The BRIDGE Framework: Robustness and Resilience in Decentralized Learning

Waheed U. Bajwa

Department of Electrical and Computer Engineering
Rutgers University–New Brunswick, NJ USA
www.inspirelab.us

Bellairs Workshop
January 23, 2024

CCF-1907658
CNS-2148104

W911NF-21-1-0301

Collaborators and References

Collaborators

- **Zhixiong Yang (former PhD student)**
- **Cheng Fang (PhD student)**
- Rishabh Dixit (PhD student)
- Mert Gurbuzbalaban (Collaborator)
- Or Shalom (Collaborator)
- Amir Leshem (Collaborator)

Code

- J. Shenouda, Z. Yang, and W. U. Bajwa, "Codebase—Adversary-resilient distributed and decentralized statistical inference and machine learning: An overview of recent advances under the Byzantine threat model," Jul. 2020.

Papers

1. Z. Yang and W. U. Bajwa, "RD-SVM: A resilient distributed support vector machine," ICASSP'16.
2. Z. Yang and W. U. Bajwa, "ByRDiE: A Byzantine-resilient distributed learning algorithm," DSW'18.
3. Z. Yang and W. U. Bajwa, "ByRDiE: Byzantine-resilient distributed coordinate descent . . . ," IEEE TSIPN, 2019.
4. Z. Yang and W. U. Bajwa, "PAC learning from distributed data in the presence of malicious nodes," CAMSAP'19.
5. Z. Yang, A. Gang, and W. U. Bajwa, "Adversary-resilient distributed and decentralized statistical inference and machine learning . . . ," IEEE SPM, 2020.
6. **C. Fang, Z. Yang, and W. U. Bajwa, "BRIDGE: Byzantine-resilient decentralized gradient descent," IEEE TSIPN, 2022.**
7. O. Shalom, A. Leshem, and W. U. Bajwa, "Mitigating data injection attacks on federated learning," ICASSP'24.

Outline

1. Motivation
2. Problem Formulation
3. The BRIDGE Framework
4. Theoretical Analysis
5. Numerical Results

Outline

1. Motivation
2. Problem Formulation
3. The BRIDGE Framework
4. Theoretical Analysis
5. Numerical Results

Machine Learning in Modern Applications

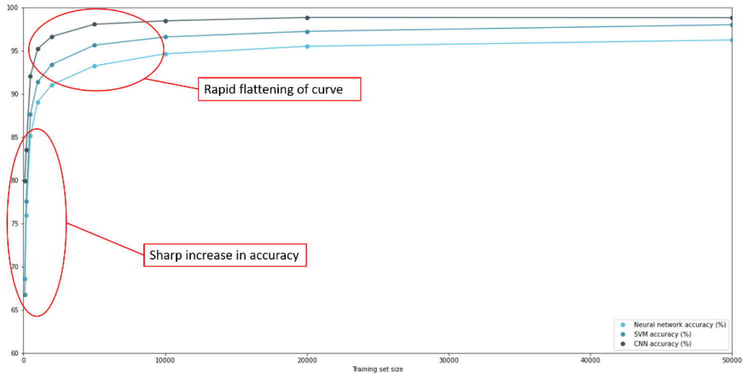
Machine learning (ML) is rapidly becoming a cornerstone in both current and futuristic applications across various industries. It underpins the development of advanced systems and technologies.

Fundamental Concept: At its core, ML involves using optimization techniques to learn model parameters from data, enabling the creation of predictive models that adapt and improve over time.

Image credit(s): Google; Finance Brokerage; dormakaba

Sample Size and Generalization Error in Machine Learning

Key Concept: For independent and identically distributed (i.i.d.) training samples, the generalization error in machine learning typically scales as $\mathcal{O}(1/\sqrt{N})$ with sample size N.



Implication: Larger datasets enhance model accuracy and generalization.

Image credit(s): Telstra Purple

The Importance of Diverse Data in Machine Learning

- In many applications, datasets often come from limited regions of the underlying distribution, leading to non-diverse datasets.
- Diverse datasets are crucial for exploring the entire distribution space, often requiring the integration of data from various sources.

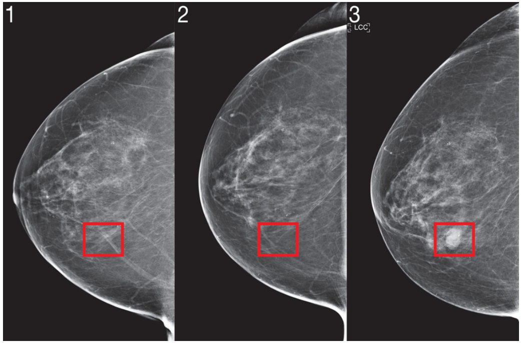


Image credit(s): MIT News; May 7, 2019

Data Integration Strategies in Machine Learning

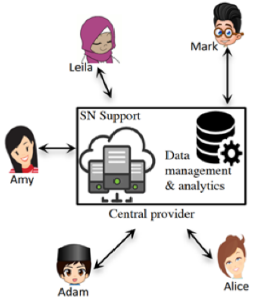
Machine Learning data integration can utilize distributed systems (with central server) or decentralized systems (without central server), influenced by the problem structure and privacy concerns.

Applications Necessitating Distributed / Decentralized Systems

- Multi-agent systems
- Internet-of-Things systems
- Smart grids
- Sensor networks

Privacy-Sensitive Data Sources

- Smartphone data
- Social network data
- Healthcare data



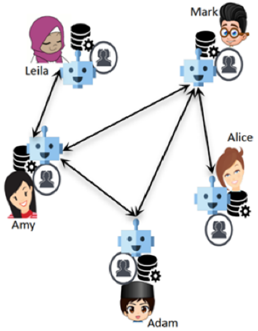
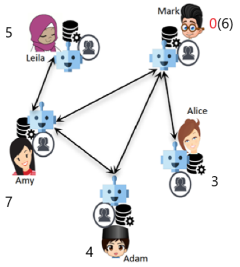


Image credit(s): Merehead

Handling Failures in Decentralized Machine Learning Algorithms

Potential Security Risks: *What happens if a device in the network, like Mark's phone, is compromised?*

Impact of Malicious Nodes: [Su and Vaidya, 2016] have demonstrated that even a single malicious node, employing basic disruptive strategies, can lead to the failure of a decentralized consensus or learning algorithm. This highlights the vulnerability of decentralized systems to targeted attacks.

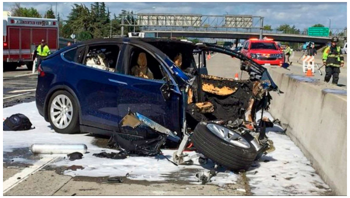


Key Consideration: Robust security measures and fault tolerance mechanisms are essential in decentralized ML to mitigate the risk of system failure due to compromised nodes.

