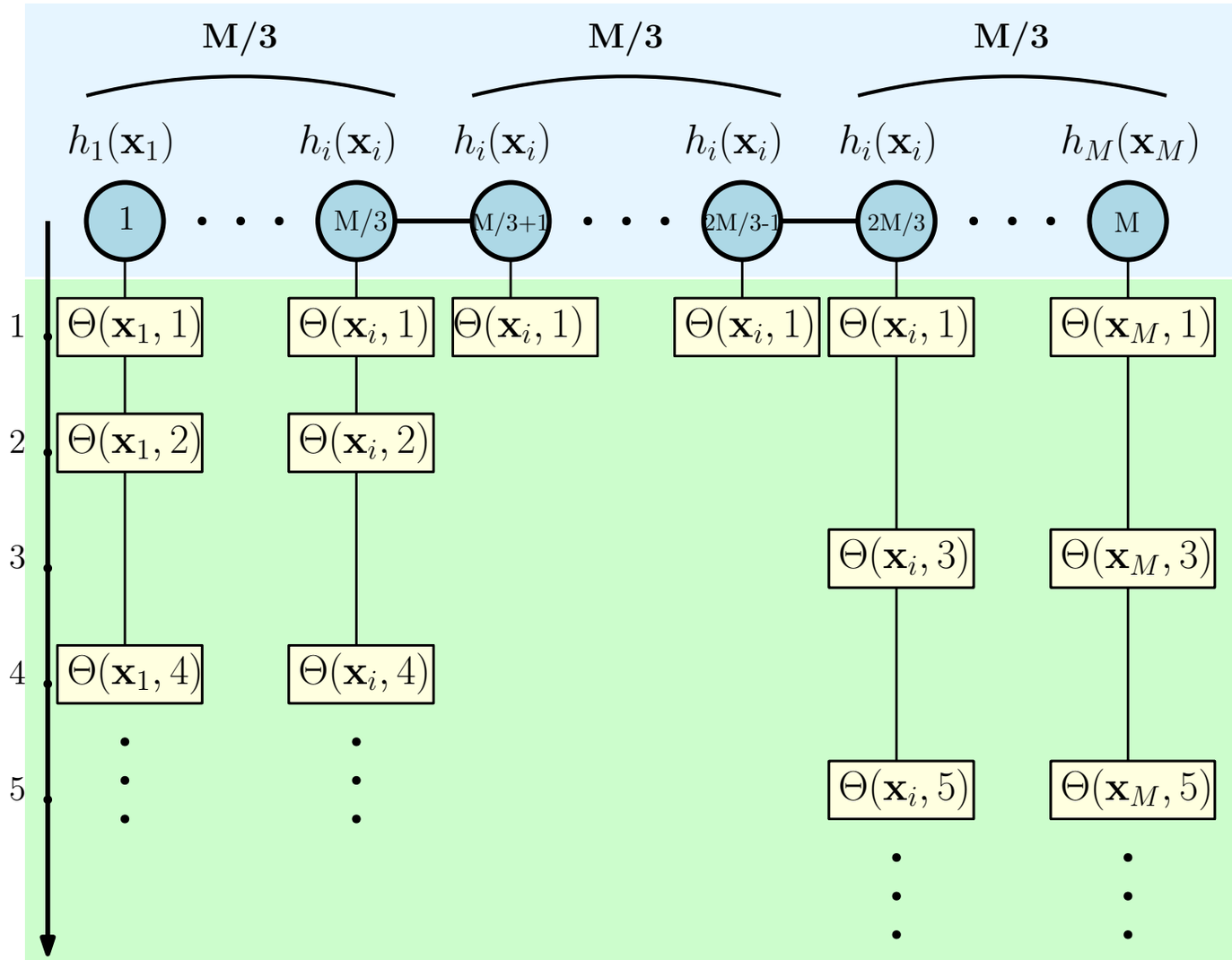
Constructed via Path Graph

Initialized by zeros

If there exists $k \in [T]$ such that $|\mathbf{x}_i[k]| < 1$, then

$$\|\frac{1}{M}\sum_{i=1}^{M} \nabla h_i(\mathbf{x}_i)\| > 1.$$

Proof sketch (1): Path Graph



Constructed via Path Graph

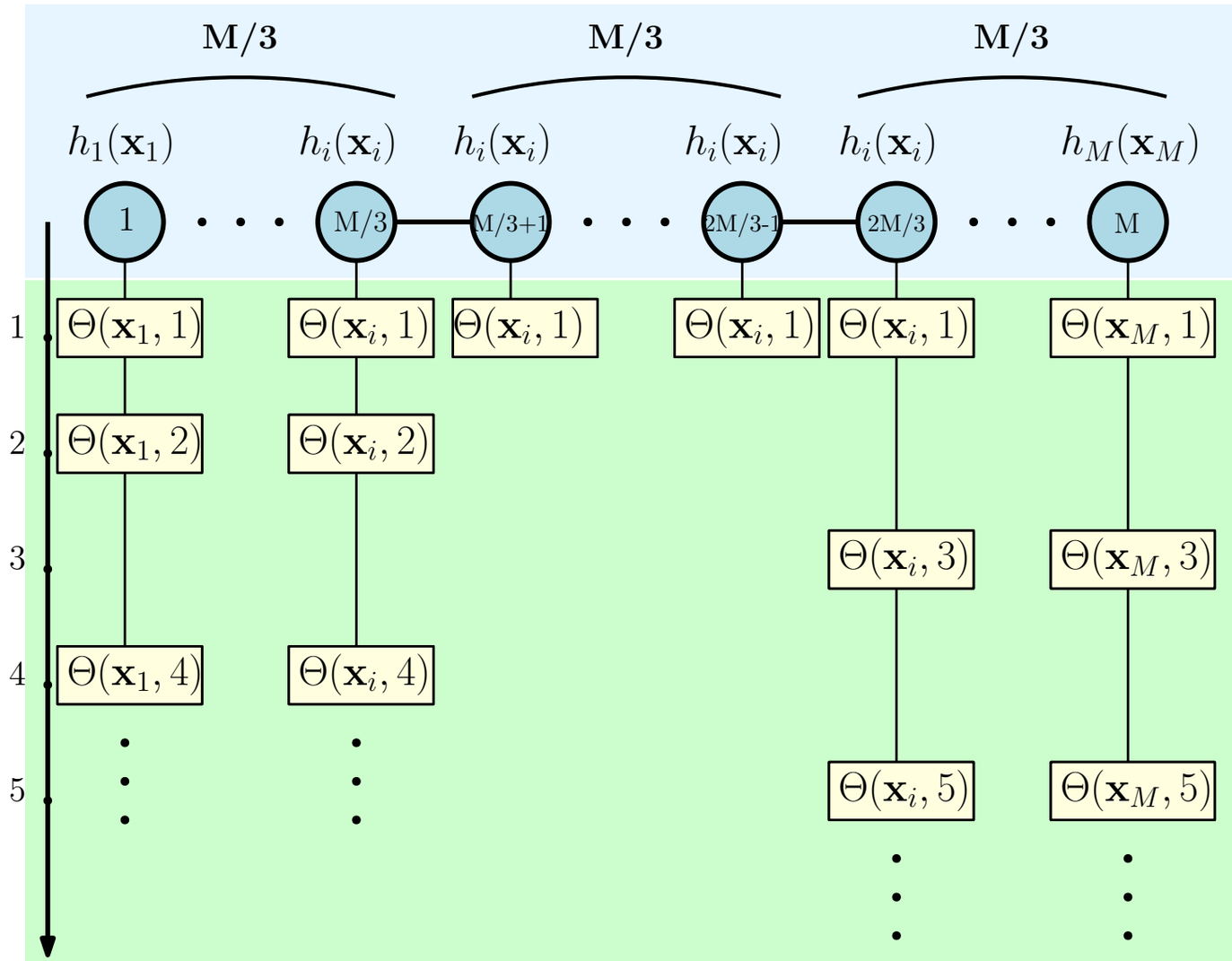
Initialized by zeros

If there exists $k \in [T]$ such that $|\mathbf{x}_i[k]| < 1$, then

$$\|\frac{1}{M}\sum_{i=1}^{M} \nabla h_i(\mathbf{x}_i)\| > 1.$$

All elements need to be non-zero

Proof sketch (1): Path Graph



Constructed via Path Graph

Initialized by zeros

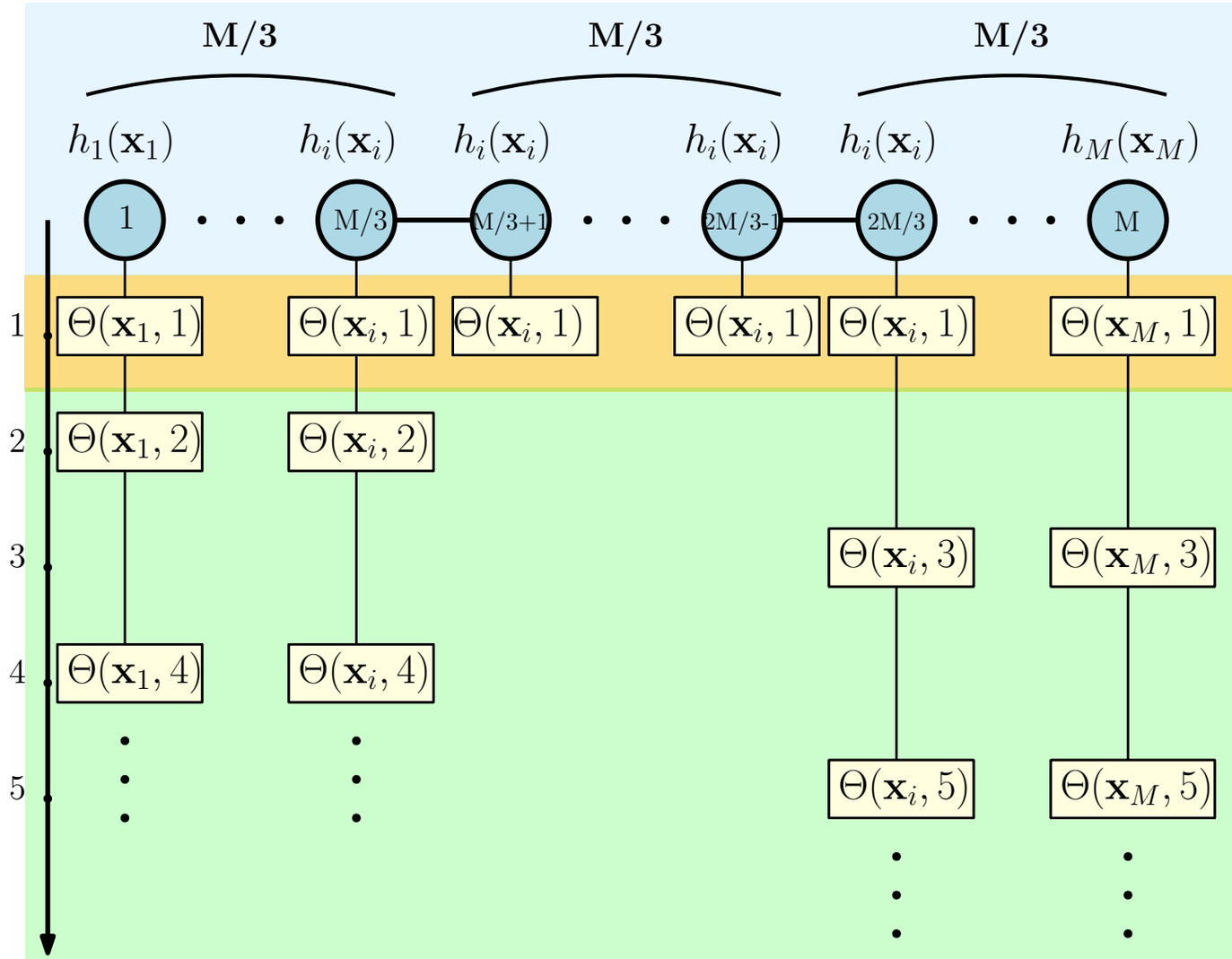
If there exists $k \in [T]$ such that $|\mathbf{x}_i[k]| < 1$, then

$$\|\frac{1}{M}\sum_{i=1}^{M} \nabla h_i(\mathbf{x}_i)\| > 1.$$

All elements need to be non-zero

If $\mathbf{x}_i[j-1] = \mathbf{x}_i[j] = 0$, then $\frac{\partial h_i(\mathbf{x}_i)}{\partial \mathbf{x}_i[j]} = 0, \forall j \geq 2.$

Proof sketch (1): Path Graph



Constructed via Path Graph

non-zero

Initialized by zeros

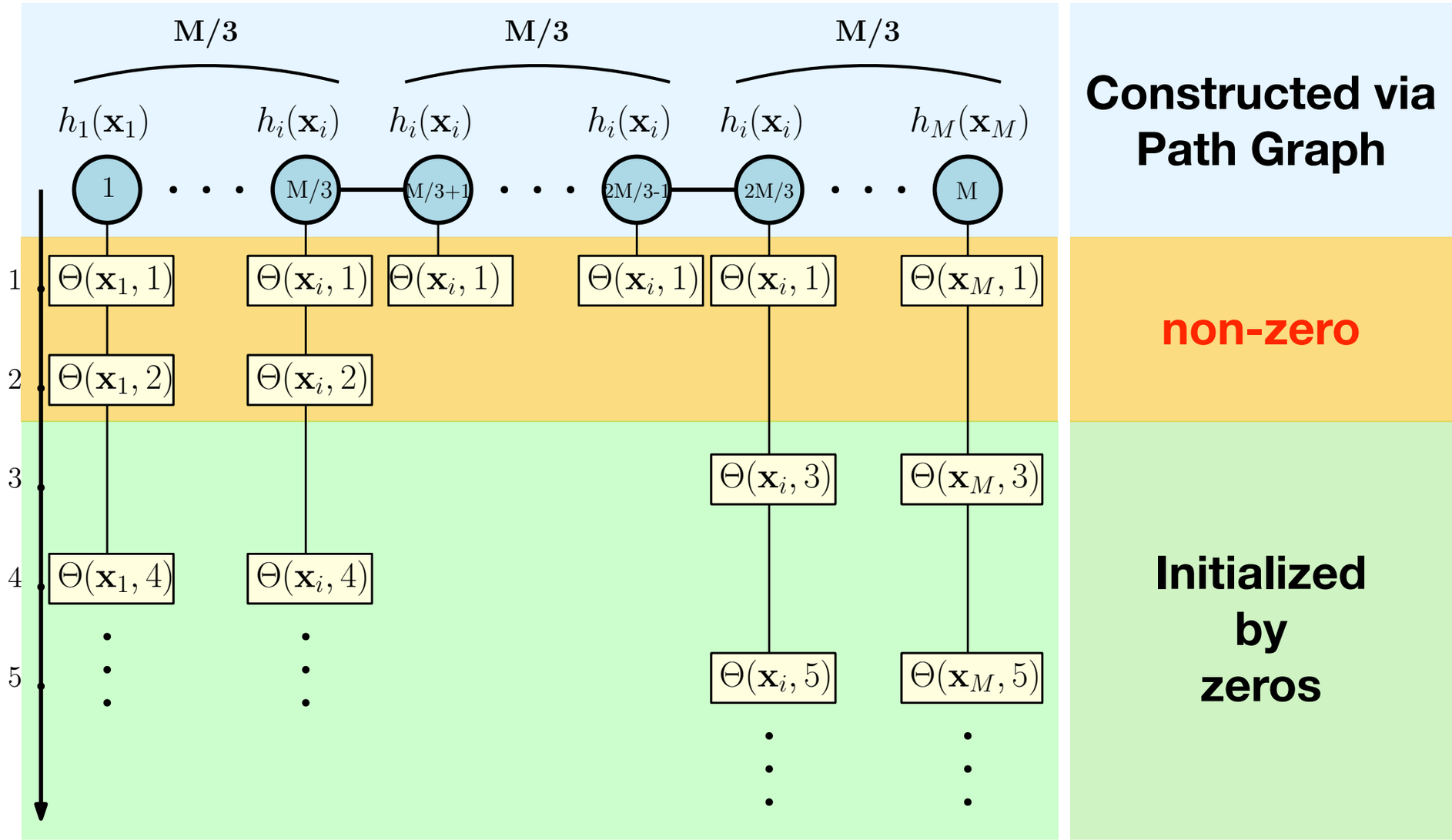
If there exists $k \in [T]$ such that $|\mathbf{x}_i[k]| < 1$, then

$$\|\frac{1}{M}\sum_{i=1}^{M} \nabla h_i(\mathbf{x}_i)\| > 1.$$

All elements need to be non-zero

If $\mathbf{x}_i[j-1] = \mathbf{x}_i[j] = 0$, then $\frac{\partial h_i(\mathbf{x}_i)}{\partial \mathbf{x}_i[j]} = 0, \forall j \geq 2.$

Proof sketch (1): Path Graph



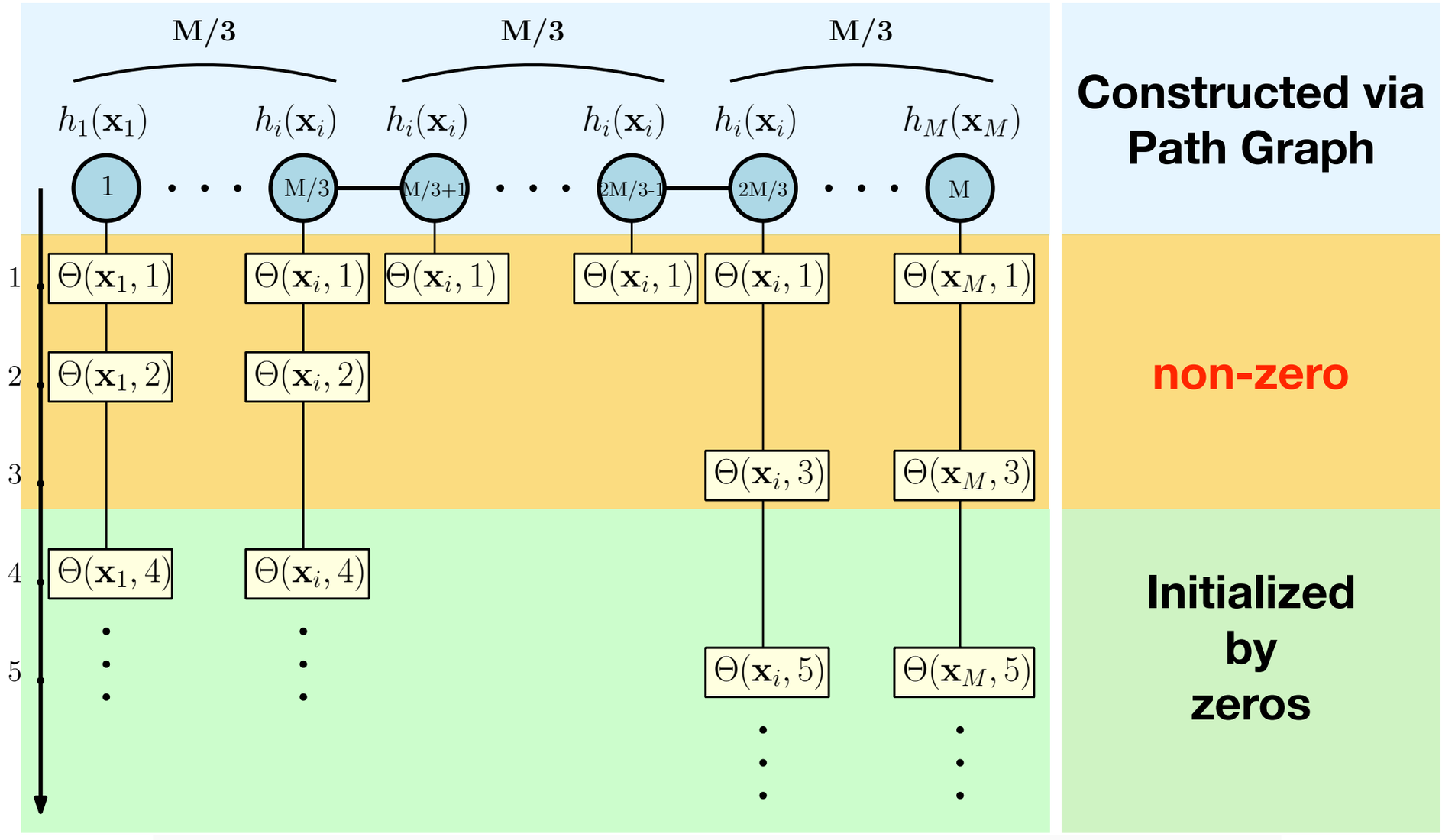
If there exists $k \in [T]$ such that $|\mathbf{x}_i[k]| < 1$, then

$$\|\frac{1}{M}\sum_{i=1}^{M} \nabla h_i(\mathbf{x}_i)\| > 1.$$

All elements need to be non-zero

If $\mathbf{x}_i[j-1] = \mathbf{x}_i[j] = 0$, then $\frac{\partial h_i(\mathbf{x}_i)}{\partial \mathbf{x}_i[j]} = 0, \forall j \geq 2.$

Proof sketch (1): Path Graph



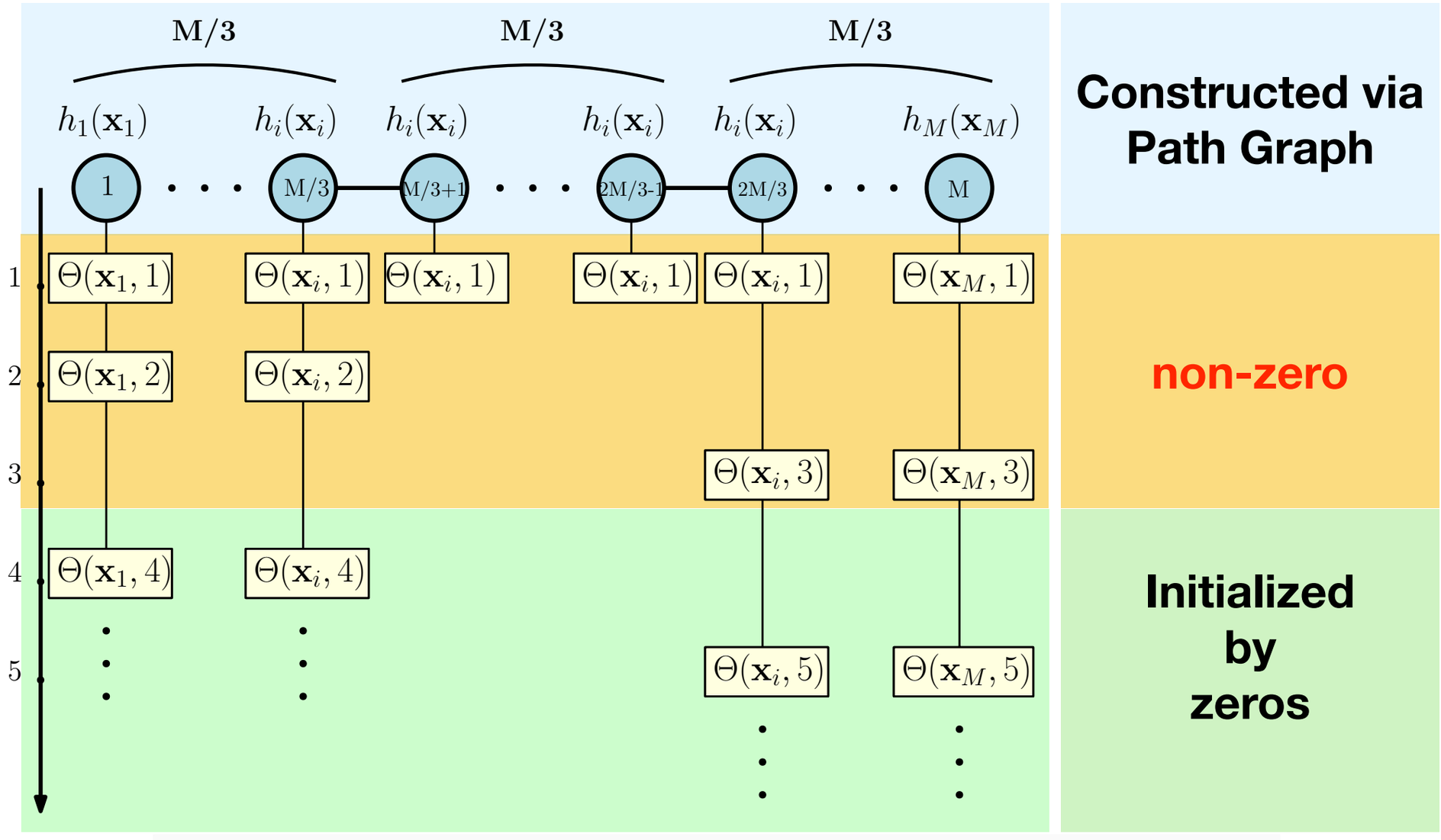
If there exists $k \in [T]$ such that $|\mathbf{x}_i[k]| < 1$, then

$$\|\frac{1}{M}\sum_{i=1}^{M} \nabla h_i(\mathbf{x}_i)\| > 1.$$

All elements need to be non-zero

$$\text{If } \mathbf{x}_i[j-1] = \mathbf{x}_i[j] = 0, \text{ then } \frac{\partial h_i(\mathbf{x}_i)}{\partial \mathbf{x}_i[j]} = 0, \forall j \geq 2.$$

Proof sketch (1): Path Graph



If there exists $k \in [T]$ such that $|\mathbf{x}_i[k]| < 1$, then

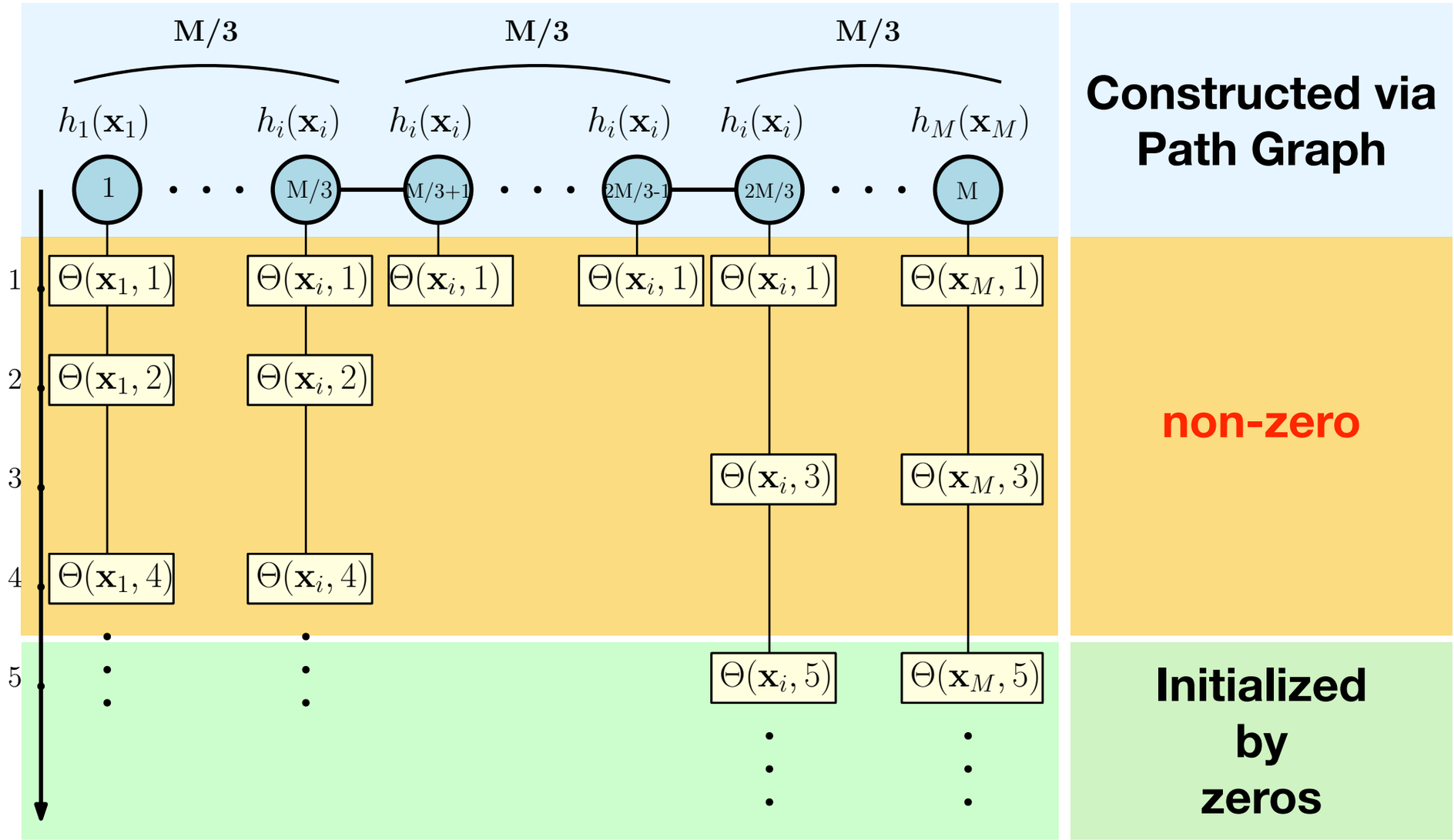
$$\|\frac{1}{M}\sum_{i=1}^{M} \nabla h_i(\mathbf{x}_i)\| > 1.$$

All elements need to be non-zero

If $\mathbf{x}_i[j-1] = \mathbf{x}_i[j] = 0$, then $\frac{\partial h_i(\mathbf{x}_i)}{\partial \mathbf{x}_i[j]} = 0, \forall i \geq 2$

Each iteration can only increase the non-zero dimensions by one

Proof sketch (1): Path Graph



- **All elements need to be non-zero**
- **Each iteration can only increase the non-zero dimensions by one**
- **Alternating update even and odd dimensions**
- **At least O(M) communication per iteration**

Rate Optimal Algorithms

Key Observation 1

Recall our consensus problem (for simplicity assume scalar variables)

$$\min_{\mathbf{x}\in\mathbb{R}^M} \; f(\mathbf{x}) := \frac{1}{M}\sum_{i=1}^{M} f_i(\mathbf{x}_i), \quad \text{s.t.} \quad \mathbf{A}\mathbf{x} = 0.$$

- **Observation 1**: Problem exhibits a **separation structure**
- The non-convexity only arises in the objective function
- The network structure only appears in the null space constraint

Key Observation 2

- **Observation 2**: Our lower bound is given by

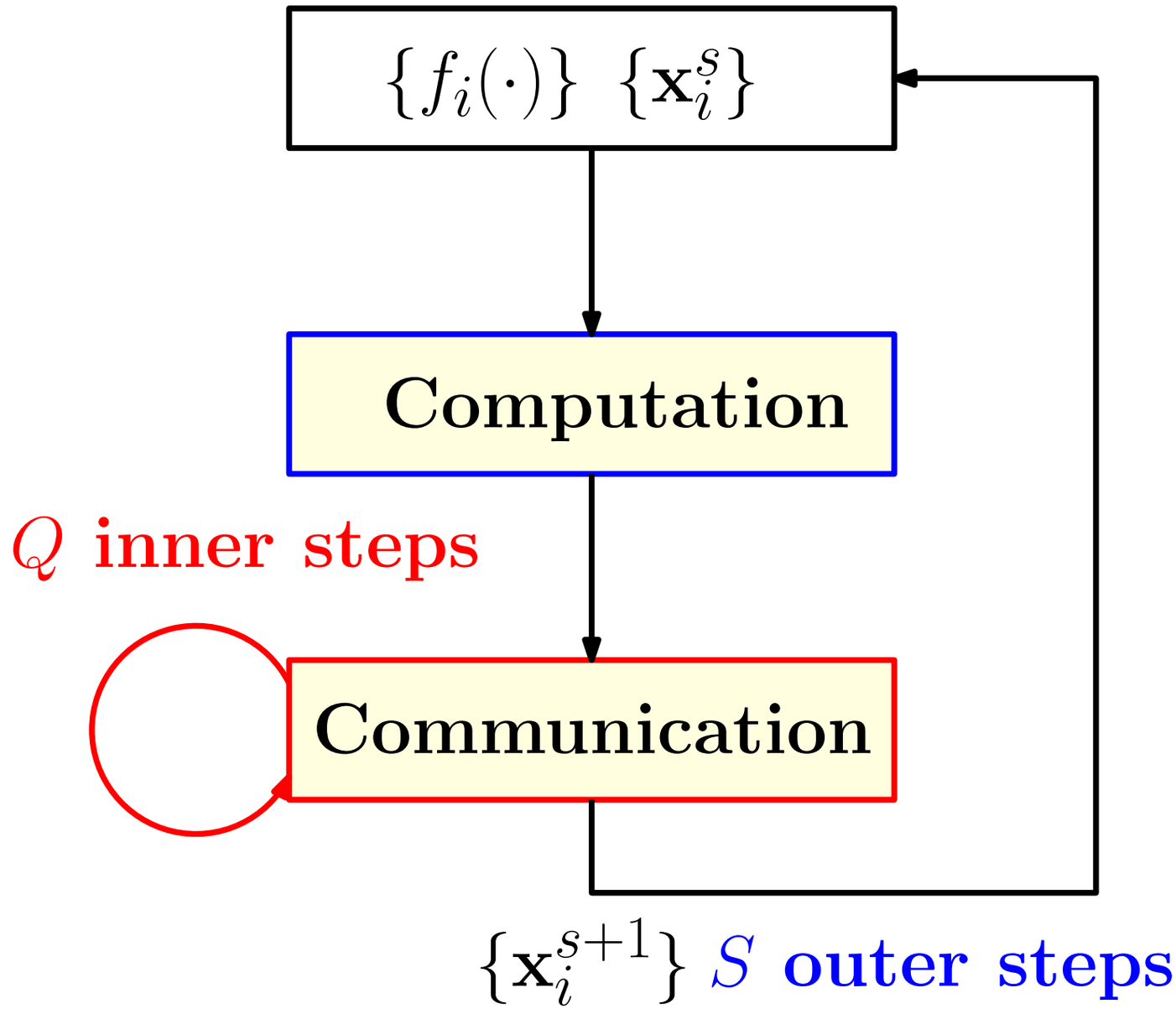
$$T \geq \Omega\left(\frac{1}{\sqrt{\xi(\mathcal{L})}}\frac{1}{\epsilon} \times U \times (f(0) - \underline{f})\right).$$

  Factored into the product of $\frac{1}{\sqrt{\xi(\mathcal{L})}}$ and $\frac{1}{\epsilon}$ (ignoring other constants)

- $1/\sqrt{\xi(\mathcal{L})}$: communication efficiency over the network

- $1/\epsilon$: computational complexity in minimizing the objective (lower bound for centralized GD)

Hierarchical Decomposition Strategy



Decompose communication and computation subtasks to form some nested loops

Primal Dual Based Methods

- Primal update

$$\mathbf{x}^{r+1} = \arg\min_{\mathbf{x}} \underbrace{\langle \nabla f(\mathbf{x}^r) + \mathbf{A}^T\lambda^r + \rho\mathbf{A}^T\mathbf{A}\mathbf{x}^r, \mathbf{x} - \mathbf{x}^r\rangle}_{\text{linearizes the entire AL function}}$$

$$+ \underbrace{\frac{\beta}{2}\|\mathbf{x} - \mathbf{x}^r\|^2}_{\text{prox for gradient}} + \underbrace{\frac{\rho\lambda_{\max}(\mathbf{A}^T\mathbf{A})}{2}\|\mathbf{x} - \mathbf{x}^r\|^2}_{\text{prox for network}}$$

[1] Hong, Mingyi, Davood Hajinezhad, and Ming-Min Zhao. "Prox-PDA: The proximal primal-dual algorithm for fast distributed nonconvex optimization and learning over networks." In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1529-1538. JMLR. org, 2017.

Primal Dual Based Methods

- Primal update

$$\mathbf{x}^{r+1} = \arg\min_{\mathbf{x}} \underbrace{\langle \nabla f(\mathbf{x}^r) + \mathbf{A}^T\lambda^r + \rho\mathbf{A}^T\mathbf{A}\mathbf{x}^r, \mathbf{x} - \mathbf{x}^r\rangle}_{\text{linearizes the entire AL function}}$$
$$+ \underbrace{\frac{\beta}{2}\|\mathbf{x} - \mathbf{x}^r\|^2}_{\text{prox for gradient}} + \underbrace{\frac{\rho\lambda_{\max}(\mathbf{A}^T\mathbf{A})}{2}\|\mathbf{x} - \mathbf{x}^r\|^2}_{\text{prox for network}}$$

  - Communication and computation are treated "equally"
  - Prox-PDA achieves "suboptimal" rate: $\mathcal{O}\left(\frac{1}{\xi(\mathcal{L})}\frac{1}{\epsilon}\right)$

[1] Hong, Mingyi, Davood Hajinezhad, and Ming-Min Zhao. "Prox-PDA: The proximal primal-dual algorithm for fast distributed nonconvex optimization and learning over networks." In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1529-1538. JMLR. org, 2017.

Primal Dual Based Methods

- Primal update

$$\mathbf{x}^{r+1} = \arg\min_{\mathbf{x}} \underbrace{\langle \nabla f(\mathbf{x}^r) + \mathbf{A}^T\lambda^r + \rho\mathbf{A}^T\mathbf{A}\mathbf{x}^r, \mathbf{x} - \mathbf{x}^r\rangle}_{\text{linearizes the entire AL function}}$$
$$+ \underbrace{\frac{\beta}{2}\|\mathbf{x} - \mathbf{x}^r\|^2}_{\text{prox for gradient}} + \underbrace{\frac{\rho\lambda_{\max}(\mathbf{A}^T\mathbf{A})}{2}\|\mathbf{x} - \mathbf{x}^r\|^2}_{\text{prox for network}}$$

- Communication and computation are treated "equally"
- Prox-PDA achieves "suboptimal" rate: $\mathcal{O}\left(\frac{1}{\xi(\mathcal{L})}\frac{1}{\epsilon}\right)$
- This is done by linearize the AL and add two regularization terms
- Intuitively, adding the network prox imposes too much regularization

$$\rho\lambda_{\max}(\mathbf{A}^T\mathbf{A})\|\mathbf{x} - \mathbf{x}^r\|^2$$

[1] Hong, Mingyi, Davood Hajinezhad, and Ming-Min Zhao. "Prox-PDA: The proximal primal-dual algorithm for fast distributed nonconvex optimization and learning over networks." In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1529-1538. JMLR. org, 2017.