Image credit(s): Merehead

Consequences of Algorithmic Failure in Real Life

Critical Importance of Accurate Classification: *A misclassification of street signs by automated systems can have dire consequences, such as severe car accidents.*



This emphasizes the need for robust and reliable algorithms in critical applications, where errors can have life-threatening consequences.

Image credit(s): Google; ABC News

Addressing Security in Decentralized ML Systems

In light of the critical importance of algorithmic reliability, our focus shifts to addressing security threats in decentralized machine learning systems.

Understanding Byzantine Failures

In a decentralized environment, Byzantine failures (Lamport, Shostak, and Pease, 1982) refer to nodes that deviate arbitrarily from the agreed-upon protocol, potentially compromising the entire network.

Addressing Security in Decentralized ML Systems

In light of the critical importance of algorithmic reliability, our focus shifts to addressing security threats in decentralized machine learning systems.

Understanding Byzantine Failures

In a decentralized environment, Byzantine failures (Lamport, Shostak, and Pease, 1982) refer to nodes that deviate arbitrarily from the agreed-upon protocol, potentially compromising the entire network.

Aim of Byzantine-Resilient Decentralized Learning Algorithms

- Efficiently using data in a decentralized manner
- Ensuring robustness against Byzantine failures within the network

Goal: To develop decentralized ML algorithms that are not only efficient in data utilization but also resilient to the unpredictable nature of Byzantine failures.

Outline

1. Motivation
2. Problem Formulation
3. The BRIDGE Framework
4. Theoretical Analysis
5. Numerical Results

The Empirical Risk Minimization (ERM) Framework of Learning

The loss function $(\mathbf{w}, \mathbf{z}) \mapsto f(\mathbf{w}, \mathbf{z})$, with $\mathbf{w} \in \mathbb{R}^d$ and $\mathbf{z} \sim (\Omega, \mathcal{F}, \mathbb{P})$, defines our optimization objective in ML—find $\mathbf{w}^*$ that minimizes the expected loss (statistical risk):

$$\mathbf{w}^* \in \arg\min_{\mathbf{w} \in \mathbb{R}^d} \mathbb{E}_{\mathbb{P}}[f(\mathbf{w}, \mathbf{z})].$$

Empirical Risk Minimization (ERM)

Use data samples $\mathcal{Z} := \{\mathbf{z}_n\}_{n=1}^N$ to approximate the (statistical) risk and solve:

$$\mathbf{w}^*_{\text{ERM}} \in \arg\min_{\mathbf{w} \in \mathbb{R}^d} \frac{1}{N} \sum_{n=1}^{N} f(\mathbf{w}, \mathbf{z}_n).$$

Objectives

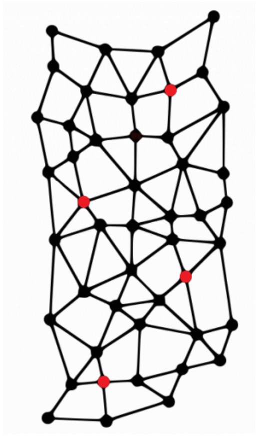
- Fast algorithmic convergence $(\mathbf{w}(t, N) \to \mathbf{w}^*_{\text{ERM}})$ to a stationary point
- Fast statistical convergence $(\mathbf{w}(t, N) \to \mathbf{w}^*)$ to the Bayes optimal solution (when possible)

The Decentralized ERM Framework of Learning

Consider a network of M nodes and a connected graph $\mathcal{G}(\mathcal{J}, \mathcal{E})$

- Nodes receive messages only from connected peers
- Local i.i.d. training dataset $\mathcal{Z}_j := \{\mathbf{z}_{jn}\}_{n=1}^{|\mathcal{Z}_j|}$ at each node
- Local loss function $f_j(\mathbf{w}) := \frac{1}{|\mathcal{Z}_j|}\sum_{n=1}^{|\mathcal{Z}_j|} f(\mathbf{w}, \mathbf{z}_{jn})$ for each node
- **Original Goal:** Collaboratively solve the decentralized ERM problem

$$\min_{\{\mathbf{w}_1,\dots,\mathbf{w}_M\}} \frac{1}{M}\sum_{j=1}^{M} f_j(\mathbf{w}_j) \quad \text{s.t.} \quad \forall i,j,\ \mathbf{w}_i = \mathbf{w}_j$$



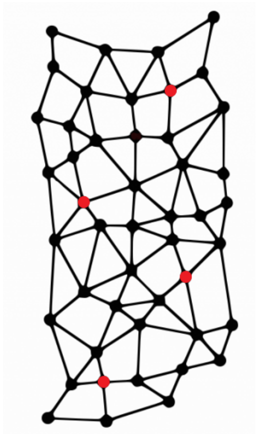
The Decentralized ERM Framework of Learning

Consider a network of M nodes and a connected graph $\mathcal{G}(\mathcal{J}, \mathcal{E})$

- Nodes receive messages only from connected peers
- Local i.i.d. training dataset $\mathcal{Z}_j := \{\mathbf{z}_{jn}\}_{n=1}^{|\mathcal{Z}_j|}$ at each node
- Local loss function $f_j(\mathbf{w}) := \frac{1}{|\mathcal{Z}_j|}\sum_{n=1}^{|\mathcal{Z}_j|} f(\mathbf{w}, \mathbf{z}_{jn})$ for each node
- **Original Goal:** Collaboratively solve the decentralized ERM problem

$$\min_{\{\mathbf{w}_1,\dots,\mathbf{w}_M\}} \frac{1}{M}\sum_{j=1}^{M} f_j(\mathbf{w}_j) \quad \text{s.t.} \quad \forall i,j,\ \mathbf{w}_i = \mathbf{w}_j$$

- Potential Byzantine failures at some nodes



The Decentralized ERM Framework of Learning

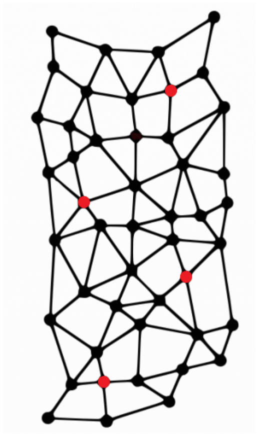
Consider a network of M nodes and a connected graph $\mathcal{G}(\mathcal{J}, \mathcal{E})$

- Nodes receive messages only from connected peers
- Local i.i.d. training dataset $\mathcal{Z}_j := \{\mathbf{z}_{jn}\}_{n=1}^{|\mathcal{Z}_j|}$ at each node
- Local loss function $f_j(\mathbf{w}) := \frac{1}{|\mathcal{Z}_j|} \sum_{n=1}^{|\mathcal{Z}_j|} f(\mathbf{w}, \mathbf{z}_{jn})$ for each node
- **Original Goal:** Collaboratively solve the decentralized ERM problem

$$\min_{\{\mathbf{w}_1, \ldots, \mathbf{w}_M\}} \frac{1}{M} \sum_{j=1}^{M} f_j(\mathbf{w}_j) \quad \text{s.t.} \quad \forall i, j, \ \mathbf{w}_i = \mathbf{w}_j$$