Algorithm Design

- Do not add the "network prox"? The problem becomes

$$\mathbf{x}^{r+1} = \arg\min_{\mathbf{x}} \underbrace{\langle \nabla f(\mathbf{x}^r) + \mathbf{A}^T \lambda^r, \mathbf{x} - \mathbf{x}^r \rangle}_{\text{linear term}} + \underbrace{\frac{\rho}{2}\|\mathbf{A}\mathbf{x}\|^2}_{\text{non-separable}} + \underbrace{\frac{\beta}{2}\|\mathbf{x} - \mathbf{x}^r\|^2}_{\text{separable}}$$

not separable among the agent

Algorithm Design

- Do not add the "network prox"? The problem becomes

$$\mathbf{x}^{r+1} = \arg\min_{\mathbf{x}} \underbrace{\langle \nabla f(\mathbf{x}^r) + \mathbf{A}^T \lambda^r, \mathbf{x} - \mathbf{x}^r \rangle}_{\text{linear term}} + \underbrace{\frac{\rho}{2}\|\mathbf{A}\mathbf{x}\|^2}_{\text{non-separable}} + \underbrace{\frac{\beta}{2}\|\mathbf{x} - \mathbf{x}^r\|^2}_{\text{separable}}$$

not separable among the agent

- Optimality condition: solving the following linear systems

$$\underbrace{-\nabla f(\mathbf{x}^r) - \mathbf{A}^T \lambda^r + \beta \mathbf{x}^r}_{:=\mathbf{d}^r} = \underbrace{(\rho \mathbf{A}^T \mathbf{A} + \beta I)}_{:=\mathbf{R}} \mathbf{x}^{r+1}$$

Algorithm Design

- How to proceed?

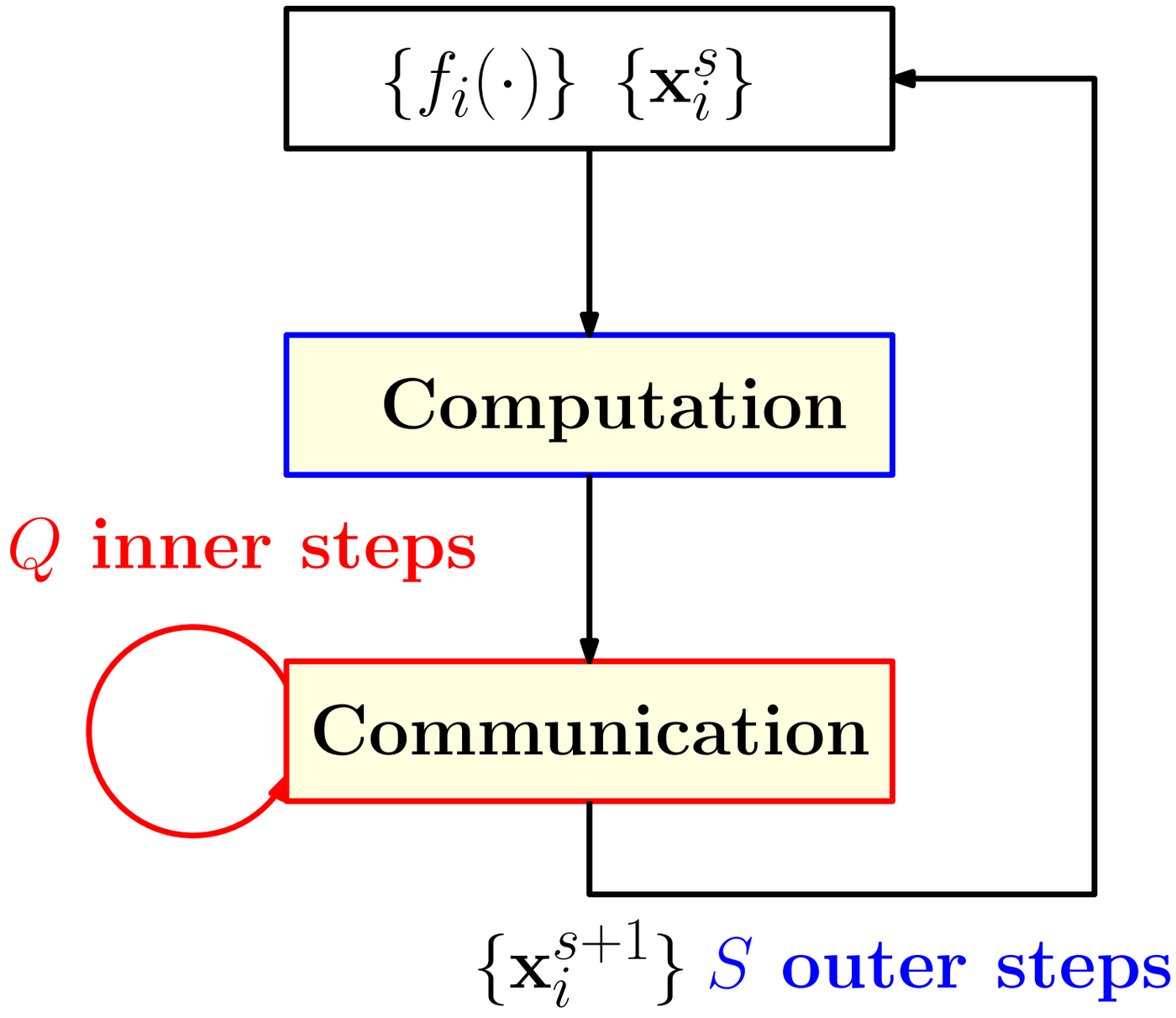
- Solving the system of equations

$$\underbrace{\left(\rho A^T A + \beta I\right)}_{:=R} x^* = d^r$$

  requires matrix inversion

- Destroys network structure, cannot be implemented distributedly
- **Idea**: Approximately solve it (by using an iterative method)

Hierarchical Decomposition Strategy



- computes local gradients
- approximately solves the linear system

Algorithm Design: xFILTER

Input: stepsize, initial iterates

Outer Iteration: Primal step

Inner Iteration: Approximate solve linear systems

Outer Iteration: Dual step + Communication

Algorithm Design: xFILTER

(S1). Assign parameters β and ρ.

(S2). Initialize $\mathbf{x}^{-1} = 0, \lambda^{-1} = 0$.

(S3) [Outer Iter.]. At iteration $s+1$, $s \geq -1$, define

$$\mathbf{R} := \rho \mathbf{A}^T \mathbf{A} + \beta \mathbf{I} \succ 0, \quad \mathbf{d} := -\nabla f(\mathbf{x}^s) - \mathbf{A}^T \lambda^s + \beta \mathbf{x}^s. \tag{1}$$

Outer Iteration: Primal step

(S4) [Inner Iter.]. Run Q **Chebyshev iterations** (with parameters $\{\alpha_t, \tau\}$)

$$\mathbf{u}_0 = \mathbf{x}^s, \mathbf{u}_1 = (\mathbf{I} - \tau \mathbf{R})\mathbf{u}_0 + \tau \mathbf{d}^s, \tag{2}$$

$$\mathbf{u}_t = \underbrace{\alpha_t(\mathbf{I} - \tau \mathbf{R})\mathbf{u}_{t-1} + (1 - \alpha_t)\mathbf{u}_{t-2} + \tau \alpha_t \mathbf{d}^s}_{\text{distributed implementable!}}, \ t = 2, \cdots, Q.$$

Inner Iteration: Approximate solve linear systems

(S5). [Outer Iter.] Set $\mathbf{x}^{s+1} = \mathbf{u}_Q$, update λ^{s+1} according to

$$\lambda^{s+1} = \lambda^s + \rho \mathbf{A} \mathbf{x}^{s+1} \tag{3}$$

Outer Iteration: Dual step + Communication

Algorithm Design: xFILTER

(S1). Assign parameters β and ρ.
(S2). Initialize $\mathbf{x}^{-1} = 0, \lambda^{-1} = 0$.
(S3) [Outer Iter.]. At iteration $s+1$, $s \geq -1$, define

$$\mathbf{R} := \rho \mathbf{A}^T \mathbf{A} + \beta \mathbf{I} \succ 0, \quad \mathbf{d}^s := -\nabla f(\mathbf{x}^s) - \mathbf{A}^T \lambda^s + \beta \mathbf{x}^s. \tag{1}$$

(S4) [Inner Iter.]. Run Q **Chebyshev iterations** (with parameters $\{\alpha_t, \tau\}$)

$$\mathbf{u}_0 = \mathbf{x}^s, \ \mathbf{u}_1 = (\mathbf{I} - \tau \mathbf{R})\mathbf{u}_0 + \tau \mathbf{d}^s, \tag{2}$$
$$\mathbf{u}_t = \underbrace{\alpha_t(\mathbf{I} - \tau \mathbf{R})\mathbf{u}_{t-1} + (1-\alpha_t)\mathbf{u}_{t-2} + \tau \alpha_t \mathbf{d}^s}_{\text{distributed implementable!}}, \ t = 2, \cdots, Q.$$

Inner Iteration: Approximate solve linear systems

(S5). [Outer Iter.] Set $\mathbf{x}^{s+1} = \mathbf{u}_Q$, update λ^{s+1} according to

$$\lambda^{s+1} = \lambda^s + \rho \mathbf{A} \mathbf{x}^{s+1} \tag{3}$$

Outer Iteration: Dual step + Communication

Algorithm Design: xFILTER

(S1). Assign parameters β and ρ.
(S2). Initialize $\mathbf{x}^{-1} = 0, \lambda^{-1} = 0$.
(S3) [Outer Iter.]. At iteration $s+1$, $s \geq -1$, define

$$\mathbf{R} := \rho \mathbf{A}^T \mathbf{A} + \beta \mathbf{I} \succ 0, \quad \mathbf{d}^s := -\nabla f(\mathbf{x}^s) - \mathbf{A}^T \lambda^s + \beta \mathbf{x}^s. \tag{1}$$

(S4) [Inner Iter.]. Run Q **Chebyshev iterations** (with parameters $\{\alpha_t, \tau\}$)

$$\begin{aligned} &\mathbf{u}_0 = \mathbf{x}^s, \ \mathbf{u}_1 = (\mathbf{I} - \tau \mathbf{R})\mathbf{u}_0 + \tau \mathbf{d}^s, \\ &\mathbf{u}_t = \underbrace{\alpha_t(\mathbf{I} - \tau \mathbf{R})\mathbf{u}_{t-1} + (1-\alpha_t)\mathbf{u}_{t-2} + \tau \alpha_t \mathbf{d}^s}_{\text{distributed implementable!}}, \ t = 2, \cdots, Q. \end{aligned} \tag{2}$$

(S5). [Outer Iter.] Set $\mathbf{x}^{s+1} = \mathbf{u}_Q$, update λ^{s+1} according to

$$\lambda^{s+1} = \lambda^s + \rho \mathbf{A} \mathbf{x}^{s+1} \tag{3}$$

Outer Iteration: Dual step + Communication

Algorithm Design: xFILTER

(S1). Assign parameters β and ρ.
(S2). Initialize $\mathbf{x}^{-1} = 0, \lambda^{-1} = 0$.
(S3) [Outer Iter.]. At iteration $s+1$, $s \geq -1$, define

$$\mathbf{R} := \rho \mathbf{A}^T \mathbf{A} + \beta \mathbf{I} \succ 0, \quad \mathbf{d}^s := -\nabla f(\mathbf{x}^s) - \mathbf{A}^T \lambda^s + \beta \mathbf{x}^s. \tag{1}$$

(S4) [Inner Iter.]. Run Q **Chebyshev iterations** (with parameters $\{\alpha_t, \tau\}$)

$$\begin{aligned} &\mathbf{u}_0 = \mathbf{x}^s, \ \mathbf{u}_1 = (\mathbf{I} - \tau \mathbf{R})\mathbf{u}_0 + \tau \mathbf{d}^s, \\ &\mathbf{u}_t = \underbrace{\alpha_t(\mathbf{I} - \tau \mathbf{R})\mathbf{u}_{t-1} + (1-\alpha_t)\mathbf{u}_{t-2} + \tau \alpha_t \mathbf{d}^s}_{\text{distributed implementable!}}, \ t = 2, \cdots, Q. \end{aligned} \tag{2}$$

(S5). **[Outer Iter.]** Set $\mathbf{x}^{s+1} = \mathbf{u}_Q$, update λ^{s+1} according to

$$\lambda^{s+1} = \lambda^s + \rho \mathbf{A} \mathbf{x}^{s+1} \tag{3}$$

Comments: On C-Iteration

- $\mathbf{x}^{s+1}$ generated lies in the following Krylov space

$$\text{span}\{\mathbf{d}^s, \mathbf{R}\mathbf{d}^s, \cdots, \mathbf{R}^Q\mathbf{d}^s\}$$

- Recall $\mathbf{R} = \rho\mathbf{A}^T\mathbf{A} + \beta\mathbf{I}$, and $\mathbf{A}^T\mathbf{A}$ is the (unnormalized) graph Laplacian matrix;

$$[\mathbf{A}^T\mathbf{A}]_{ij} = \begin{cases} d_i & \text{if } i = j \\ -1 & \text{if } (i,j) \in \mathcal{E}, i \neq j \\ 0 & \text{otherwise.} \end{cases}$$

- The operation $\mathbf{A}^T\mathbf{A}\mathbf{x}^s$ can be done by performing one round of message exchange among the neighbors

Comments: On Error

- Q Chebyshev iterations are performed, and the obtained $\mathbf{x}^{s+1}$ satisfies

$$\mathbf{R}\mathbf{x}^{s+1} = \mathbf{d}^s + \mathbf{R}\epsilon^{s+1},\ \forall\ s \geq -1 \tag{1}$$

where ϵ^{s+1} denotes the residual error.

- Let $\mathbf{R}\mathbf{x}_* = \mathbf{d}^s$; To achieve

$$\|\epsilon^{s+1}\|^2 = \|\mathbf{x}^{s+1} - \mathbf{x}_*\|^2 \leq \eta\|\mathbf{x}^s - \mathbf{x}_*\|^2 \tag{2}$$

for some $\eta > 0$, it requires Q iterations [Tsynkov07]

$$Q \geq \frac{1}{4}\ln(4/\eta)\sqrt{1/\xi(\mathbf{R})}.$$

Comments: On Error

- Using C-iteration ensures fast reduction of the error
- **Key Relation 1**: Number of C-iteration is proportional to $\sqrt{1/\xi(\mathbf{R})}$, rather than $1/\xi(\mathbf{R})$
  - Essential for the optimal performance
  - For path graph, total communication reduces from $\mathcal{O}(M^2/\epsilon)$ to $\mathcal{O}(M/\epsilon)$
- **Key Relation 2**: By properly choosing η, we have

$$\|\epsilon^{s+1}\|^2 \leq \|\mathbf{x}^{s+1} - \mathbf{x}^s\|^2$$

Only requires finite number Q inner iterations (independent of ϵ); the inner solution gets better and better as the outer iteration converges

Convergence Result and Tightness

Claim 2. [Sun-Hong 18] Fix a graph $\mathcal{N}_M$. To achieve $h^*(x^s) \leq \epsilon$, it requires at most the following number of iterations (T denotes the *total* iterations)

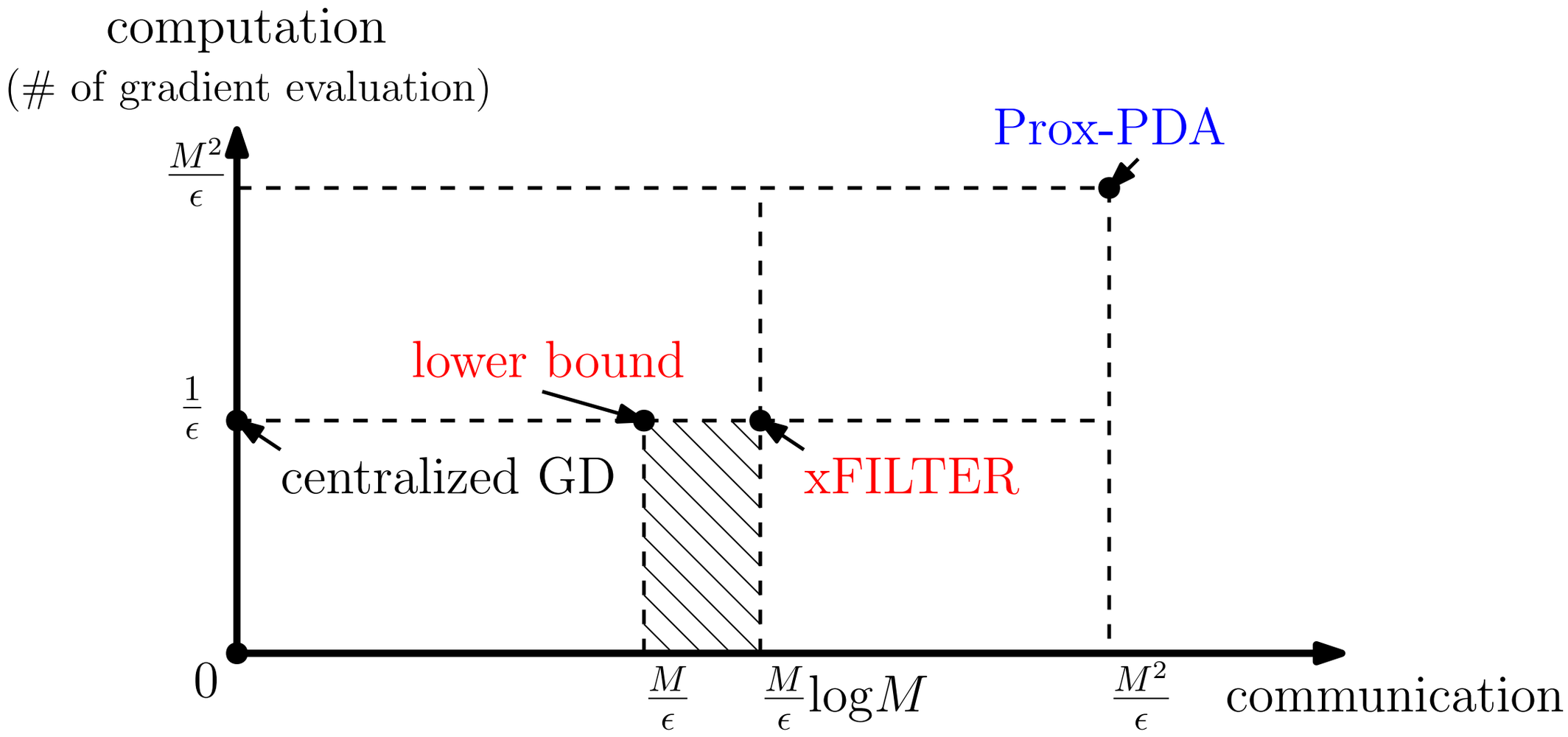
$$T \leq \underbrace{\frac{1}{\epsilon}\left(f(x^0) - \underline{f} + \frac{5}{M}\|d_0\|^2_{L^{-1}}\right) \times C_1}_{\text{outer iteration}} \times \underbrace{\frac{1}{4}\log(C_2)\sqrt{50/\xi(\mathcal{L})}}_{\text{inner iterations } Q}$$

where C_1 and C_2 are given by

$$C_1 \leq 128\left(\frac{96}{M}\sum_{i=1}^{M} L_i + 19\right)$$

$$C_2 \leq \left(\frac{50^2(L_{\max}/L_{\min})^4 \times (16 + 128M\max\{50 \times 96L_{\max}/M, 1\})}{\xi^3(\mathcal{L}) \times \min\{96^2 L^2_{\min}/M^2, 1\}}\right).$$

Summary: Main Results



- Illustration of the main results using a path graph

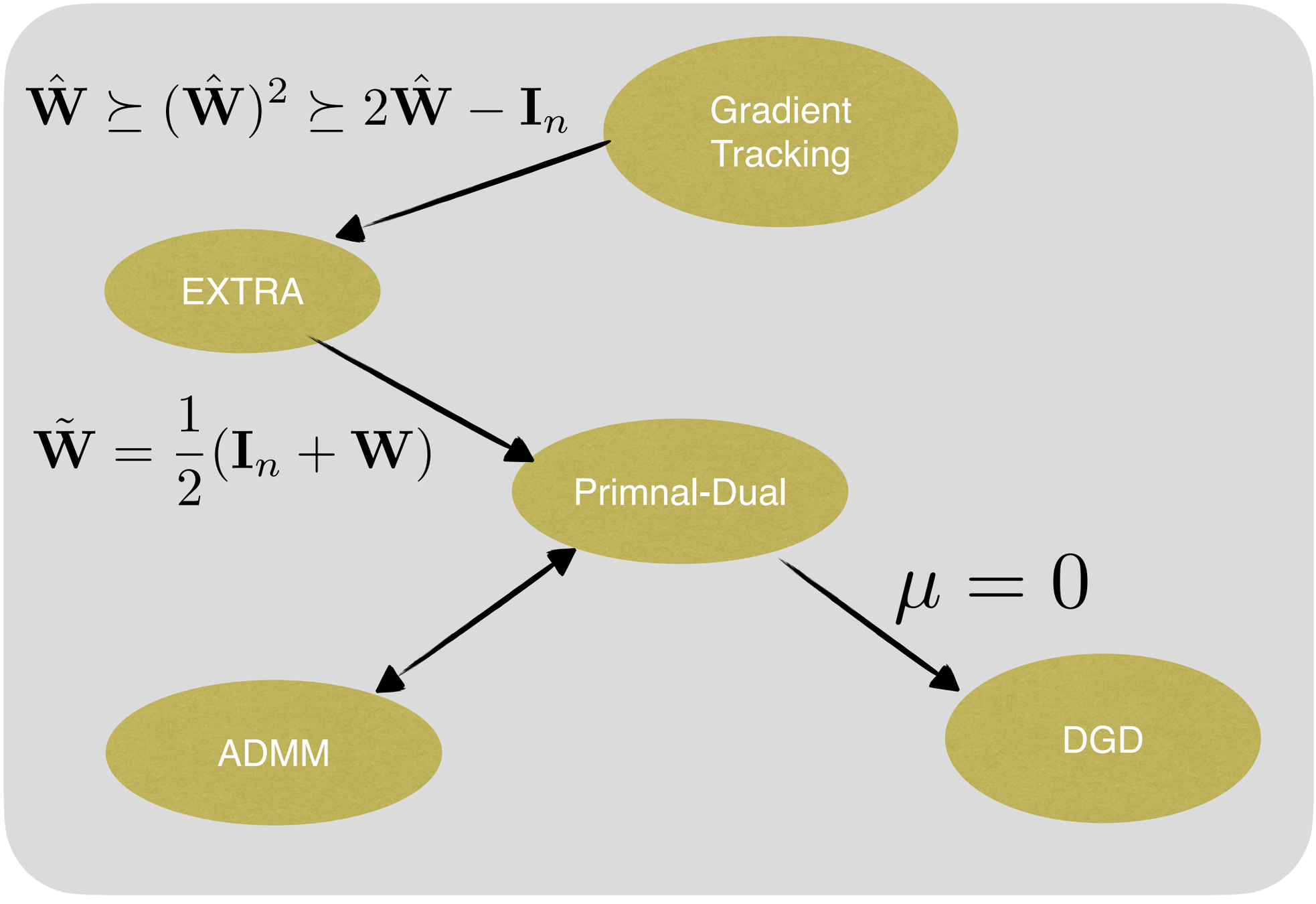
- Note that in the proposed methods, only outer iteration requires gradient evaluation

Summary: Graph Dependency

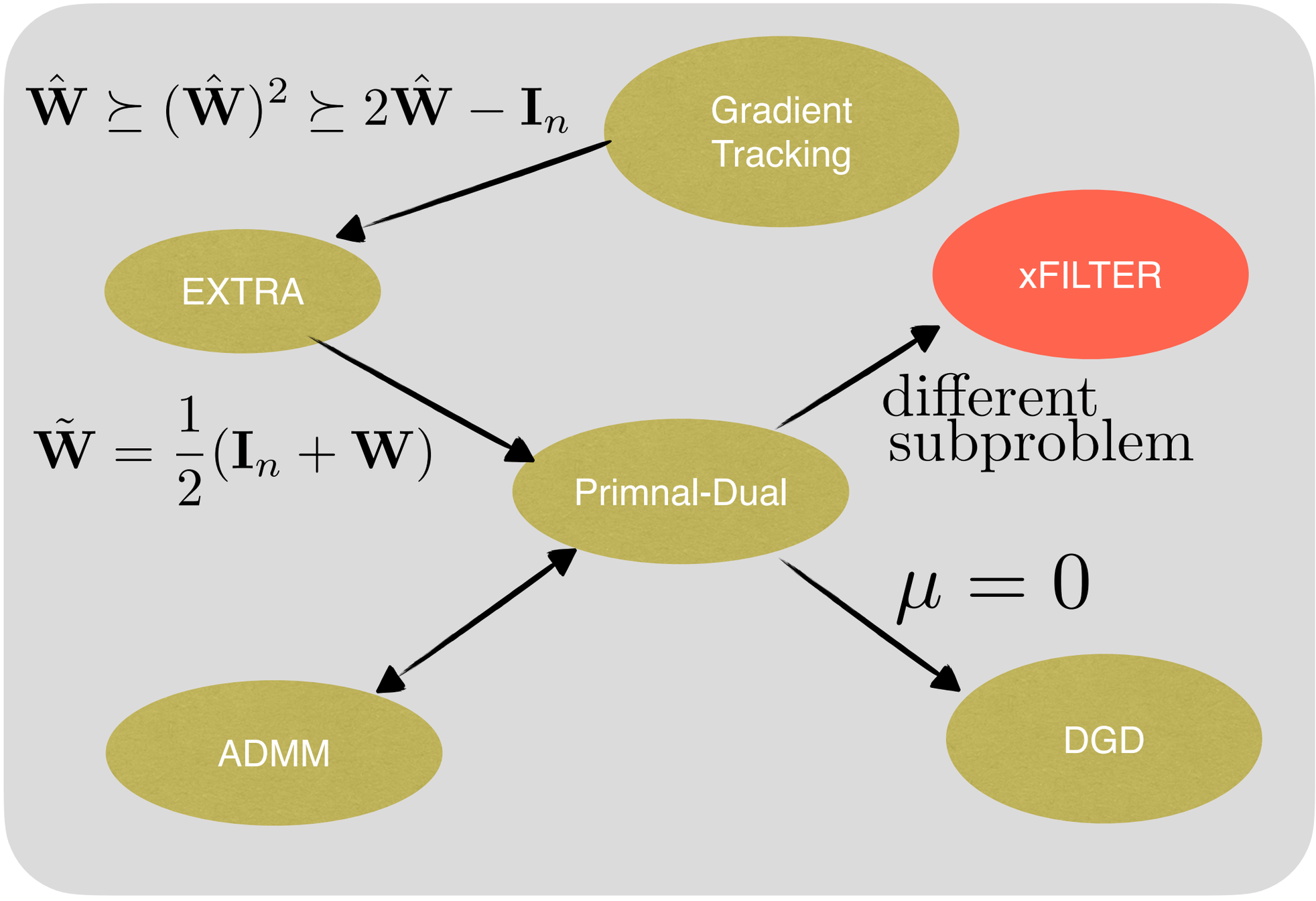
$$\min_{\mathbf{x}\in\mathbb{R}^{MP}} \quad f(\mathbf{x}) = \frac{1}{M}\sum_{i=1}^{M} f_i(\mathbf{x}_i), \quad \text{s.t.} \quad \mathbf{x}_i = \mathbf{x}_k, \quad \forall (i,k) \in \mathcal{E}.$$

	Computation Complexity	Communication Complexity
Existing Work [Hong et al., 2016] [Sun-Scutari 2018]	$\mathcal{O}\left(\frac{1}{\xi(\mathcal{L})} \times \frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{1}{\xi(\mathcal{L})} \times \frac{1}{\epsilon}\right)$
Lower Bound [This Work]	$\mathcal{O}\left(\frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{1}{\sqrt{\xi(\mathcal{L})}} \times \frac{1}{\epsilon}\right)$
xFILTER [This Work]	$\mathcal{O}\left(\frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\frac{1}{\sqrt{\xi(\mathcal{L})}} \times \frac{1}{\epsilon}\right)$

Summary: Main Results

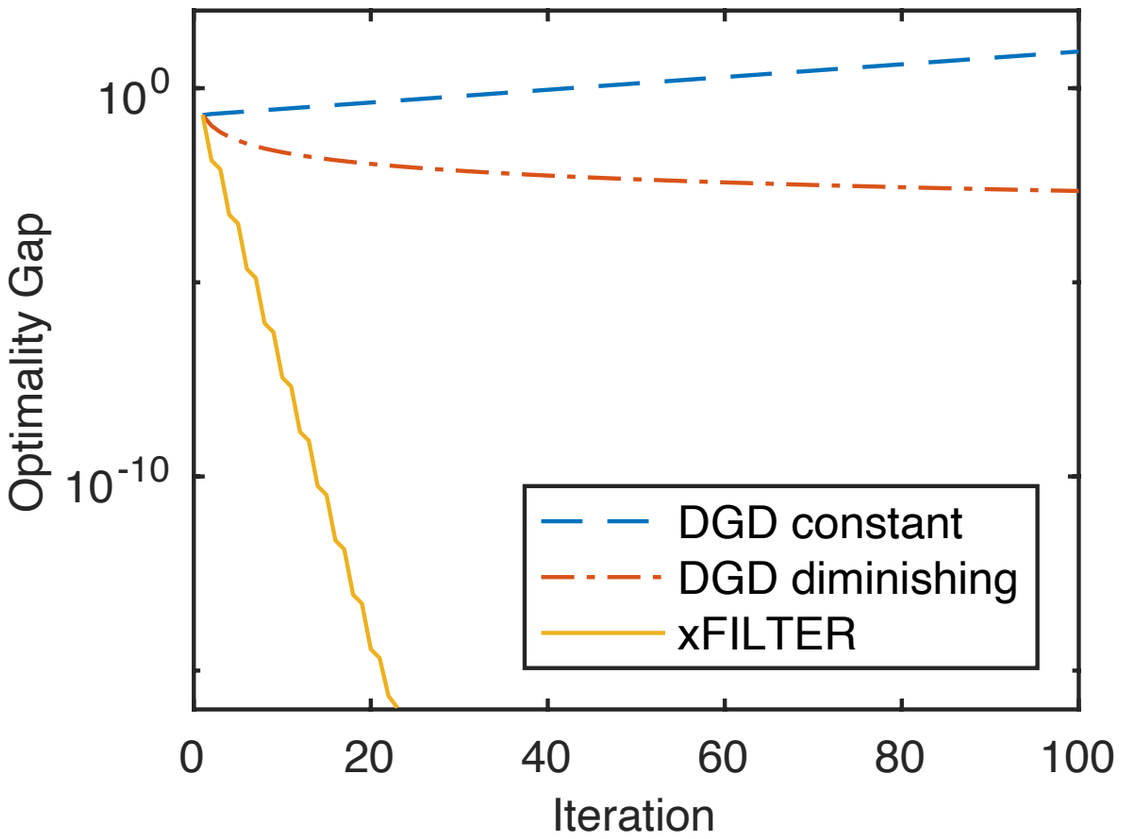


Summary: Main Results

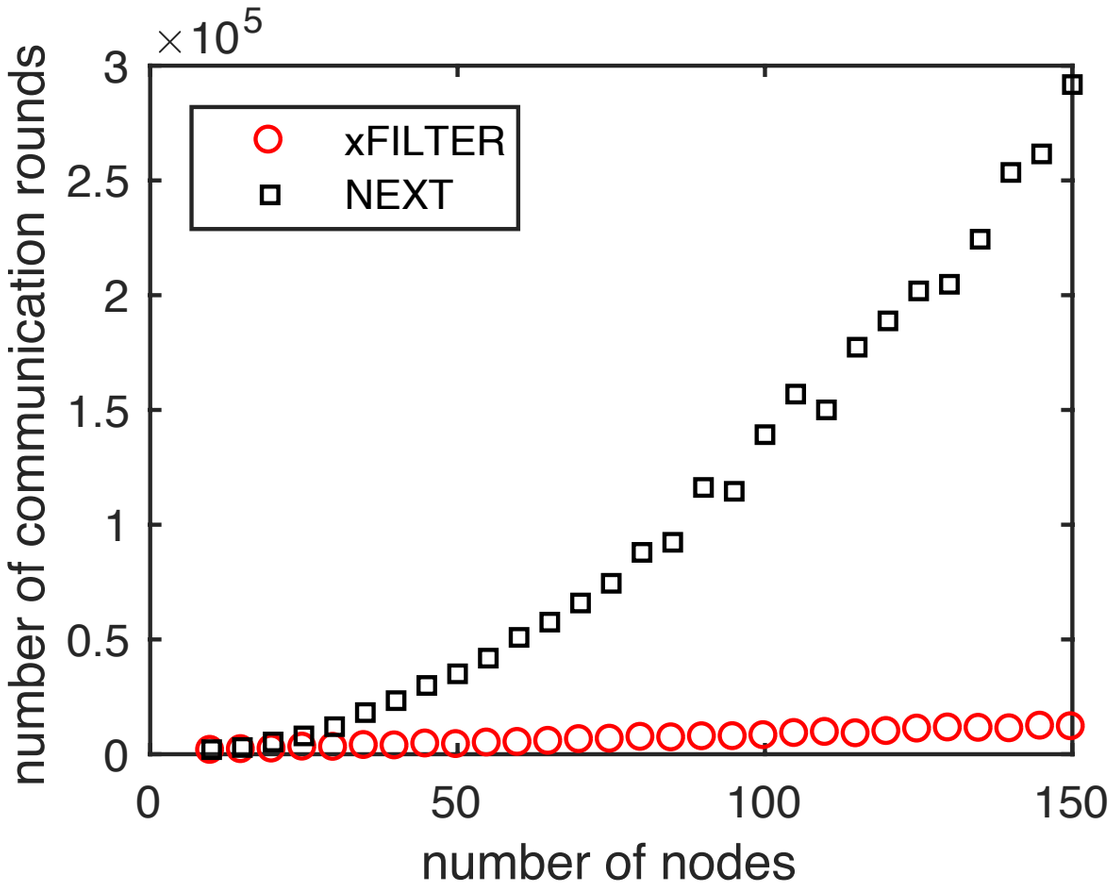


Numerical Results

Numerical Results: Examples



For Quadratic counter-example
xFILTER converges with
constant step-size



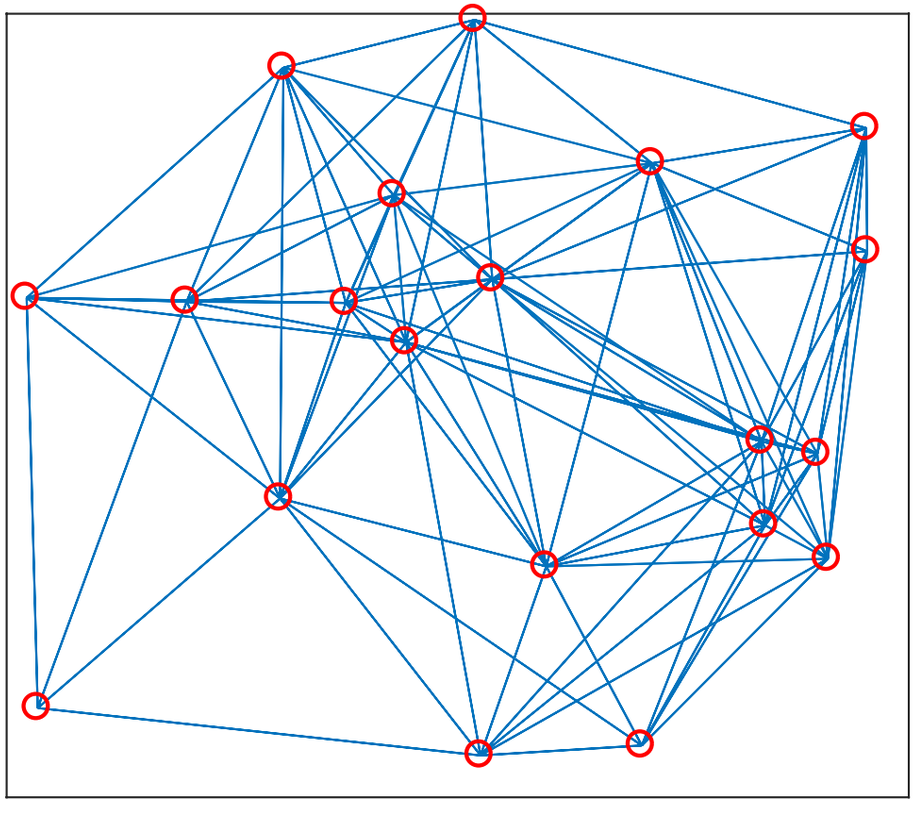
For line graphs
xFILTER scales linearly
with number of nodes