- Potential Byzantine failures at some nodes
- **Revised Goal:** Collaboratively solve the decentralized ERM problem at non-compromised nodes

$$\min_{\{\mathbf{w}_1, \ldots, \mathbf{w}_M\}} \frac{1}{r} \sum_{j \in \mathcal{R}} f_j(\mathbf{w}_j) \quad \text{s.t.} \quad \forall i, j \in \mathcal{R}, \ \mathbf{w}_i = \mathbf{w}_j$$



Relationship to Prior Works

- Distributed / federated learning: [M. Li et al., 2014; Konečný et al., 2016]
- Decentralized Gradient Descent (DGD): [Nedić and Ozdaglar, 2009; Ram, Nedić, and Veeravalli, 2010; Nedić and Olshevsky, 2015; Pu and Nedić, 2021]
- Byzantine-resilient averaging consensus: [LeBlanc et al., 2013; Vaidya, Tseng, and Liang, 2014]
- Scalar-valued models: [Su and Vaidya, 2016; Sundaram and Gharesifard, 2019]

Algorithm	Nonconvex	Byzantine failures	Algorithmic convergence rate	Statistical convergence rate
ByRDiE[1]	×	√	√	√
Kuwaranancharoen et. al[2]	×	√	×	×
Peng and Ling[3]	×	√	√	×
MOZI[4]	×	√	√	×
ICwTM[5]	√	√	√	×
DGD[6]	×	×	√	×
NEXT[7]	√	×	×	×
Nonconvex DGD[8]	√	×	√	×
D-GET[9]	√	×	√	√
GT-SARAH[10]	√	×	√	√
BRIDGE (This Talk)	√	√	√	√

Outline

1 Motivation

2 Problem Formulation

3 The BRIDGE Framework

4 Theoretical Analysis

5 Numerical Results

The BRIDGE Framework in Decentralized ML

The DGD method [Nedić and Ozdaglar, 2009] updates the local variable $\mathbf{w}_j(t)$ as

$$\mathbf{w}_j(t+1) = \sum_{i \in \mathcal{N}_j \cup \{j\}} a_{ji}\mathbf{w}_i(t) - \rho(t)\nabla f_j(\mathbf{w}_j(t)).$$

The BRIDGE Framework in Decentralized ML

The DGD method [Nedić and Ozdaglar, 2009] updates the local variable $\mathbf{w}_j(t)$ as

$$\mathbf{w}_j(t+1) = \sum_{i \in \mathcal{N}_j \cup \{j\}} a_{ji}\mathbf{w}_i(t) - \rho(t)\nabla f_j(\mathbf{w}_j(t)).$$

Overview of the BRIDGE Framework

- Requires local datasets $\mathcal{Z}_j$, maximum Byzantine nodes b, step size sequence $\{\rho(t)\}$, and max iterations $t_{\max}$.
- **Initialization:** $t \leftarrow 0$ and $\mathbf{w}_j(0)$ for all non-faulty nodes
- Iterative Process:
  - Regular nodes broadcast $\mathbf{w}_j(t)$ to neighbors
  - Regular nodes receive $\mathbf{w}_i(t)$ from neighbors
  - Screening of incoming values: $\mathbf{y}_j(t) \leftarrow \text{screen}(\{\mathbf{w}_i(t)\}_{i \in \mathcal{N}_j})$
  - Update step: $\mathbf{w}_j(t+1) \leftarrow \mathbf{y}_j(t) - \rho(t)\nabla f_j(\mathbf{w}_j(t))$

Different Variants of the BRIDGE Framework

The BRIDGE-T variant employs the coordinate-wise trimmed mean for screening [D. Yin et al., 2018]. For each iteration t and coordinate $k \in \{1, \ldots, d\}$, it computes:

$$\overline{\mathcal{N}}_j^k(t) := \underset{\mathcal{X}:\mathcal{X}\subset\mathcal{N}_j, |\mathcal{X}|=b}{\arg\min} \sum_{i\in\mathcal{X}} [\mathbf{w}_i(t)]_k,$$

$$\underline{\mathcal{N}}_j^k(t) := \underset{\mathcal{X}:\mathcal{X}\subset\mathcal{N}_j, |\mathcal{X}|=b}{\arg\max} \sum_{i\in\mathcal{X}} [\mathbf{w}_i(t)]_k,$$

$$\mathcal{C}_j^k(t) := \mathcal{N}_j \setminus \left\{ \overline{\mathcal{N}}_j^k(t) \bigcup \underline{\mathcal{N}}_j^k(t) \right\}.$$

Filtered value computation:

$$[\mathbf{y}_j(t)]_k = \frac{1}{|\mathcal{N}_j| - 2b + 1} \sum_{i\in\mathcal{C}_j^k(t)\cup\{j\}} [\mathbf{w}_i(t)]_k.$$

Different Variants of the BRIDGE Framework

The BRIDGE-T variant employs the coordinate-wise trimmed mean for screening [D. Yin et al., 2018]. For each iteration t and coordinate $k \in \{1, \ldots, d\}$, it computes:

$$\overline{\mathcal{N}}_j^k(t) := \arg\min_{\mathcal{X}:\mathcal{X}\subset\mathcal{N}_j, |\mathcal{X}|=b} \sum_{i\in\mathcal{X}} [\mathbf{w}_i(t)]_k,$$

$$\underline{\mathcal{N}}_j^k(t) := \arg\max_{\mathcal{X}:\mathcal{X}\subset\mathcal{N}_j, |\mathcal{X}|=b} \sum_{i\in\mathcal{X}} [\mathbf{w}_i(t)]_k,$$

$$\mathcal{C}_j^k(t) := \mathcal{N}_j \setminus \left\{\overline{\mathcal{N}}_j^k(t) \bigcup \underline{\mathcal{N}}_j^k(t)\right\}.$$

Filtered value computation:

$$[\mathbf{y}_j(t)]_k = \frac{1}{|\mathcal{N}_j| - 2b + 1} \sum_{i\in\mathcal{C}_j^k(t)\cup\{j\}} [\mathbf{w}_i(t)]_k.$$

The BRIDGE-M variant uses the coordinate-wise median as the filtered value:

$$[\mathbf{y}_j(t)]_k = \text{median}\big(\{[\mathbf{w}_i(t)]_k\}_{i\in\mathcal{N}_j\cup\{j\}}\big).$$

Different Variants of the BRIDGE Framework

The BRIDGE-K variant employs the so-called *Krum function* for screening [Blanchard, Guerraoui, and Stainer, 2017]. It identifies $i_j^*(t)$, the node with the minimum sum of Euclidean distances to its closest neighbors, excluding $(b+2)$ extreme values:

$$i_j^*(t) = \arg\min_{i \in \mathcal{N}_j} \sum_{\substack{h \in \mathcal{N}_j \cup \{j\} \\ h \sim i}} \|\mathbf{w}_h(t) - \mathbf{w}_i(t)\|, \qquad \text{and} \qquad \mathbf{y}_j(t) = \mathbf{w}_{i_j^*}(t).$$

- **Notation:** $h \sim i$ if $\mathbf{w}_h(t)$ is one of the $|\mathcal{N}_j| - b - 2$ vectors with the smallest Euclidean distance from $\mathbf{w}_i(t)$.
- The iterate value of neighbor index $i_j^*(t)$ can be thought of as the central point, determined after removing the most extreme values based on Euclidean distance, in the neighborhood cluster.