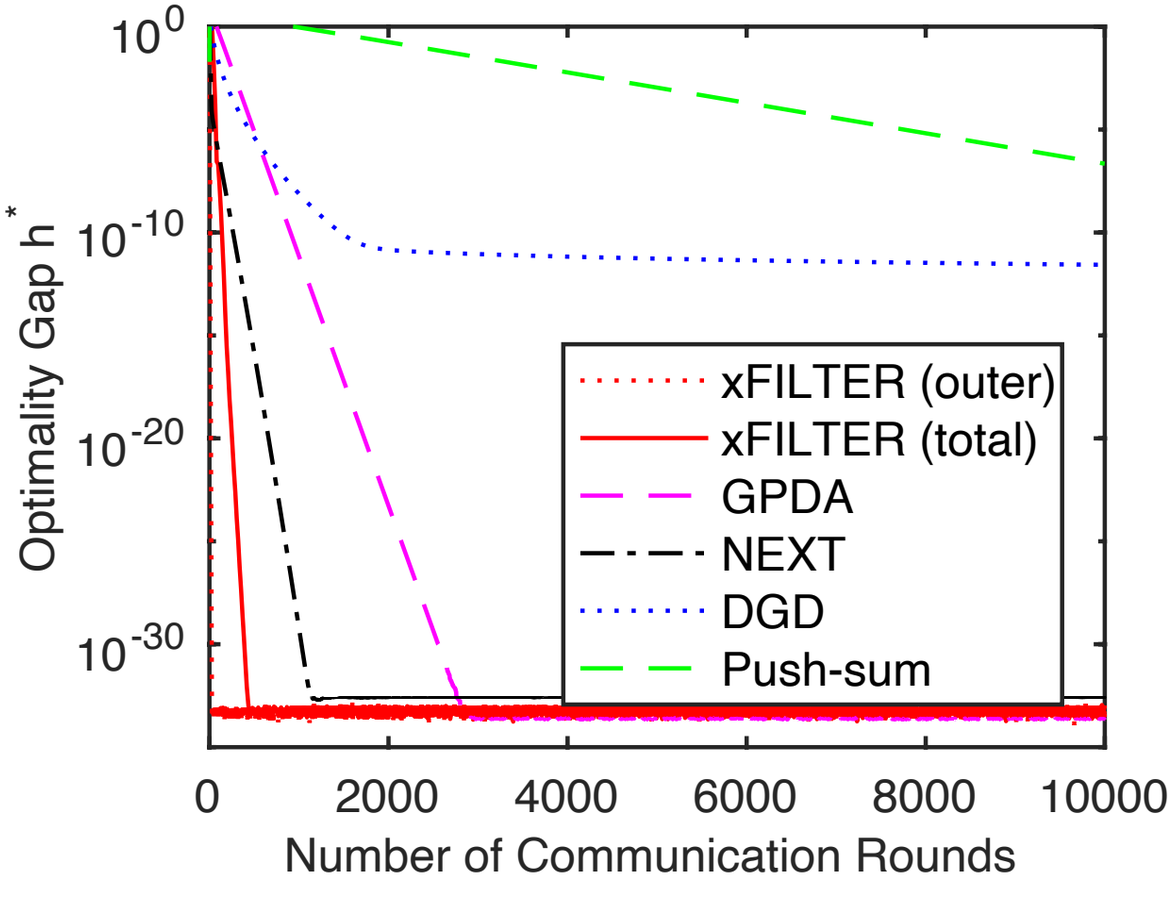
Numerical Results: Binary Classification

Distributed binary classification problem [Antoniadis et al, 11]

$$f_i(\mathbf{x}_i) = \frac{1}{MJ}\left[\sum_{j=1}^{J} \log\left(1+\exp(-\mathbf{y}_{ij}\mathbf{x}_i^T\mathbf{v}_{ij})\right) + \sum_{k=1}^{K} \frac{\lambda\alpha\mathbf{x}_{i,k}^2}{1+\alpha\mathbf{x}_{i,k}^2}\right]$$

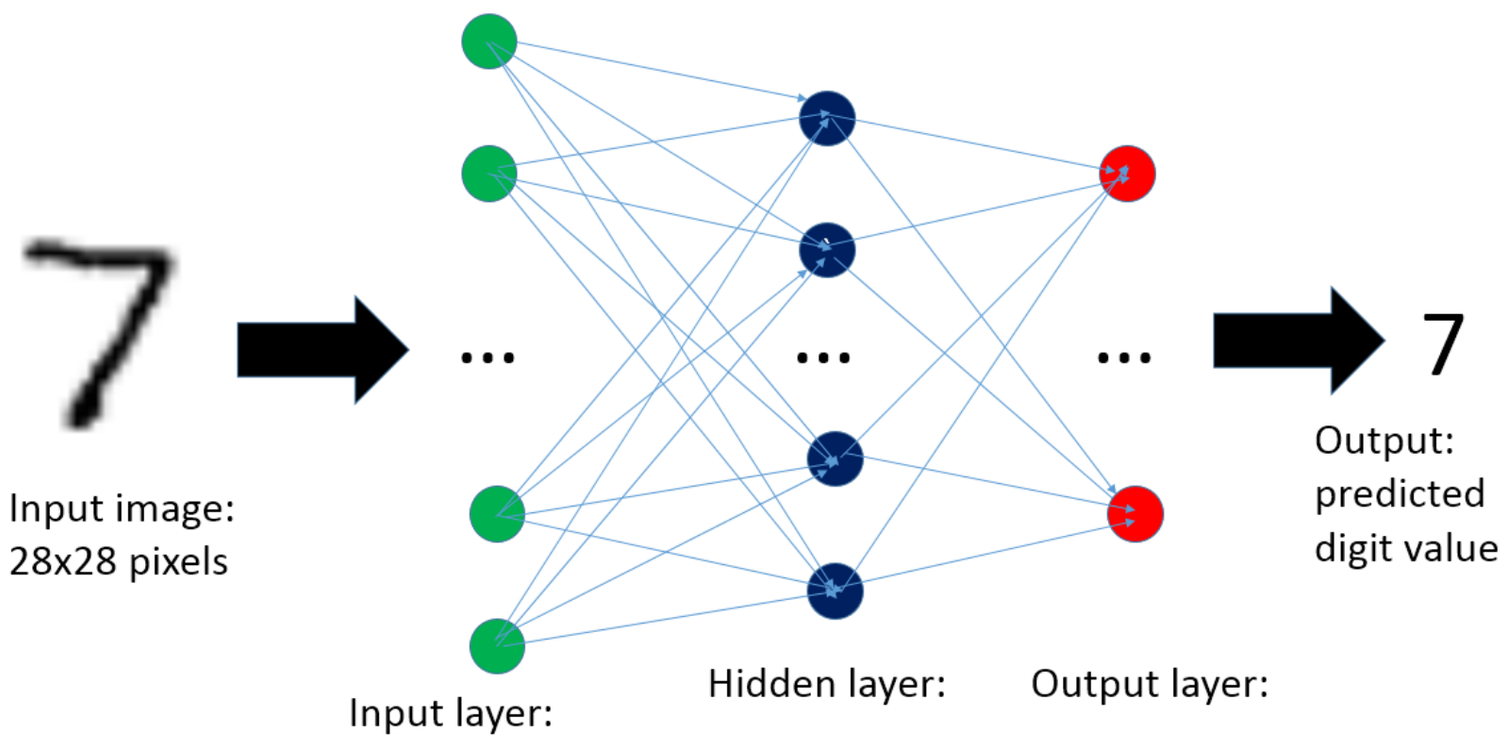
Compared with NEXT [Di Lorenzo-Scutari, 16], Prox-PDA [Hong et al, 17], DGD [Nedic´ et al, 09], and Push-sum [Tatarenko-Touri, 17]





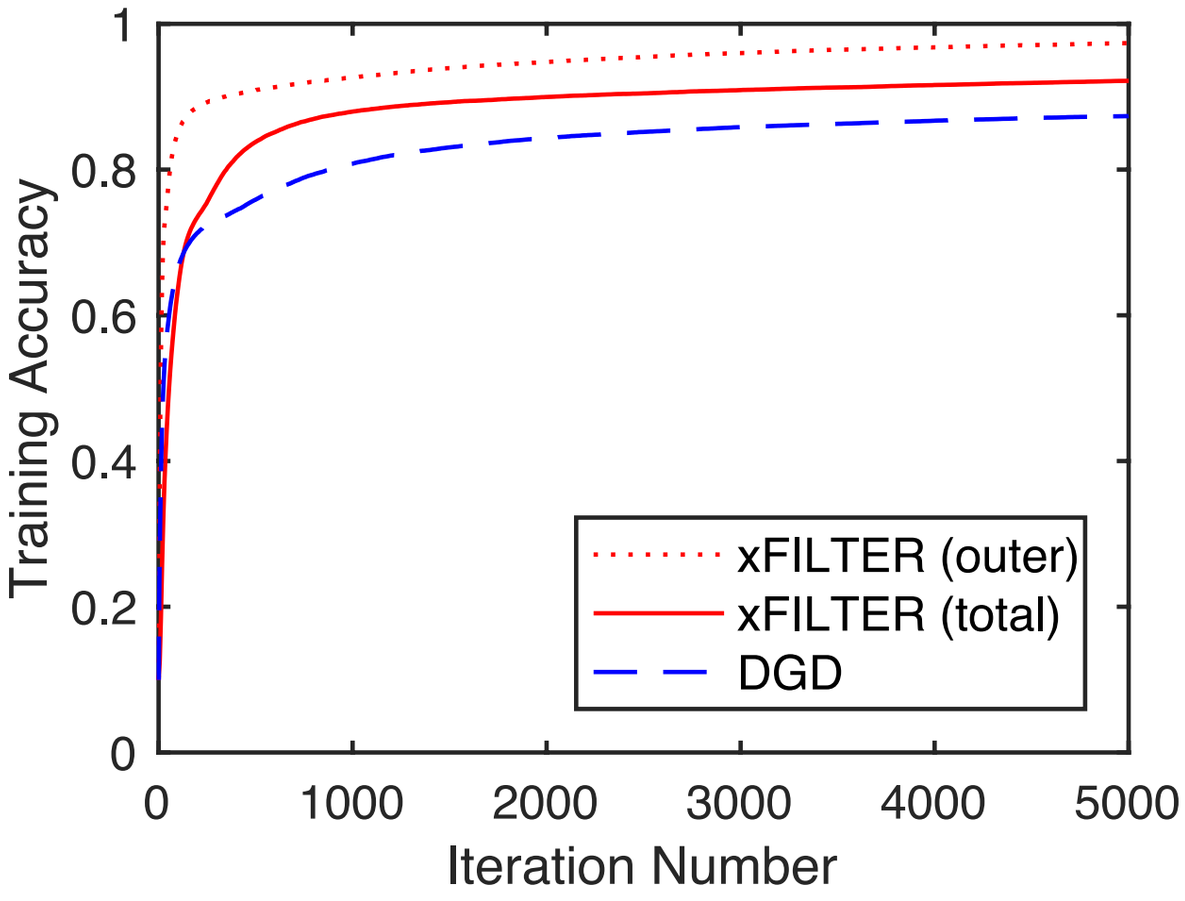
xFILTER converges faster than state-of-the-art methods

Numerical Results: Neural Networks

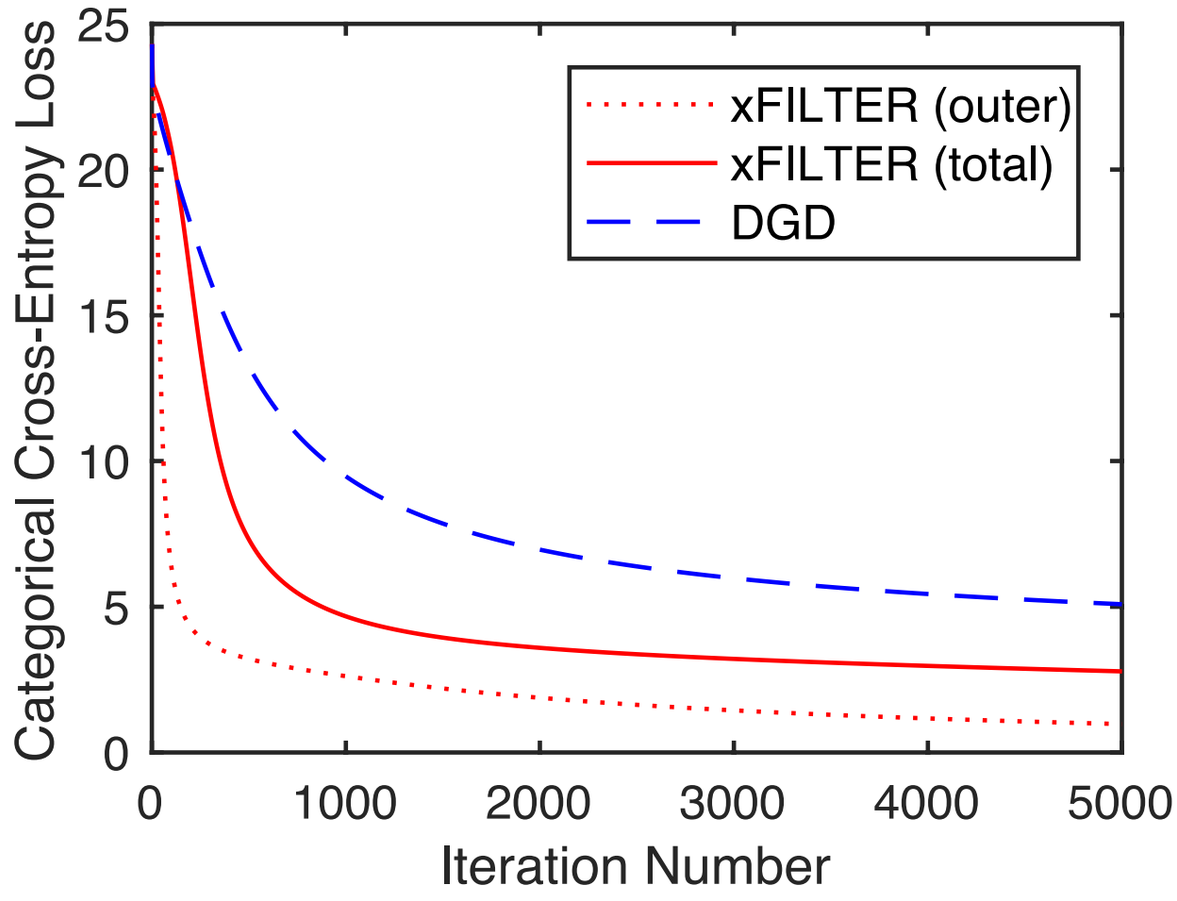


- Message Passing Interface (MPI) + TensorFlow
- One hidden layer fully connected neural network (with 128 neurons)
- MNIST dataset over 10 machines

Numerical Results: Neural Networks



(a) Training Loss



(b) Training Accuracy

Performance comparison on decentralized training neural networks over path graphs

Recent Developments & Open Problems

Empirical Risk Minimization

In machine learning problems, local cost functions usually depend on samples

$$f_i(\mathbf{x}_i) = \frac{1}{N}\sum_{j=1}^{N} f_i^j(\mathbf{x}_i), \forall i$$

Specialized problem: *M* agents, each contains *N* data points

$$\min_{\mathbf{x}\in\mathbb{R}^{MP}} f(\mathbf{x}) = \frac{1}{MN}\sum_{i=1}^{M} \underbrace{\sum_{j=1}^{N} f_i^j(\mathbf{x}_i)}_{\text{local samples}} \quad , \quad \text{s.t.} \quad \mathbf{x}_i = \mathbf{x}_k, \quad \forall (i,k) \in \mathcal{E}$$

New Models?

- All local data is available at the same time?
- **Data model**: Data are “streaming in”, each time only observe mini-batches of data
- **Computation model**: local stochastic gradients $\{g_i^r\}$ can be computed
- **Communication model:** each node talk to its neighbors
- Focus on the regime with

$$N \gg M$$

Sample Complexity

Consider an *M*-node problem, each contains *N* data points

$$\min_{\mathbf{x}\in\mathbb{R}^{MP}} f(\mathbf{x}) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} f_i^j(\mathbf{x}_i), \quad \text{s.t.} \quad \mathbf{x}_i = \mathbf{x}_k, \quad \forall (i,k) \in \mathcal{E}$$

Sample Complexity

Consider an *M*-node problem, each contains *N* data points

$$\min_{\mathbf{x}\in\mathbb{R}^{MP}} f(\mathbf{x}) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} f_i^j(\mathbf{x}_i), \quad \text{s.t.} \quad \mathbf{x}_i = \mathbf{x}_k, \quad \forall (i,k) \in \mathcal{E}$$

In this case, computational complexity becomes:

Sample Complexity: Total number of samples $j \in [N]$ are required across the entire network, to evaluate $(f^j(\mathbf{w}), \nabla f^j(\mathbf{w}))$, so to achieve an ϵ stationary solution.

Sample Complexity

Consider an *M*-node problem, each contains *N* data points

$$\min_{\mathbf{x}\in\mathbb{R}^{MP}} f(\mathbf{x}) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} f_i^j(\mathbf{x}_i), \quad \text{s.t.} \quad \mathbf{x}_i = \mathbf{x}_k, \quad \forall (i,k) \in \mathcal{E}$$

In this case, computational complexity becomes:

Sample Complexity: Total number of samples $j \in [N]$ are required across the entire network, to evaluate $(f^j(\mathbf{w}), \nabla f^j(\mathbf{w}))$, so to achieve an ϵ stationary solution.

Directly applying xFILTER we will have

$$\mathcal{O}\left(MN \times \frac{1}{\epsilon}\right)$$

Sample Complexity

Consider an *M*-node problem, each contains *N* data points

$$\min_{\mathbf{x}\in\mathbb{R}^{MP}} f(\mathbf{x}) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} f_i^j(\mathbf{x}_i), \quad \text{s.t.} \quad \mathbf{x}_i = \mathbf{x}_k, \quad \forall (i,k) \in \mathcal{E}$$

In this case, computational complexity becomes:

Sample Complexity: Total number of samples $j \in [N]$ are required across the entire network, to evaluate $(f^j(\mathbf{w}), \nabla f^j(\mathbf{w}))$, so to achieve an ϵ stationary solution.

Directly applying xFILTER we will have

$$\mathcal{O}\left(MN \times \frac{1}{\epsilon}\right)$$

Can we do better?

Stochastic DGD

Algorithm Updates [Lian et al., 17][Liang et al 17]

$$\mathbf{x}^{r+1} = \mathbf{W}\mathbf{x}^r - \alpha^r \mathbf{g}^r$$

Convergence rate is $\mathcal{O}(1/\epsilon^2)$

Additional Assumptions homogeneity of data:

$$\sum_{I=1}^{M} \|\nabla f_i(\mathbf{x}) - \nabla f(\mathbf{x})\|^2 \leq \xi < \infty$$

Other related DGD based methods also requires this, e.g., the gradient push [Assran 19]

GT-Based Stochastic Method

Algorithm Updates [Lu et al., 18]

$$\mathbf{x}^{r+1} = \mathbf{W}\mathbf{x}^r - \alpha^r \mathbf{y}^r$$
$$\mathbf{y}^{r+1} = \mathbf{W}\mathbf{y}^r + (\mathbf{g}^{r+1} - \mathbf{g}^r)$$

Convergence rate is $\mathcal{O}(1/\epsilon^2)$

No additional Assumptions needed

Other related algorithms: D2 [Tang et al., 18], etc

[1] Tang, Hanlin, Xiangru Lian, Ming Yan, Ce Zhang, and Ji Liu. "D $^ 2$: Decentralized Training over Decentralized Data." *arXiv preprint arXiv:1803.07068* (2018).