Different Variants of the BRIDGE Framework

The BRIDGE-K variant employs the so-called *Krum function* for screening [Blanchard, Guerraoui, and Stainer, 2017]. It identifies $i_j^*(t)$, the node with the minimum sum of Euclidean distances to its closest neighbors, excluding $(b+2)$ extreme values:

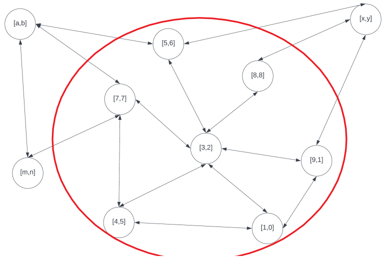
$$i_j^*(t) = \arg\min_{i \in \mathcal{N}_j} \sum_{\substack{h \in \mathcal{N}_j \cup \{j\} \\ h \sim i}} \|\mathbf{w}_h(t) - \mathbf{w}_i(t)\|, \qquad \text{and} \qquad \mathbf{y}_j(t) = \mathbf{w}_{i_j^*}(t).$$

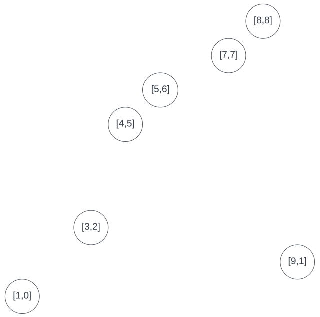
- **Notation:** $h \sim i$ if $\mathbf{w}_h(t)$ is one of the $|\mathcal{N}_j| - b - 2$ vectors with the smallest Euclidean distance from $\mathbf{w}_i(t)$.
- The iterate value of neighbor index $i_j^*(t)$ can be thought of as the central point, determined after removing the most extreme values based on Euclidean distance, in the neighborhood cluster.

The BRIDGE-B variant uses the so-called *Bulyan function* [Mhamdi, Guerraoui, and Rouault, 2018], which combines the Krum and coordinate-wise trimmed mean screening rules:

- Select $\theta = |\mathcal{N}_j| - (2b+1)$ "Krum" vectors within the neighborhood.
- Apply trimmed-mean screening rule to the selected vectors.

Illustrative Example: Comparing Screening Methods



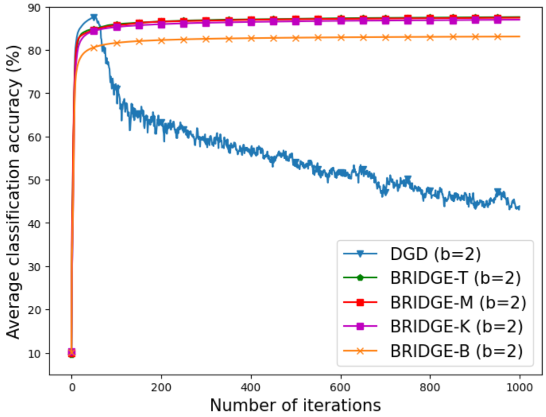


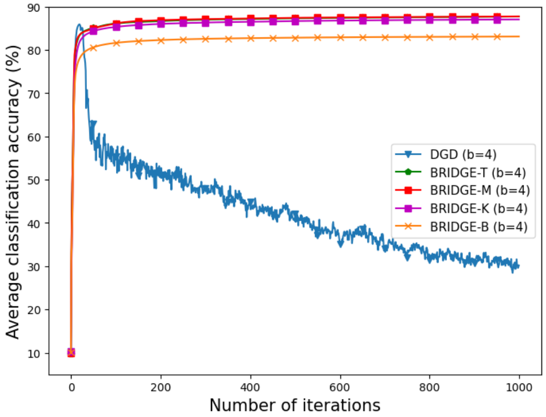
Output Comparison (with $b = 1$ Byzantine node)

- **BRIDGE-T:** Average of middle values $= (5.4, 4.2)$
- **BRIDGE-M:** Median of values $= (5, 5)$
- **BRIDGE-K:** Krum selected values $= (5, 6)$
- **BRIDGE-B:** Trimmed mean of selected values $= (4.5, 5.5)$

Numerical Results: Screening Methods on MNIST Dataset

- Experiment conducted on the MNIST dataset
- Network of 50 nodes following an Erdos-Renyi model with $p = 0.5$
- Linear classifier with squared hinge loss (convex), under i.i.d. data distribution





Comparative Summary of Different BRIDGE Variants

Variant	Screening Method	Min. Neighborhood Size	Avg. Computational Complexity
BRIDGE-T	Coordinate-wise	$2b+1$	$\mathcal{O}(nd)$, $n := \max_j \lvert\mathcal{N}_j\rvert$
BRIDGE-M	Coordinate-wise	1	$\mathcal{O}(nd)$
BRIDGE-K	Vector-based	$b+3$	$\mathcal{O}(n^2d)$
BRIDGE-B	Vector + Coordinate-wise	$\max(4b, 3b+2)+1$	$\mathcal{O}(n^2d)$

Outline

1 Motivation

2 Problem Formulation

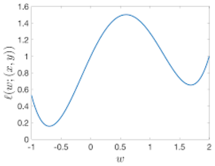
3 The BRIDGE Framework

4 Theoretical Analysis

5 Numerical Results

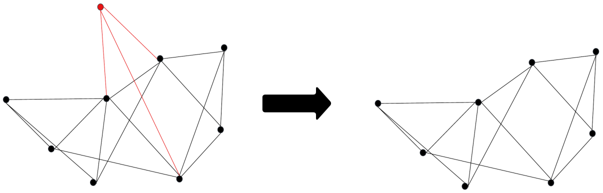
Assumptions: Loss Function Characteristics