Sample Complexity Bounds

Consider a M node problem, each contains N data points

$$\min_{\mathbf{x}\in\mathbb{R}^{MP}} f(\mathbf{x}) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} f_i^j(\mathbf{x}_i), \quad \text{s.t.} \quad \mathbf{x}_i = \mathbf{x}_k, \quad \forall (i,k) \in \mathcal{E}$$

	Sample Complexity	Communication Complexity
Lower Bound	$\mathcal{O}(\sqrt{MN}/\epsilon)$ **[Fang et al, 18] Centralized method**	$\mathcal{O}(1/\epsilon)$ **[Sun-Hong, 18]**
Upper Bound (deterministic)	$\mathcal{O}(MN/\epsilon)$ **[Sun-Scutari 2018][Sun-H., 18]**	$\mathcal{O}(1/\epsilon)$ **[Sun-Scutari 2018][Sun-H., 18]**
Upper Bound (stochastic)	$\mathcal{O}(M/\epsilon^2)$ **[Tang et al 18][Lu et al 18]**	$\mathcal{O}(1/\epsilon^2)$ **[Tang et al 18][Lu et al 18]**

Sample Complexity Bounds

Consider a M node problem, each contains N data points

$$\min_{\mathbf{x}\in\mathbb{R}^{MP}} f(\mathbf{x}) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} f_i^j(\mathbf{x}_i), \quad \text{s.t.} \quad \mathbf{x}_i = \mathbf{x}_k, \quad \forall (i,k) \in \mathcal{E}$$

	Sample Complexity	Communication Complexity
Lower Bound	$\mathcal{O}(\sqrt{MN}/\epsilon)$ [Fang et al, 18] Centralized method	$\mathcal{O}(1/\epsilon)$ [Sun-Hong, 18]
Upper Bound (deterministic)	$\mathcal{O}(MN/\epsilon)$ [Sun-Scutari 2018][Sun-H., 18]	$\mathcal{O}(1/\epsilon)$ [Sun-Scutari 2018][Sun-H., 18]
Upper Bound (stochastic)	$\mathcal{O}(M/\epsilon^2)$ [Tang et al 18][Lu et al 18]	$\mathcal{O}(1/\epsilon^2)$ [Tang et al 18][Lu et al 18]

Bridge the Gap?

Reducing complexity

- Possible to efficiently utilize local samples?
- In [Sun-Lu-H.19], an algorithm for achieving the following

$$\mathcal{O}(MN + MN^{1/2}\epsilon^{-1}).$$

- Combine variance reduced methods [Fang et al 19][Wang et al 18] And gradient tracking based methods

H. Sun, S. Lu and M. Hong, “Improving the sample and communication complexity for decentralized non-convex optimization: A joint gradient estimation and tracking approach”, ICML 2020.

Centralized Non-Convex Optimization

Consider a one node problem with N data points

$$\min_{\mathbf{w}\in\mathbb{R}^d} h(\mathbf{w}) = \frac{1}{N}\sum_{j=1}^{N} f^j(\mathbf{w})$$

$$\|\frac{1}{N}\sum_{j=1}^{N} \nabla f^j(\mathbf{w})\|^2 \leq \epsilon$$

Centralized Non-Convex Optimization

Consider a one node problem with N data points

$$\min_{\mathbf{w}\in\mathbb{R}^d} h(\mathbf{w}) = \frac{1}{N}\sum_{j=1}^{N} f^j(\mathbf{w})$$

$$\|\frac{1}{N}\sum_{j=1}^{N} \nabla f^j(\mathbf{w})\|^2 \leq \epsilon$$

GD [Nesterov 03]

$$\mathbf{w}^{r+1} = \mathbf{w}^r - \alpha\frac{1}{N}\sum_{j=1}^{N} \nabla f^j(\mathbf{w}^r)$$

Centralized Non-Convex Optimization

Consider a one node problem with N data points

$$\min_{\mathbf{w}\in\mathbb{R}^d} h(\mathbf{w}) = \frac{1}{N}\sum_{j=1}^{N} f^j(\mathbf{w})$$

$$\|\frac{1}{N}\sum_{j=1}^{N} \nabla f^j(\mathbf{w})\|^2 \leq \epsilon$$

GD [Nesterov 03]

$$\mathbf{w}^{r+1} = \mathbf{w}^r - \alpha\frac{1}{N}\sum_{j=1}^{N} \nabla f^j(\mathbf{w}^r)$$

$$\mathcal{O}(N/\epsilon)$$

Centralized Non-Convex Optimization

Consider a one node problem with N data points

$$\min_{\mathbf{w}\in\mathbb{R}^d} h(\mathbf{w}) = \frac{1}{N}\sum_{j=1}^{N} f^j(\mathbf{w})$$

$$\|\frac{1}{N}\sum_{j=1}^{N} \nabla f^j(\mathbf{w})\|^2 \leq \epsilon$$

GD [Nesterov 03]

$$\mathbf{w}^{r+1} = \mathbf{w}^r - \alpha\frac{1}{N}\sum_{j=1}^{N} \nabla f^j(\mathbf{w}^r)$$

SGD [Ghadimi & Lan 13]

$$\mathbf{w}^{r+1} = \mathbf{w}^r - \alpha\nabla f^{\xi}(\mathbf{w}^r)$$

$$\mathcal{O}(N/\epsilon)$$

Centralized Non-Convex Optimization

Consider a one node problem with N data points

$$\min_{\mathbf{w}\in\mathbb{R}^d} h(\mathbf{w}) = \frac{1}{N}\sum_{j=1}^{N} f^j(\mathbf{w})$$

$$\|\frac{1}{N}\sum_{j=1}^{N} \nabla f^j(\mathbf{w})\|^2 \leq \epsilon$$

GD [Nesterov 03]

$$\mathbf{w}^{r+1} = \mathbf{w}^r - \alpha\frac{1}{N}\sum_{j=1}^{N} \nabla f^j(\mathbf{w}^r)$$

$$\mathcal{O}(N/\epsilon)$$

SGD [Ghadimi & Lan 13]

$$\mathbf{w}^{r+1} = \mathbf{w}^r - \alpha\nabla f^{\xi}(\mathbf{w}^r)$$

$$\mathcal{O}(1/\epsilon^2)$$

Centralized Non-Convex Optimization

Consider a one node problem with N data points

$$\min_{\mathbf{w}\in\mathbb{R}^d} h(\mathbf{w}) = \frac{1}{N}\sum_{j=1}^{N} f^j(\mathbf{w})$$

$$\|\frac{1}{N}\sum_{j=1}^{N} \nabla f^j(\mathbf{w})\|^2 \leq \epsilon$$

GD [Nesterov 03]

$$\mathbf{w}^{r+1} = \mathbf{w}^r - \alpha\frac{1}{N}\sum_{j=1}^{N} \nabla f^j(\mathbf{w}^r)$$

SGD [Ghadimi & Lan 13]

$$\mathbf{w}^{r+1} = \mathbf{w}^r - \alpha\nabla f^{\xi}(\mathbf{w}^r)$$

$\mathcal{O}(N/\epsilon)$  $\mathcal{O}(1/\epsilon^2)$

Centralized Non-Convex Optimization

Consider a one node problem with N data points

$$\min_{\mathbf{w}\in\mathbb{R}^d} h(\mathbf{w}) = \frac{1}{N}\sum_{j=1}^{N} f^j(\mathbf{w})$$

$$\|\frac{1}{N}\sum_{j=1}^{N} \nabla f^j(\mathbf{w})\|^2 \leq \epsilon$$

GD [Nesterov 03]

$$\mathbf{w}^{r+1} = \mathbf{w}^r - \alpha\frac{1}{N}\sum_{j=1}^{N} \nabla f^j(\mathbf{w}^r)$$

SGD [Ghadimi & Lan 13]

$$\mathbf{w}^{r+1} = \mathbf{w}^r - \alpha\nabla f^{\xi}(\mathbf{w}^r)$$

$\mathcal{O}(N/\epsilon)$  $\mathcal{O}(1/\epsilon^2)$

Variance Reduction (VR)

[Fang et al, 18]
[Nguyen et al, 19]

$\mathcal{O}(\sqrt{N}/\epsilon)$

Centralized Non-Convex Optimization

Consider a one node problem with N data points

$$\min_{\mathbf{w}\in\mathbb{R}^d} h(\mathbf{w}) = \frac{1}{N}\sum_{j=1}^{N} f^j(\mathbf{w})$$

$$\|\frac{1}{N}\sum_{j=1}^{N} \nabla f^j(\mathbf{w})\|^2 \leq \epsilon$$

GD [Nesterov 03]

$$\mathbf{w}^{r+1} = \mathbf{w}^r - \alpha\frac{1}{N}\sum_{j=1}^{N} \nabla f^j(\mathbf{w}^r)$$

SGD [Ghadimi & Lan 13]

$$\mathbf{w}^{r+1} = \mathbf{w}^r - \alpha\nabla f^{\xi}(\mathbf{w}^r)$$

$\mathcal{O}(N/\epsilon)$  $\mathcal{O}(1/\epsilon^2)$

Variance Reduction (VR)

[Fang et al, 18]
[Nguyen et al, 19]

$\mathcal{O}(\sqrt{N}/\epsilon)$  **Optimal** [Fang et al, 18]

Variance Reduction (VR)

Consider a one node problem with N data points

$$\min_{\mathbf{w}\in\mathbb{R}^d} h(\mathbf{w}) = \frac{1}{N}\sum_{j=1}^{N} f^j(\mathbf{w})$$

$$\mathbf{w}^{r+1} = \mathbf{w}^r - \alpha \mathbf{g}^r$$

★ **SGD:** $\mathbf{g}^r = \nabla f^{\xi}(\mathbf{w}^r)$

★ **High variance of SGD:** $\mathbb{E}[\|\nabla f^{\xi}(\mathbf{w}^r)\|^2] \leq \sigma^2$

★ σ^2 **is non-negligible even when** $\mathbf{w}^r = \mathbf{w}^*$

Q: How to design $\mathbf{g}^r$ with reduced variability?

$$\mathbb{E}[\|\mathbf{g}^r\|^2] < \mathbb{E}[\|\nabla f^{\xi}(\mathbf{w}^r)\|^2]$$

Variance Reduction (VR)

A: Find some $\mathbf{s}^r$ highly correlated with gradient.

$$\mathbf{g}^r = \nabla f^{\xi}(\mathbf{w}^r) - \mathbf{s}^r$$

★ **Suppose** $\mathbb{E}[\mathbf{s}^r] = \mathbf{0}$

★ **Unbiased estimation:** $\mathbb{E}[\mathbf{g}^r] = \mathbb{E}[\nabla f^{\xi}(\mathbf{w}^r)]$

★ **Reduced variability:**

$$\mathbb{E}[\|\mathbf{g}^r\|^2] = \mathbb{E}[\|\nabla f^{\xi}(\mathbf{w}^r)\|^2] + \mathbb{E}[\|\mathbf{s}^r\|^2] - 2\mathbb{E}\left\langle \nabla f^{\xi}(\mathbf{w}^r), \mathbf{s}^r \right\rangle$$

If current iterate is not too far away from previous iterates, then historical gradient info might be useful in producing such s to reduce variance

Aggregate previous gradient info to help improve convergence rate

Stochastic Variance Reduced Gradient (SVRG)

Key idea: If we have access to history point $\mathbf{w}^{\text{old}}$ and $\nabla h(\mathbf{w}^{\text{old}})$

$$\mathbf{g}^r = \nabla f^{\xi}(\mathbf{w}^r) - \nabla f^{\xi}(\mathbf{w}^{\text{old}}) + \nabla h(\mathbf{w}^{\text{old}})$$

[1] Johnson, Rie, and Tong Zhang. "Accelerating stochastic gradient descent using predictive variance reduction." In *Advances in neural information processing systems*, pp. 315-323. 2013.
[2] Reddi, Sashank J., Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. "Stochastic variance reduction for nonconvex optimization." In *International conference on machine learning*, pp. 314-323. 2016.

Stochastic Variance Reduced Gradient (SVRG)

Key idea: If we have access to history point $\mathbf{w}^{\text{old}}$ and $\nabla h(\mathbf{w}^{\text{old}})$

$$\mathbf{g}^r = \nabla f^{\xi}(\mathbf{w}^r) - \nabla f^{\xi}(\mathbf{w}^{\text{old}}) + \nabla h(\mathbf{w}^{\text{old}})$$

$$\mathbb{E}[\nabla f^{\xi}(\mathbf{w}^{\text{old}}) - \nabla h(\mathbf{w}^{\text{old}})] = \mathbf{0}$$

[1] Johnson, Rie, and Tong Zhang. "Accelerating stochastic gradient descent using predictive variance reduction." In *Advances in neural information processing systems*, pp. 315-323. 2013.
[2] Reddi, Sashank J., Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. "Stochastic variance reduction for nonconvex optimization." In *International conference on machine learning*, pp. 314-323. 2016.

Stochastic Variance Reduced Gradient (SVRG)

Key idea: If we have access to history point $\mathbf{w}^{\text{old}}$ and $\nabla h(\mathbf{w}^{\text{old}})$

$$\mathbf{g}^r = \nabla f^{\xi}(\mathbf{w}^r) - \nabla f^{\xi}(\mathbf{w}^{\text{old}}) + \nabla h(\mathbf{w}^{\text{old}})$$

$$\mathbb{E}[\nabla f^{\xi}(\mathbf{w}^{\text{old}}) - \nabla h(\mathbf{w}^{\text{old}})] = \mathbf{0}$$

★ **Unbiased estimation of** $\nabla h(\mathbf{w}^r)$

[1] Johnson, Rie, and Tong Zhang. "Accelerating stochastic gradient descent using predictive variance reduction." In *Advances in neural information processing systems*, pp. 315-323. 2013.
[2] Reddi, Sashank J., Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. "Stochastic variance reduction for nonconvex optimization." In *International conference on machine learning*, pp. 314-323. 2016.

Stochastic Variance Reduced Gradient (SVRG)

Key idea: If we have access to history point $\mathbf{w}^{\text{old}}$ and $\nabla h(\mathbf{w}^{\text{old}})$

$$\mathbf{g}^r = \nabla f^{\xi}(\mathbf{w}^r) - \nabla f^{\xi}(\mathbf{w}^{\text{old}}) + \nabla h(\mathbf{w}^{\text{old}})$$

$$\rightarrow \mathbf{0} \text{ if } \mathbf{w}^r \approx \mathbf{w}^{\text{old}}$$

$$\mathbb{E}[\nabla f^{\xi}(\mathbf{w}^{\text{old}}) - \nabla h(\mathbf{w}^{\text{old}})] = \mathbf{0}$$

★ **Unbiased estimation of** $\nabla h(\mathbf{w}^r)$

[1] Johnson, Rie, and Tong Zhang. "Accelerating stochastic gradient descent using predictive variance reduction." In *Advances in neural information processing systems*, pp. 315-323. 2013.
[2] Reddi, Sashank J., Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. "Stochastic variance reduction for nonconvex optimization." In *International conference on machine learning*, pp. 314-323. 2016.

Stochastic Variance Reduced Gradient (SVRG)

Key idea: If we have access to history point $\mathbf{w}^{\text{old}}$ and $\nabla h(\mathbf{w}^{\text{old}})$

$$\mathbf{g}^r = \nabla f^{\xi}(\mathbf{w}^r) - \nabla f^{\xi}(\mathbf{w}^{\text{old}}) + \nabla h(\mathbf{w}^{\text{old}})$$

$\rightarrow \mathbf{0}$ if $\mathbf{w}^r \approx \mathbf{w}^{\text{old}}$

$\rightarrow \mathbf{0}$ if $\mathbf{w}^{\text{old}} \approx \mathbf{w}^*$

$$\mathbb{E}[\nabla f^{\xi}(\mathbf{w}^{\text{old}}) - \nabla h(\mathbf{w}^{\text{old}})] = \mathbf{0}$$

★ **Unbiased estimation of** $\nabla h(\mathbf{w}^r)$

[1] Johnson, Rie, and Tong Zhang. "Accelerating stochastic gradient descent using predictive variance reduction." In *Advances in neural information processing systems*, pp. 315-323. 2013.
[2] Reddi, Sashank J., Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. "Stochastic variance reduction for nonconvex optimization." In *International conference on machine learning*, pp. 314-323. 2016.

Stochastic Variance Reduced Gradient (SVRG)

Key idea: If we have access to history point $\mathbf{w}^{\text{old}}$ and $\nabla h(\mathbf{w}^{\text{old}})$

$$\mathbf{g}^r = \nabla f^{\xi}(\mathbf{w}^r) - \nabla f^{\xi}(\mathbf{w}^{\text{old}}) + \nabla h(\mathbf{w}^{\text{old}})$$

$\rightarrow \mathbf{0}$ if $\mathbf{w}^r \approx \mathbf{w}^{\text{old}}$

$\rightarrow \mathbf{0}$ if $\mathbf{w}^{\text{old}} \approx \mathbf{w}^*$

$$\mathbb{E}[\nabla f^{\xi}(\mathbf{w}^{\text{old}}) - \nabla h(\mathbf{w}^{\text{old}})] = \mathbf{0}$$

★ **Unbiased estimation of** $\nabla h(\mathbf{w}^r)$

★ **Converges to zero if**

$$\mathbf{w}^r \approx \mathbf{w}^{\text{old}} \approx \mathbf{w}^*$$

[1] Johnson, Rie, and Tong Zhang. "Accelerating stochastic gradient descent using predictive variance reduction." In *Advances in neural information processing systems*, pp. 315-323. 2013.
[2] Reddi, Sashank J., Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. "Stochastic variance reduction for nonconvex optimization." In *International conference on machine learning*, pp. 314-323. 2016.

Stochastic Variance Reduced Gradient (SVRG)

Key idea: If we have access to history point $\mathbf{w}^{\text{old}}$ and $\nabla h(\mathbf{w}^{\text{old}})$

$$\mathbf{g}^r = \nabla f^{\xi}(\mathbf{w}^r) - \nabla f^{\xi}(\mathbf{w}^{\text{old}}) + \nabla h(\mathbf{w}^{\text{old}})$$

$\rightarrow \mathbf{0}$ if $\mathbf{w}^r \approx \mathbf{w}^{\text{old}}$

$\rightarrow \mathbf{0}$ if $\mathbf{w}^{\text{old}} \approx \mathbf{w}^*$

$$\mathbb{E}[\nabla f^{\xi}(\mathbf{w}^{\text{old}}) - \nabla h(\mathbf{w}^{\text{old}})] = \mathbf{0}$$

★ **Unbiased estimation of** $\nabla h(\mathbf{w}^r)$

★ **Converges to zero if**

$$\mathbf{w}^r \approx \mathbf{w}^{\text{old}} \approx \mathbf{w}^*$$

Variance is Reduced! → $\mathcal{O}(N^{2/3}/\epsilon)$

[1] Johnson, Rie, and Tong Zhang. "Accelerating stochastic gradient descent using predictive variance reduction." In *Advances in neural information processing systems*, pp. 315-323. 2013.

[2] Reddi, Sashank J., Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. "Stochastic variance reduction for nonconvex optimization." In *International conference on machine learning*, pp. 314-323. 2016.