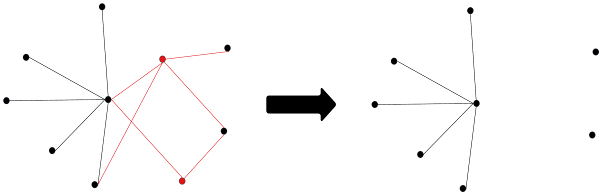
- Bounded and Lipschitz Gradients: $\forall \mathbf{w} \in \mathbb{R}^d$, $\|\nabla f(\mathbf{w}, \mathbf{z})\| \leq L$ a.s., and $\forall \mathbf{w}_1, \mathbf{w}_2 \in \mathbb{R}^d$, $\|\nabla f(\mathbf{w}_1, \mathbf{z}) - \nabla f(\mathbf{w}_2, \mathbf{z})\| \leq L' \|\mathbf{w}_1 - \mathbf{w}_2\|$ a.s.
- Bounded Training Loss: There exists a constant C such that $\sup_{\mathbf{w} \in \mathbb{R}^d, \mathbf{z} \in \mathcal{Z}} f(\mathbf{w}, \mathbf{z}) \leq C$ a.s.
- Strong Convexity (Convex): $f(\mathbf{w}_1, \mathbf{z}) \geq f(\mathbf{w}_2, \mathbf{z}) + \langle \nabla f(\mathbf{w}_2, \mathbf{z}), \mathbf{w}_1 - \mathbf{w}_2 \rangle + \frac{\lambda}{2} \|\mathbf{w}_1 - \mathbf{w}_2\|^2$ a.s.
- Local Strong Convexity (Nonconvex): $f(\mathbf{w}, \mathbf{z})$ is nonconvex, a.s. twice differentiable, and there exist positive constants λ and β such that $\forall \mathbf{w} \in \mathbb{B}(\mathbf{w}_s^*, \beta), \nabla^2 F(\mathbf{w}) \succeq \lambda \mathbf{I}$.



Assumption: Sufficient Network Connectivity in Decentralized Systems

Each reduced graph $\mathcal{G}_{\text{red}}(b)$ of the original network $\mathcal{G}(\mathcal{J}, \mathcal{E})$ must include at least one source component with a size greater than or equal to $(b+1)$.





Analyzing BRIDGE-T: Focusing on Local Updates

Expressing BRIDGE-T Updates: The BRIDGE-T update can be expressed using only values from nonfaulty nodes. For $[\mathbf{g}(t)]_j = [\nabla f_j(\mathbf{w}_j(t))]_k, j \in \mathcal{R}$, the update is:

$$\mathbf{\Omega}(t+1) = \mathbf{Y}(t)\mathbf{\Omega}(t) - \rho(t)\mathbf{g}(t).$$

Definitions and Cases: Define $\mathcal{N}_j^r := \mathcal{R} \cap \mathcal{N}_j$, $\mathcal{N}_j^b := \mathcal{N}_j \setminus \mathcal{N}_j^r$, $b^* := |\mathcal{B}|$, and b_j^k as the count of Byzantine nodes in $\mathcal{C}_j^k$. Consider $q_j^k := b - b^* + b_j^k$ with two scenarios: (i) $q_j^k = 0$ and (ii) $q_j^k > 0$.

Analyzing BRIDGE-T: Focusing on Local Updates

Expressing BRIDGE-T Updates: The BRIDGE-T update can be expressed using only values from nonfaulty nodes. For $[\mathbf{g}(t)]_j = [\nabla f_j(\mathbf{w}_j(t))]_k, j \in \mathcal{R}$, the update is:

$$\mathbf{\Omega}(t+1) = \mathbf{Y}(t)\mathbf{\Omega}(t) - \rho(t)\mathbf{g}(t).$$

Definitions and Cases: Define $\mathcal{N}_j^r := \mathcal{R} \cap \mathcal{N}_j$, $\mathcal{N}_j^b := \mathcal{N}_j \setminus \mathcal{N}_j^r$, $b^* := |\mathcal{B}|$, and b_j^k as the count of Byzantine nodes in $\mathcal{C}_j^k$. Consider $q_j^k := b - b^* + b_j^k$ with two scenarios: (i) $q_j^k = 0$ and (ii) $q_j^k > 0$.

Case (i) – Exact Filtering: If $b - b^* = 0$ and $b_j^k = 0$, $\mathbf{Y}$ can be described as:

$$[\mathbf{Y}]_{ji} = \begin{cases} \frac{1}{|\mathcal{N}_j| - 2b + 1}, & i \in \{j\} \cup \mathcal{C}_j^k, \\ 0, & \text{otherwise.} \end{cases}$$

Further Analysis of BRIDGE-T: Focusing on Local Updates

Case (ii) – Over and Miss Filtering: This involves either over filtering $(b - b^* > 0)$ and/or miss filtering $(b_j^k > 0)$. Given $\overline{\mathcal{N}}_j^k \cap \mathcal{N}_j^r \neq \emptyset$ and $\underline{\mathcal{N}}_j^k \cap \mathcal{N}_j^r \neq \emptyset$, there exist m_j', m_j'' such that for all $i \in \mathcal{C}_j^k$, $[\mathbf{w}_i]_k = \theta_i [\mathbf{w}_{m_j'}]_k + (1-\theta_i)[\mathbf{w}_{m_j''}]_k$ with $\theta_i \in (0,1)$.

Further Analysis of BRIDGE-T: Focusing on Local Updates

Case (ii) – Over and Miss Filtering: This involves either over filtering $(b - b^* > 0)$ and/or miss filtering $(b_j^k > 0)$. Given $\overline{\mathcal{N}}_j^k \cap \mathcal{N}_j^r \neq \emptyset$ and $\underline{\mathcal{N}}_j^k \cap \mathcal{N}_j^r \neq \emptyset$, there exist m_j', m_j'' such that for all $i \in \mathcal{C}_j^k$, $[\mathbf{w}_i]_k = \theta_i[\mathbf{w}_{m_j'}]_k + (1-\theta_i)[\mathbf{w}_{m_j''}]_k$ with $\theta_i \in (0,1)$.

Matrix $\mathbf{Y}$ Elements:

$$[\mathbf{Y}]_{ji} = \begin{cases} \frac{1}{2(|\mathcal{N}_j|-2b+1)}, & i \in \mathcal{N}_j^r \cap \mathcal{C}_j^k, \\ \frac{1}{|\mathcal{N}_j|-2b+1}, & i = j, \\ \sum_{i' \in \mathcal{N}_j^b \cap \mathcal{C}_j^k} \frac{\theta_{i'}}{q_j^k(|\mathcal{N}_j|-2b+1)} + \sum_{i' \in \mathcal{N}_j^r \cap \mathcal{C}_j^k} \frac{\theta_{i'}}{q_j^k(|\mathcal{N}_j|-2b+1)}, & i \in \overline{\mathcal{N}}_j^k \cap \mathcal{N}_j^r, \\ \sum_{i' \in \mathcal{N}_j^b \cap \mathcal{C}_j^k} \frac{1-\theta_{i'}}{q_j^k(|\mathcal{N}_j|-2b+1)} + \sum_{i' \in \mathcal{N}_j^r \cap \mathcal{C}_j^k} \frac{1-\theta_{i'}}{q_j^k(|\mathcal{N}_j|-2b+1)}, & i \in \underline{\mathcal{N}}_j^k \cap \mathcal{N}_j^r, \\ 0, & \text{otherwise.} \end{cases}$$

Main Result: Achieving Consensus in the Network

Consensus Vector Definition: A "consensus vector" $\mathbf{v}(t) \in \mathbb{R}^d$ is defined as one whose k-th entry $[\mathbf{v}(t)]_k$ is given by any element from $\bar{\mathbf{v}}(t) = \lim_{T\to\infty} \mathbf{\Omega}(t+T+1)$.

Theorem (Consensus; Fang et al., 2022)

Given a step-size sequence $\rho(t)$ satisfying $\rho(t+1) \leq \rho(t)$, $\rho(t) \to 0$, $\sum_{t=0}^{\infty} \rho(t) = \infty$, and $\sum_{t=0}^{\infty} \rho^2(t) < \infty$, with $\rho(t) = \frac{1}{\lambda(t_0+t)}$ and $t_0 \geq \frac{L}{\lambda}$, under bounded and Lipschitz gradients and sufficient network connectivity conditions, the gap between $\mathbf{w}_j(t)$ for all $j \in \mathcal{R}$ and $\mathbf{v}(t)$ diminishes over time:

$$\lim_{t\to\infty} \max_{j\in\mathcal{R}} \|\mathbf{w}_j(t) - \mathbf{v}(t)\| \leq \lim_{t\to\infty} \left[\sqrt{d} r C_w \mu^{\frac{t}{\nu}} + \sqrt{d} r L \sum_{\tau=0}^{t} \rho(\tau) \mu^{\frac{t-\tau+1}{\nu}} \right] = 0.$$

Conditions and Rate of Consensus Convergence

Convergence Conditions

- Bounded Initialization: Ensure $|[\mathbf{w}_j(0)]_k| \leq C_w$ for the initial iterates.
- Independent of the assumptions of strong convexity or local strong convexity.

Rate of Convergence

- Dominated by $\sqrt{d} r L \sum_{\tau=0}^{t} \rho(\tau) \mu^{\frac{t-\tau+1}{\nu}}$, particularly by the subterm $\rho(\tau)$.
- Attains a rate of $\mathcal{O}(\sqrt{d}\rho(t)) = \mathcal{O}(\sqrt{d}/t)$, assuming $\rho(t)$ is chosen as $\mathcal{O}(1/t)$.

Main Result: Statistical Convergence for Strongly Convex Loss Functions

Theorem (Strongly Convex Loss Functions; Fang et al., 2022)

Under the assumptions of bounded and Lipschitz gradients, bounded training loss, strong convexity, sufficient network connectivity, and i.i.d. training data, for any $\epsilon > \frac{\epsilon''}{\lambda} > 0$, and for all $j \in \mathcal{R}$, with probability at least $1-\delta$ and for sufficiently large t:

$$\|\mathbf{w}_j(t+1) - \mathbf{w}^*\| \leq \epsilon, \tag{1}$$

where δ and ϵ'' are defined as:

$$\delta = 2\exp\left(-\frac{4rN\epsilon''^2}{16L^2rd\|\boldsymbol{\alpha}_m\|^2 + \epsilon''^2} + r\log\left(\frac{12L\sqrt{rd}}{\epsilon''}\right) + d\log\left(\frac{12L'\beta\sqrt{d}}{\epsilon''}\right)\right),$$

$$\epsilon'' = \mathcal{O}\left(\sqrt{\frac{d\|\boldsymbol{\alpha}_m\|^2\log\frac{2}{\delta}}{N}}\right).$$

Main Result: Statistical Convergence for Nonconvex Loss Functions

Lemma (Behavior of Iterates; Fang et al., 2022)

Under the assumptions of bounded and Lipschitz gradients, bounded training loss, local strong convexity, and sufficient network connectivity, with i.i.d. training data, and initialization vector $\mathbf{v}(0) \in \mathbb{B}(\mathbf{w}^*, \beta_0)$ for an appropriate $\beta_0 < \beta$, the iterates $\mathbf{w}_j(t)$ will not escape from $\mathbb{B}(\mathbf{w}^*, \beta)$ for all $j \in \mathcal{R}, t \in \mathbb{R}$.

Main Result: Statistical Convergence for Nonconvex Loss Functions

Lemma (Behavior of Iterates; Fang et al., 2022)

Under the assumptions of bounded and Lipschitz gradients, bounded training loss, local strong convexity, and sufficient network connectivity, with i.i.d. training data, and initialization vector $\mathbf{v}(0) \in \mathbb{B}(\mathbf{w}^*, \beta_0)$ for an appropriate $\beta_0 < \beta$, the iterates $\mathbf{w}_j(t)$ will not escape from $\mathbb{B}(\mathbf{w}^*, \beta)$ for all $j \in \mathcal{R}, t \in \mathbb{R}$.

Theorem (Locally Strongly Convex Loss Functions; Fang et al., 2022)

Given the aforementioned assumptions and i.i.d. training data, with appropriate initialization, for any $\epsilon > \frac{\epsilon''}{\lambda} > 0$ and sufficiently large t, all $j \in \mathcal{R}$ will satisfy with probability at least $1 - \delta$:

$$\|\mathbf{w}_j(t+1) - \mathbf{w}^*\| \leq \epsilon, \tag{2}$$

where δ and ϵ'' are defined as:

$$\delta = 2\exp\left(-\frac{4rN\epsilon''^2}{16L^2rd\|\boldsymbol{\alpha}_m\|^2 + \epsilon''^2} + r\log\left(\frac{12L\sqrt{rd}}{\epsilon''}\right) + d\log\left(\frac{12L'\beta\sqrt{d}}{\epsilon''}\right)\right), \quad \text{and} \quad \epsilon'' = \mathcal{O}\left(\sqrt{\frac{d\|\boldsymbol{\alpha}_m\|^2\log\frac{2}{\delta}}{N}}\right).$$

Algorithmic and Statistical Learning Rates Comparison

Method	**Centralized GD**	**DGD**	**BRIDGE-T**
Convexity	Strongly Convex	Convex	Locally Strongly Convex
Step Size	Constant	Diminishing	Diminishing
Algorithmic Rate	$\mathcal{O}(c^t)$	$\mathcal{O}\left(\frac{1}{\sqrt{t}}\right)$	$\mathcal{O}\left(\frac{\ln t}{t}\right)$
Statistical Rate	$\mathcal{O}\left(\sqrt{\frac{1}{N}}\right)$	$\mathcal{O}\left(\sqrt{\frac{1}{MN}}\right)$	$\mathcal{O}\left(\sqrt{\frac{\|\boldsymbol{\alpha}_m\|^2}{N}}\right)$

Note: The vector $\boldsymbol{\alpha}_m \in \mathbb{R}^r$ is problem-dependent, with $[\boldsymbol{\alpha}_m]_j \geq 0$ and $\sum_{j=1}^{r}[\boldsymbol{\alpha}_m]_j = 1$.