Stochastic Variance Reduced Gradient (SVRG)

Key idea: If we have access to history point $\mathbf{w}^{\text{old}}$ and $\nabla h(\mathbf{w}^{\text{old}})$

$$\mathbf{g}^r = \nabla f^{\xi}(\mathbf{w}^r) - \nabla f^{\xi}(\mathbf{w}^{\text{old}}) + \nabla h(\mathbf{w}^{\text{old}})$$

$\rightarrow \mathbf{0}$ if $\mathbf{w}^r \approx \mathbf{w}^{\text{old}}$

$\rightarrow \mathbf{0}$ if $\mathbf{w}^{\text{old}} \approx \mathbf{w}^*$

$$\mathbb{E}[\nabla f^{\xi}(\mathbf{w}^{\text{old}}) - \nabla h(\mathbf{w}^{\text{old}})] = \mathbf{0}$$

Periodically calculate full gradient

★ **Unbiased estimation of** $\nabla h(\mathbf{w}^r)$

★ **Converges to zero if**

$$\mathbf{w}^r \approx \mathbf{w}^{\text{old}} \approx \mathbf{w}^*$$

Variance is Reduced! → $\mathcal{O}(N^{2/3}/\epsilon)$

[1] Johnson, Rie, and Tong Zhang. "Accelerating stochastic gradient descent using predictive variance reduction." In *Advances in neural information processing systems*, pp. 315-323. 2013.

[2] Reddi, Sashank J., Ahmed Hefny, Suvrit Sra, Barnabas Poczos, and Alex Smola. "Stochastic variance reduction for nonconvex optimization." In *International conference on machine learning*, pp. 314-323. 2016.

StochAstic Recursive grAdient algoritHm (SARAH)

SVRG: $\mathbf{g}^r = \nabla f^{\xi}(\mathbf{w}^r) - \nabla f^{\xi}(\mathbf{w}^{\text{old}}) + \nabla h(\mathbf{w}^{\text{old}})$

Key idea: recursive/adaptive update of gradient estimates

SARAH: $\mathbf{g}^r = \nabla f^{\xi}(\mathbf{w}^r) - \nabla f^{\xi}(\mathbf{w}^{r-1}) + \mathbf{g}^{r-1}$

★ **No longer unbiased**

★ **Overall unbiased**

Rate is Improved! → $\mathcal{O}(\sqrt{N}/\epsilon)$

[1] Nguyen, Lam M., Jie Liu, Katya Scheinberg, and Martin Takáč. "Stochastic recursive gradient algorithm for nonconvex optimization." *arXiv preprint arXiv:1705.07261* (2017).
[2] Fang, Cong, Chris Junchi Li, Zhouchen Lin, and Tong Zhang. "Spider: Near-optimal non-convex optimization via stochastic path-integrated differential estimator." In *Advances in Neural Information Processing Systems*, pp. 689-699. 2018.

StochAstic Recursive grAdient algoritHm (SARAH)

Algorithm 1 SARAH

Parameters: the learning rate $\alpha > 0$, the inner loop size q, and the outer loop size S

Initialize: $\tilde{\mathbf{w}}_0$

Iterate:

for $s = 1, 2, \ldots, S$ **do**

$\mathbf{w}_0^{(s)} = \tilde{\mathbf{w}}_{s-1}$

$\mathbf{v}_0^{(s)} = \frac{1}{N} \sum_{j=1}^{N} \nabla f^j(\mathbf{w}_0^{(s)})$ Periodically requires full gradient

$\mathbf{w}_1^{(s)} = \mathbf{w}_0^{(s)} - \alpha \mathbf{v}_0^{(s)}$

Iterate:

for $t = 1, \ldots, q$ **do**

Sample j_t uniformly at random from $[n]$

$\mathbf{v}_t^{(s)} = \nabla f^{j_t}(\mathbf{w}_t^{(s)}) - \nabla f^{j_t}(\mathbf{w}_{t-1}^{(s)}) + \mathbf{v}_{t-1}^{(s)}$ Estimate gradient using samples

$\mathbf{w}_{t+1}^{(s)} = \mathbf{w}_t^{(s)} - \alpha \mathbf{v}_t^{(s)}$

end for

Set $\tilde{\mathbf{w}}_s = \mathbf{w}_{m+1}^{(s)}$

end for

[1] Nguyen, Lam M., Jie Liu, Katya Scheinberg, and Martin Takáč. "Stochastic recursive gradient algorithm for nonconvex optimization." *arXiv preprint arXiv:1705.07261* (2017).

DGET: A Decentralized Stochastic Algorithm with Reduced Complexity

DGET: A Decentralized Stochastic Algorithm with Reduced Complexity

- Local update using estimated gradient:

$$\mathbf{x}_i^r = \sum_{k \in \mathcal{N}_i} \mathbf{W}_{ik} \mathbf{x}_k^{r-1} - \alpha \mathbf{y}_i^{r-1}.$$

DGET: A Decentralized Stochastic Algorithm with Reduced Complexity

- Local update using estimated gradient:

$$\mathbf{x}_i^r = \sum_{k \in \mathcal{N}_i} \mathbf{W}_{ik} \mathbf{x}_k^{r-1} - \alpha \mathbf{y}_i^{r-1}.$$

- Estimate the local gradient using a subset of local samples

$$\begin{cases} \mathbf{v}_i^r = \nabla f_i(\mathbf{x}_i^r), & \mathrm{mod}(r, q) = 0 \\ \mathbf{v}_i^r = \dfrac{1}{|S_2|} \displaystyle\sum_{j \in S_2} \left[\nabla f_i^j(\mathbf{x}_i^r) - \nabla f_i^j(\mathbf{x}_i^{r-1}) \right] + \mathbf{v}_i^{r-1}, & \mathrm{mod}(r, q) \neq 0 \end{cases}$$

DGET: A Decentralized Stochastic Algorithm with Reduced Complexity

- Local update using estimated gradient:

$$\mathbf{x}_i^r = \sum_{k \in \mathcal{N}_i} \mathbf{W}_{ik} \mathbf{x}_k^{r-1} - \alpha \mathbf{y}_i^{r-1}.$$

- Estimate the local gradient using a subset of local samples

$$\begin{cases} \mathbf{v}_i^r = \nabla f_i(\mathbf{x}_i^r), & \mathrm{mod}(r, q) = 0 \\ \mathbf{v}_i^r = \dfrac{1}{|S_2|} \displaystyle\sum_{j \in S_2} \left[\nabla f_i^j(\mathbf{x}_i^r) - \nabla f_i^j(\mathbf{x}_i^{r-1}) \right] + \mathbf{v}_i^{r-1}, & \mathrm{mod}(r, q) \neq 0 \end{cases}$$

- Track the global full gradient

$$\mathbf{y}_i^r = \sum_{k \in \mathcal{N}_i} \mathbf{W}_{ik} \mathbf{y}_k^{r-1} + \mathbf{v}_i^r - \mathbf{v}_i^{r-1}.$$

Assumptions

Assumption 1. *The objective function has Lipschitz continuous gradient with constant L:*

$$\|\nabla f_j^i(\mathbf{x}_i) - \nabla f_j^i(\mathbf{x}_i')\| \leq L\|\mathbf{x}_i - \mathbf{x}_i'\|, \forall i, j. \tag{1}$$

Assumption 2. *The mixing matrix $\mathbf{W} \in \mathbb{R}^{m\times m}$ is symmetric, and satisfying the following*

$$|\lambda_{\max}(\mathbf{W})| := \eta < 1, \quad \mathbf{W1} = \mathbf{1}, \tag{2}$$

where $\lambda_{\max}(\mathbf{W})$ denotes the second largest eigenvalue of $\mathbf{W}$.

Reduced Sample Complexity

Claim 3. Under Assumption 1 and 2, pick the inner loop iteration and sample size as $q = |S_2| = \sqrt{N}$, then to achieve ϵ stationary solution of the problem, the total number of iterations T and communication rounds required by D-GET are both in the order of $\mathcal{O}(\epsilon^{-1})$; The total number of samples is at most

$$\mathcal{O}(MN + MN^{1/2}\epsilon^{-1}).$$

Remark. This bound is $\sqrt{N}$ times better than the best existing decentralized upper bound [Hong et al., 16][Sun-Scutari 18][Sun-Hong 18]

Remark. This bound is $\sqrt{M}$ times worse than the centralized lower bound [Fang et al, 2018]

Sample Complexity Bounds

Consider a M node problem, each contains N data points

$$\min_{\mathbf{x}\in\mathbb{R}^{MP}} f(\mathbf{x}) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N} f_i^j(\mathbf{x}_i), \quad \text{s.t.} \quad \mathbf{x}_i = \mathbf{x}_k, \quad \forall (i,k) \in \mathcal{E}$$

	Sample Complexity	Communication Complexity
Lower Bound	$\mathcal{O}(\sqrt{MN}/\epsilon)$ **[Fang et al, 18] Centralized method**	$\mathcal{O}(1/\epsilon)$ **[Sun-H., 18]**
Upper Bound (deterministic)	$\mathcal{O}(M\sqrt{N}/\epsilon)$ **[Sun-Lu-H., 18]**	$\mathcal{O}(1/\epsilon)$ **[Sun-Lu-H., 18]**
Upper Bound (stochastic)	$\mathcal{O}(M/\epsilon^2)$ **[Tang et al 18][Lu et al 18]**	$\mathcal{O}(1/\epsilon^2)$ **[Tang et al 18][Lu et al 18]**

Open Questions: Streaming Data Model

- Q1: Possible to achieve the “centralized” sample complexity lower bound? Or has to do worse due to distributed data?
- Q2: Optimal graph dependency and sample complexity?

Open Questions: Different Problem Classes?

- Up to now the set of problems see is very limited
- Smooth problem
- Each component function has Lipschitz gradient
- Slightly generalize the problem setting?
- Unexplored territory!

A Simple “Difficult” Example

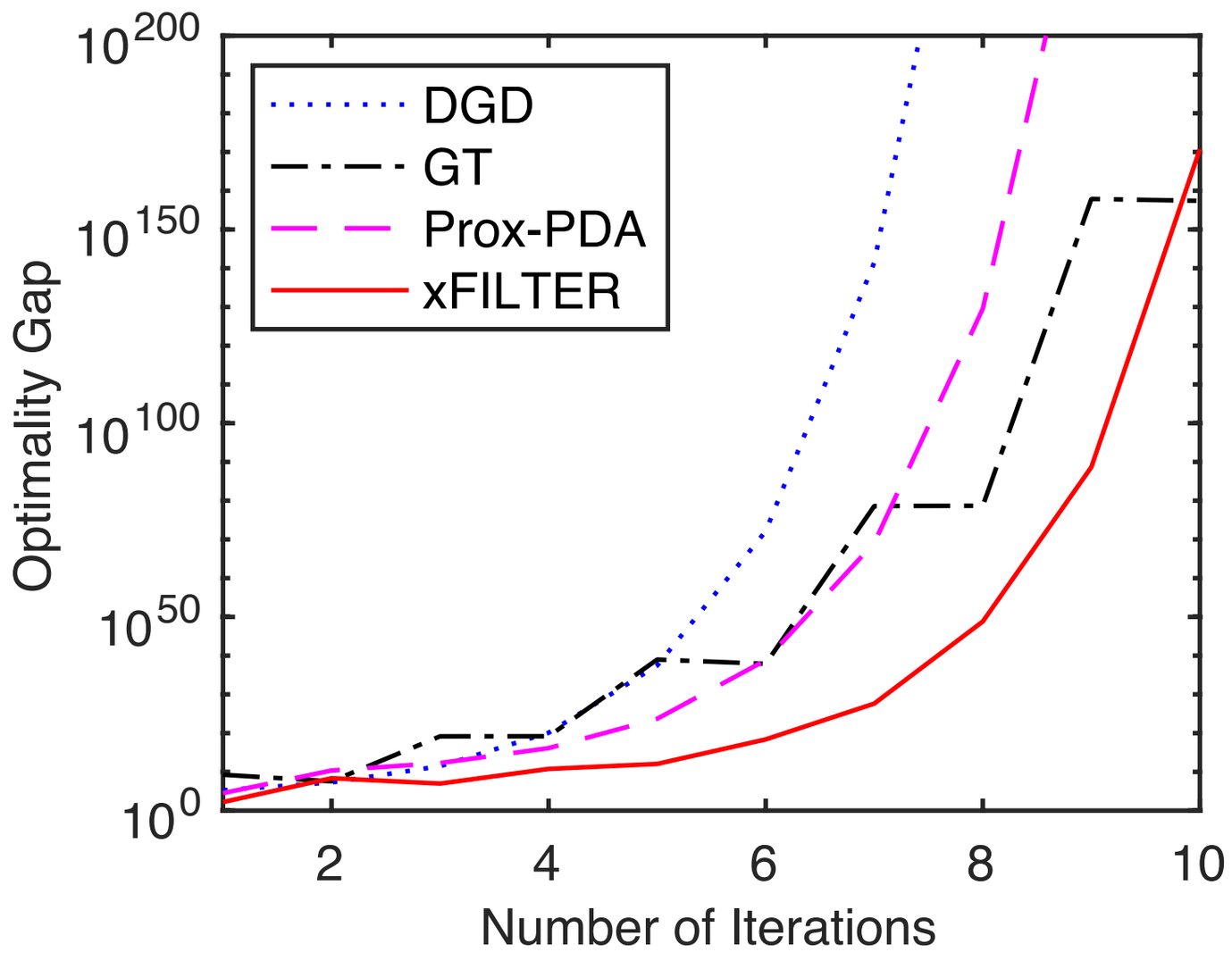
- Consider the problem: $f(\mathbf{x}) := x_1^3 - x_2^3$

- A simple problem that makes almost all existing distributed methods diverge!

A Simple “Difficult” Example

- Consider the problem: $f(\mathbf{x}) := x_1^3 - x_2^3$

- A simple problem that makes almost all existing distributed methods diverge!



Open Questions: Different Problem Classes

- It is not clear if we can extend all these methods to non-smooth problems

- That is, design algorithm for the following problem?

$$f(\mathbf{x}) := \sum_{i=1}^{N} f_i(x_i) + \underbrace{h_i(x_i)}_{\text{node-specific non-smooth terms}}$$

- Node-specific constrains (e.g., manifold constrains)

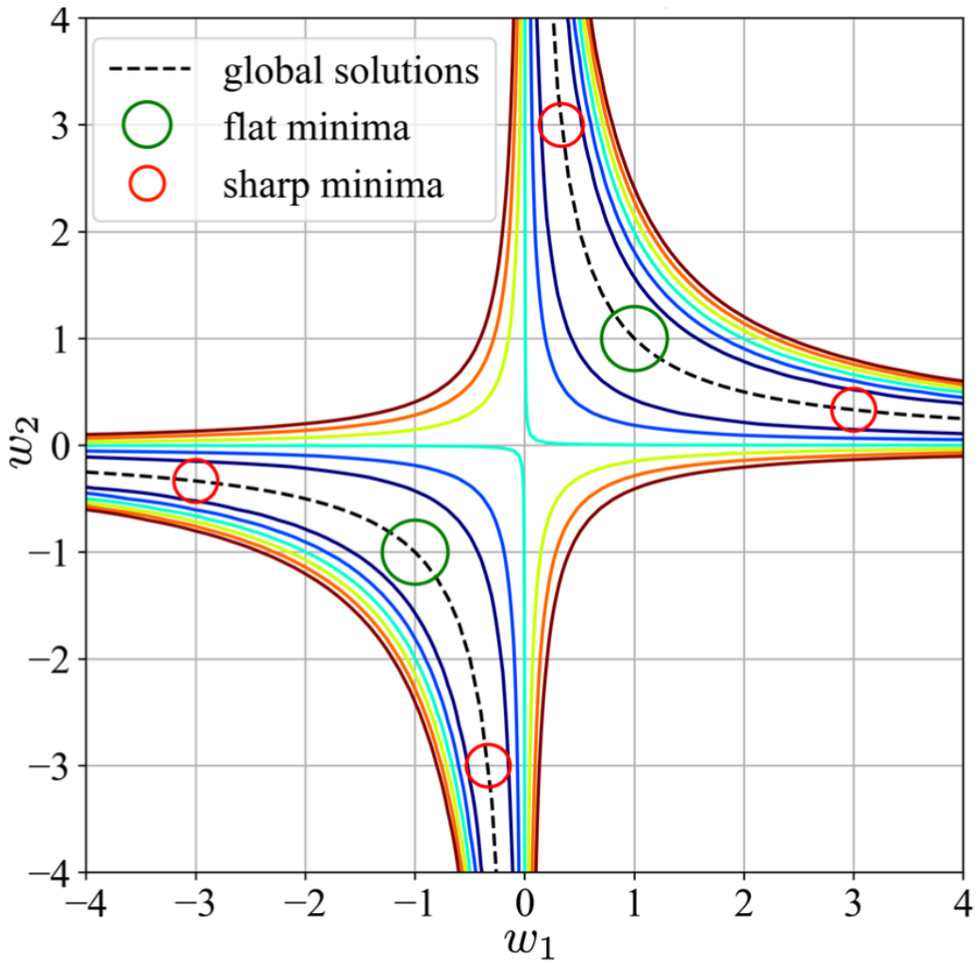
- Homogeneous data (lower bounds v.s. upper bounds?)

Open Questions: Beyond First-Order Solutions

- Up to now we only talked about first-order solutions
- High-Order solutions have better “quality”
  - Recent works on achieving second-order stationary solutions [H.-Lee-Razaviyan 18] [Daneshmand et al 18]
  - Or some approximation of them [Swenson et al 19a]
- Globally optimal solutions by annealing [Swenson et al 19a]
- Globally “optimal” rates? Problems with special structures?