Outline

1 Motivation

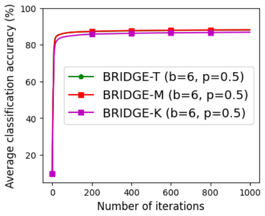
2 Problem Formulation

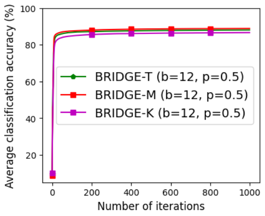
3 The BRIDGE Framework

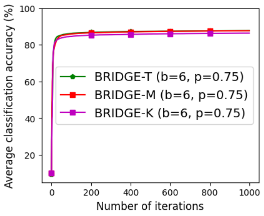
4 Theoretical Analysis

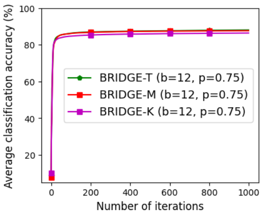
5 Numerical Results

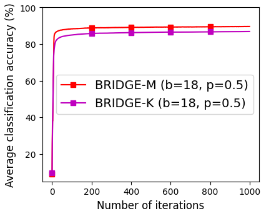
Convex Case with MNIST Dataset

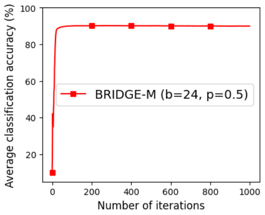


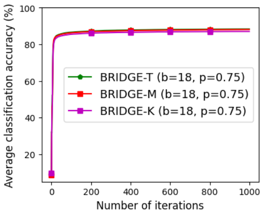


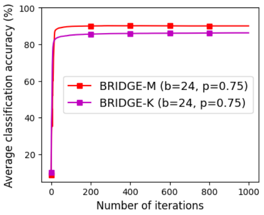




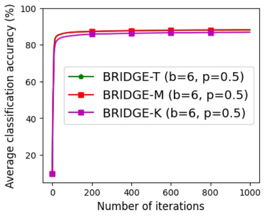


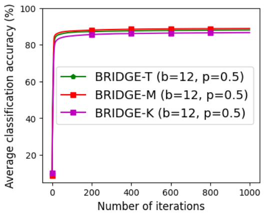


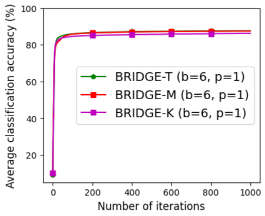


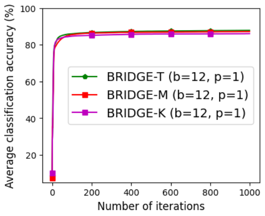


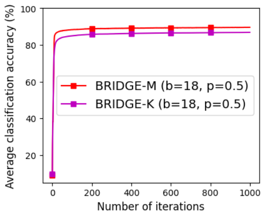
Convex Case with MNIST Dataset

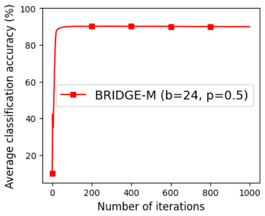


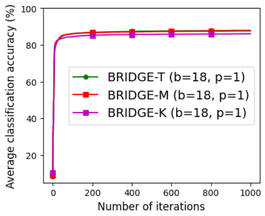


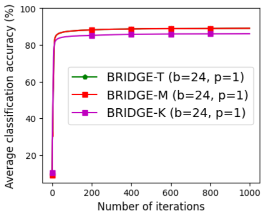




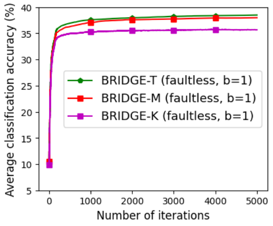


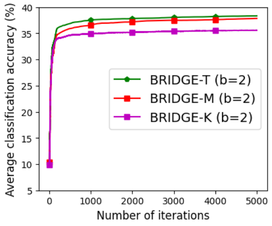


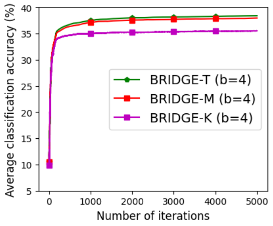


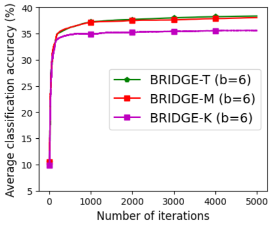


Convex Case with CIFAR10 Dataset



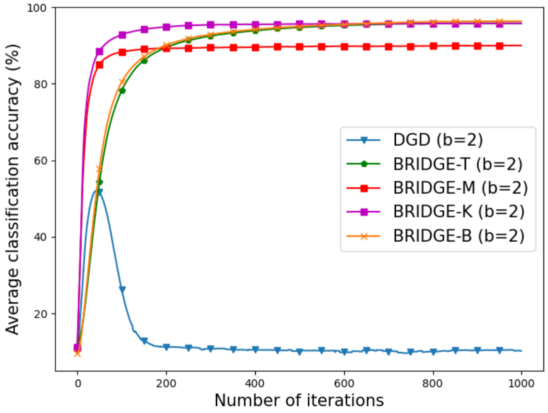


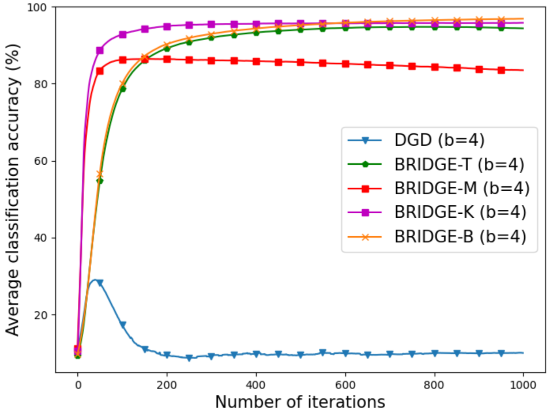




Nonconvex Case with MNIST Dataset

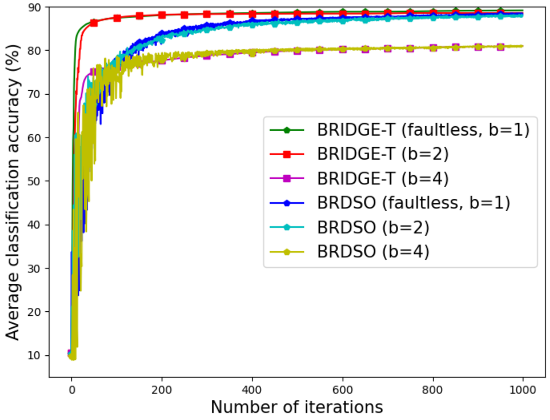
- Experiment conducted on the MNIST dataset with a 50-node network ($p = 0.5$).
- Utilized a convolutional neural network (nonconvex) with i.i.d. data distribution.