Beyond First-Order Methods

- Up to now we only talked about first-order algorithms
- Higher-order methods possess several nice features



[Ginsburg et al., 19]

Experiments with deep linear networks

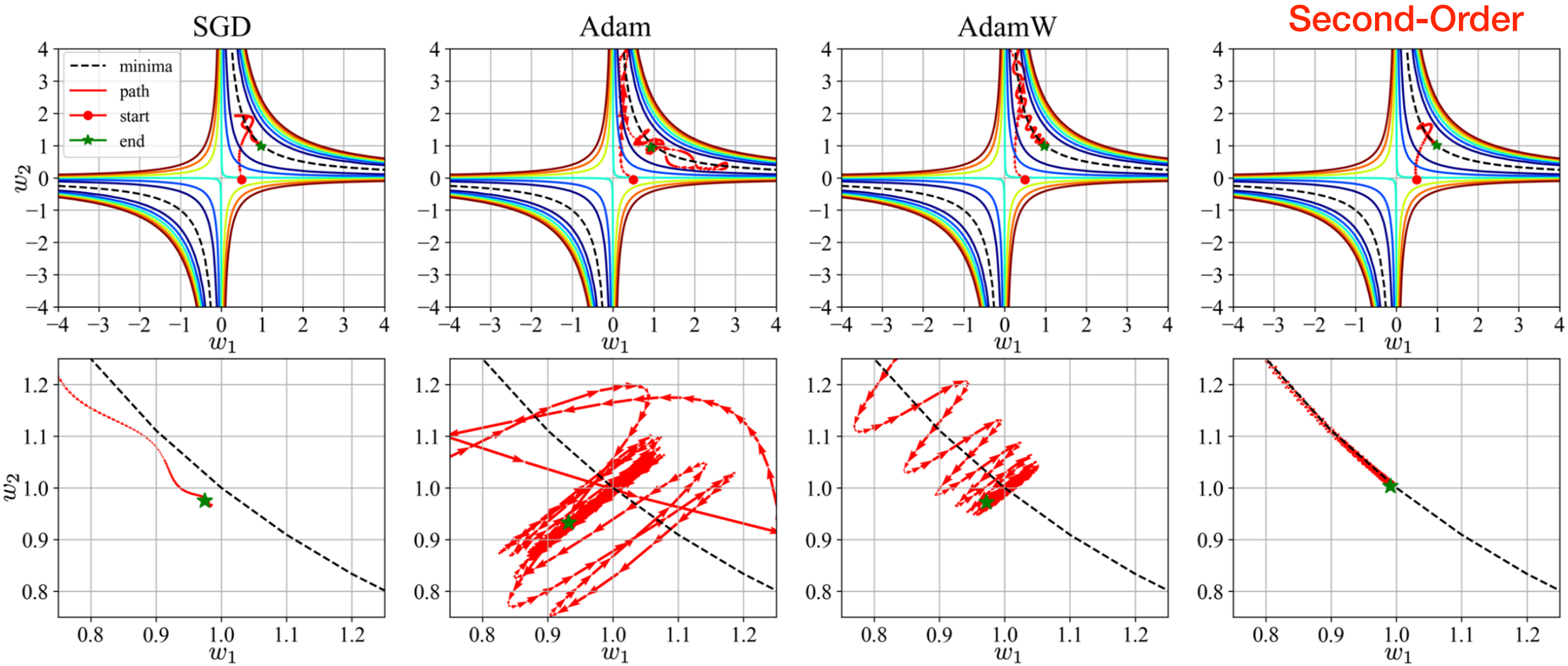
$$Y = W_1 \times W_2 \times X$$

flat minima: good generalization

sharp minima: bad generalization

Beyond First Order Methods

- Up to now we only talked about first-order problems
- Higher-order methods possess several nice features



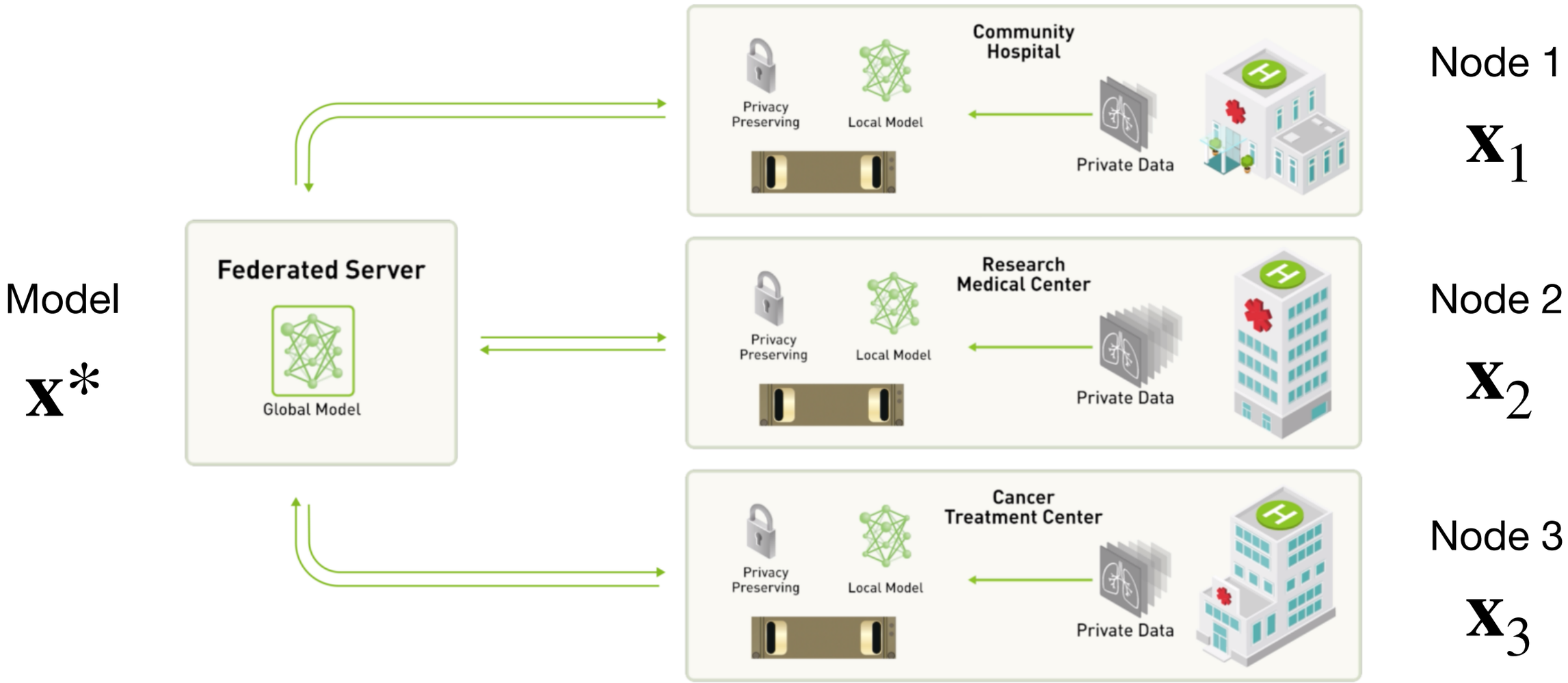
[Ginsburg et al., 19]

Open Questions: Beyond First-Order Algorithms

- Up to now we only talked about first-order algorithms
- Higher-order methods possess several nice features
  - mitigate the effects of ill-conditioning
  - avoid or diminish the need for hyper-parameter tuning
  - take advantage of distributed computing environments
- Not clear how to utilize high-order information in distributed methods (for non-convex problems)

Open Questions:
Beyond Consensus Models

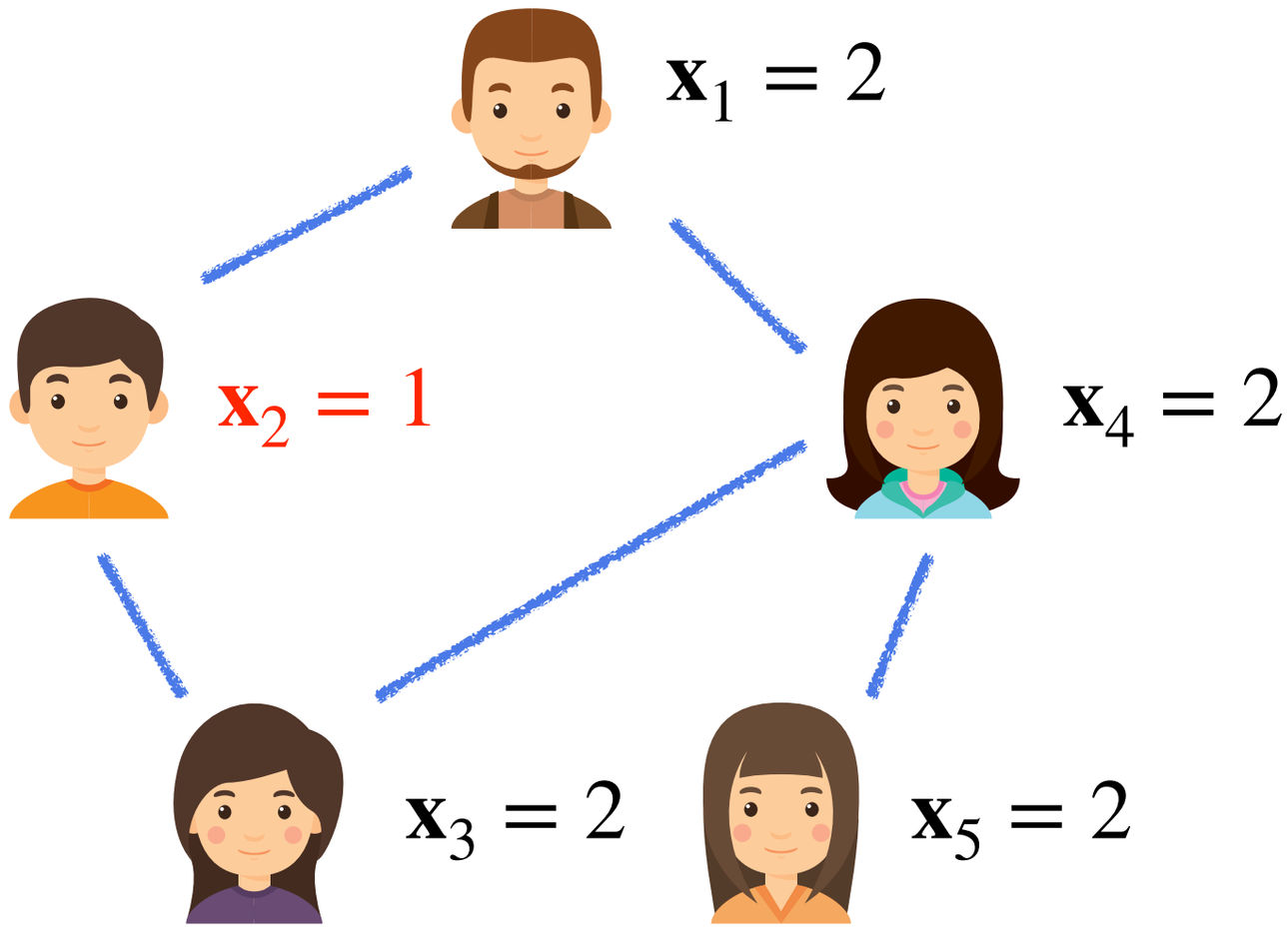
- Federated Learning: Local agents prefer local specific models



$$D(\mathbf{x}^* - \mathbf{x}_1) \leq \epsilon, D(\mathbf{x}^* - \mathbf{x}_2) \leq \epsilon, D(\mathbf{x}^* - \mathbf{x}_3) \leq \epsilon$$

Do we always need “exact consensus”?

- Federated Learning: Local agents prefer local specific models
- Adversarial Attacks: Exact consensus will be vulnerable to attack



1+1=?

$$D(\mathbf{x}_1 - \mathbf{x}_2) \leq \epsilon$$

Key References

- T.-H. Chang, M. Hong, H.-T. Wai, X. Zhang and S. Lu, “Distributed Learning in the Non-Convex World: From Batch to Streaming Data, and Beyond”, 2019

- H. Sun and M. Hong, “Distributed non-convex first-order optimization and information processing: Lower complexity bounds and rate optimal algorithms”, *IEEE TSP, 2019*

- M. Hong, J. Lee, M. Razaviyayn,“Gradient primal-dual algorithm converges to second-order stationary solution for nonconvex distributed optimization over networks,” in ICML, 2018

Thank you!

Proof sketch (2): Detailed Function

Functions to optimize:

$$h_i(\mathbf{x}_i) = \begin{cases} \Theta(\mathbf{x}_i, 1) + 3 \sum_{j=1}^{\lfloor T/2 \rfloor} \underbrace{\Theta(\mathbf{x}_i, 2j)}_{\text{even function}}, & i \in [1, \frac{M}{3}] \\ \Theta(\mathbf{x}_i, 1), & i \in [\frac{M}{3}, \frac{2M}{3}] \\ \Theta(\mathbf{x}_i, 1) + 3 \sum_{j=1}^{\lfloor T/2 \rfloor} \underbrace{\Theta(\mathbf{x}_i, 2j+1)}_{\text{odd function}}, & i \in [\frac{2M}{3}, M] \end{cases}$$

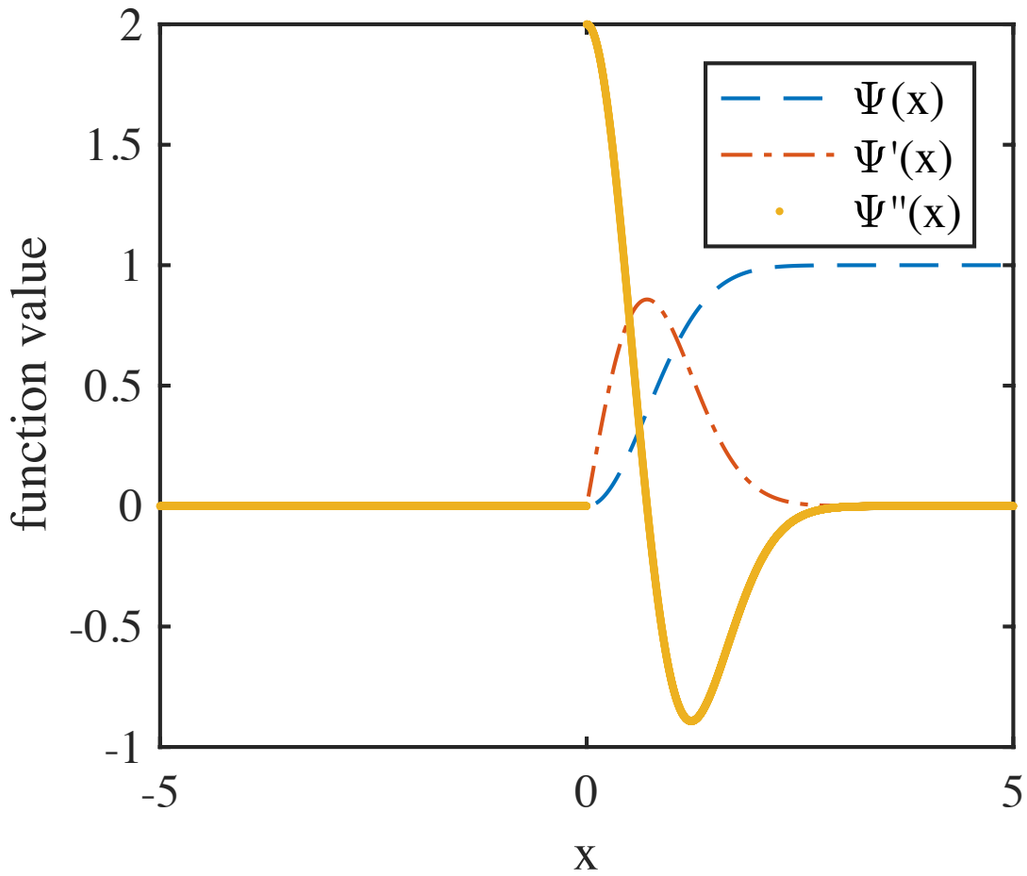
where each component function is related to two coordinates:

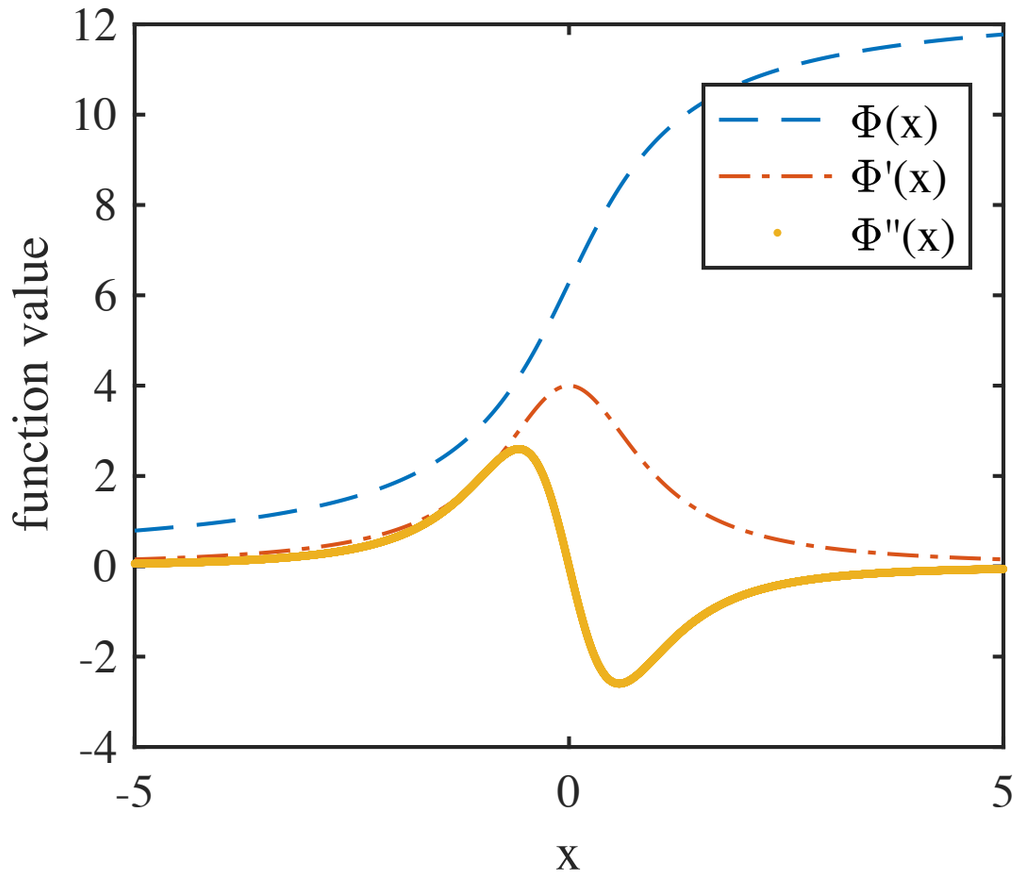
$$\Theta(\mathbf{x}_i, j) = \Psi(-\mathbf{x}_i[j-1])\Phi(-\mathbf{x}_i[j]) - \Psi(\mathbf{x}_i[j-1])\Phi(\mathbf{x}_i[j]), \forall j \geq 2$$
$$\Theta(\mathbf{x}_i, 1) = -\Psi(1)\Phi(-\mathbf{x}_i[1]).$$

The scalar functions $\Psi, \Phi : \mathbb{R} \to \mathbb{R}$ as below

$$\Psi(w) := \begin{cases} 0 & w \leq 0 \\ 1 - e^{-w^2} & w > 0, \end{cases} \quad \text{and} \quad \Phi(w) = 4 \arctan w + 2\pi.$$

Proof sketch (2): Detailed Function





$$\frac{\partial \Theta(x_i, j)}{\partial x_i[j]} = -\Psi(-x_i[j-1])\Phi'(-x_i[j]) - \Psi(x_i[j-1])\Phi'(x_i[j])$$

$$\Psi(w)\Phi'(v) > 1, \quad \forall\, w \geq 1, |v| < 1$$

Comments: On Filtering Step

- Local average approximation: $\mathbf{d}_i = \frac{1}{M}\sum_{i=1}^{M} \mathbf{x}_i + \mathbf{e}_i$
- Gradually remove such kind of estimation noise

$$\mathbf{x}_*^{r+1} := \arg\min_{\mathbf{x}\in\mathbb{R}^M} \underbrace{\|\mathbf{x}-\mathbf{d}^r\|^2}_{\mathbf{x}\text{ track }\mathbf{d}} + \mu \underbrace{\mathbf{x}^T\mathcal{L}\mathbf{x}}_{\text{smoothing penalty}}$$

$$= \underbrace{(I_M + \mu\mathcal{L})^{-1}}_{\text{LPF}} \mathbf{d}^r$$

Approximate LPF by Chebyshev Filter