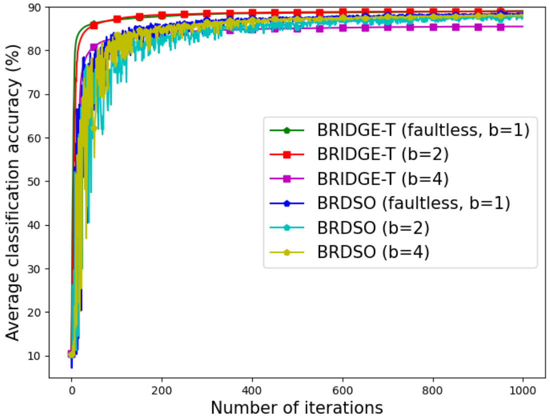


Convex Case with MNIST Dataset: Non-I.I.D. Data Distribution

Experiments conducted on the MNIST dataset, within a 50-node network (connectivity $p = 0.5$; BRDSO [Peng, W. Li, and Ling, 2021]).



Performance under Extreme Non-I.I.D. Scenario



Performance under Moderate Non-I.I.D. Scenario

Conclusion and Future Work

Summary of Work

- Developed and analyzed the BRIDGE framework for decentralized machine learning, emphasizing resilience to Byzantine failures.
- Demonstrated efficacy through numerical results with various datasets and network conditions.

Future Work

- Explore faster convergence rates in strongly convex settings using accelerated methods.
- Theoretical analysis of alternative screening methods beyond the trimmed-mean approach.
- Analyze second-order convergence guarantees for general nonconvex loss functions.
- Investigate the impact of various factors like asynchronous communication, non-i.i.d. data distribution, non-smooth objective functions, and diverse network topologies.

Bibliography I

- Blanchard, P., R. Guerraoui, and J. Stainer (2017). "Machine Learning with Adversaries: Byzantine Tolerant Gradient Descent". In: *Proc. Advances in Neural Inf. Process. Syst.* Pp. 118–128.
- Guo, S. et al. (2020). "Towards Byzantine-resilient Learning in Decentralized Systems". In: *arXiv preprint arXiv:2002.08569.*
- Konečný, J. et al. (2016). "Federated Learning: Strategies for Improving Communication Efficiency". In: *Proc. NeurIPS Workshop on Private Multi-Party Machine Learning.*
- Kuwaranancharoen, K., L. Xin, and S. Sundaram (2020). "Byzantine-Resilient Distributed Optimization of Multi-Dimensional Functions". In: *Proc. American Control Conference (ACC)*, pp. 4399–4404.
- Lamport, L., R. Shostak, and M. Pease (1982). "The Byzantine generals problem". In: *ACM Trans. Programming Languages and Syst.* 4.3, pp. 382–401.
- LeBlanc, H. J. et al. (2013). "Resilient asymptotic consensus in robust networks". In: *IEEE J. Sel. Areas in Commun.* 31.4, pp. 766–781.
- Li, M. et al. (2014). "Scaling Distributed Machine Learning with the Parameter Server". In: *Proc. 11th USENIX Symp. Operating Systems Design and Implementation (OSDI'14).* Broomfield, CO, pp. 583–598.
- Lorenzo, P. D. and G. Scutari (2016). "NEXT: In-Network Nonconvex Optimization". In: *IEEE Transactions on Signal and Information Processing over Networks* 2, pp. 120–136.
- Mhamdi, E. E., R. Guerraoui, and S. Rouault (2018). "The Hidden Vulnerability of Distributed Learning in Byzantium". In: *Proc. 35th Int. Conf. Machine Learning*, pp. 3521–3530.
- El-Mhamdi, E.-M. et al. (2020). "Collaborative Learning as an Agreement Problem". In: *arXiv preprint arXiv:2008.00742v3.*
- Nedić, A. and A. Olshevsky (2015). "Distributed optimization over time-varying directed graphs". In: *IEEE Trans. Autom. Control* 60.3, pp. 601–615.
- Nedić, A. and A. Ozdaglar (2009). "Distributed Subgradient Methods for Multi-Agent Optimization". In: *IEEE Trans. Autom. Control* 54.1, pp. 48–61.

Bibliography II

- Peng, J., W. Li, and Q. Ling (2021). "Byzantine-robust decentralized stochastic optimization over static and time-varying networks". In: *Signal Processing* 183, p. 108020.
- Pu, S. and A. Nedić (2021). "Distributed Stochastic Gradient Tracking Methods". In: *Mathematical Programming* 187, pp. 409–457.
- Ram, S. S., A. Nedić, and V. Veeravalli (2010). "Distributed stochastic subgradient projection algorithms for convex optimization". In: *J. Optim. Theory and Appl.* 147.3, pp. 516–545.
- Su, L. and N. H. Vaidya (2016). "Fault-tolerant multi-agent optimization: Optimal iterative distributed algorithms". In: *Proc. ACM Symp. Principles of Distributed Computing*, pp. 425–434.
- Sun, H., S. Lu, and M. Hong (2020). "Improving the Sample and Communication Complexity for Decentralized Non-Convex Optimization: Joint Gradient Estimation and Tracking". In: *Proc. 37th Intl. Conf. Machine Learning*, pp. 9217–9228.
- Sundaram, S. and B. Gharesifard (2019). "Distributed optimization under adversarial nodes". In: *IEEE Trans. Autom. Control* 64.3, pp. 1063–1076.
- Vaidya, N. H., L. Tseng, and G. Liang (2014). "Iterative Byzantine vector consensus in incomplete graphs". In: *Proc. 15th Int. Conf. Distributed Computing and Networking*, pp. 14–28.
- Xin, R., U. A. Khan, and S. Kar (2020). "Fast decentralized non-convex finite-sum optimization with recursive variance reduction". In: *arXiv preprint arXiv:2008.07428*.
- Yang, Z. and W. U. Bajwa (2019). "ByRDiE: Byzantine-Resilient Distributed Coordinate Descent for Decentralized Learning". In: *IEEE Trans. Signal Inf. Process. Netw.* 5.4, pp. 611–627.
- Yin, D. et al. (2018). "Byzantine-robust distributed learning: Towards optimal statistical rates". In: *Proc. 35th Intl. Conf. Machine Learning*, pp. 5650–5659.
- Zeng, J. and W. Yin (2018). "On Nonconvex Decentralized Gradient Descent". In: *IEEE Transactions on Signal Processing* 66, pp. 2834–2848.