Randomized Douglas-Rachford Splitting Algorithms for Nonconvex Federated Composite Optimization

Quoc Tran-Dinh
quoctd@email.unc.edu

Department of Statistics and Operations Research
The University of North Carolina at Chapel Hill (UNC)

Variational Analysis and Optimization Webinar
Remote [September 29, 2021]

Joint work with

Nhan Pham (UNC), Dzung Phan (IBM), and Lam Nguyen (IBM)

THE UNIVERSITY
of NORTH CAROLINA
at CHAPEL HILL

Reference

This talk is based on the following manuscript:

- Q. T-D, N. Pham, D. Phan, and L. Nguyen: **FedDR – Randomized Douglas-Rachford Splitting Algorithms for Nonconvex Federated Composite Optimization**, ***March, 2021.***

  Preprint: https://arxiv.org/abs/2103.03452.

Outline

Outline

Problem Statement, Motivation, and Contribution

Federated Learning with Randomized DR – FedDR

Federated Learning with Asynchronous DR – assyncFedDR

Numerical Examples

Conclusions and Future Research

Optimization Model of Federated Learning

Composite Optimization Model in Federated Learning (FL)

$$F^\star = \min_{x \in \mathbb{R}^p} \Big\{ F(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x) + g(x) \Big\}, \tag{1}$$

- $f_i : \mathbb{R}^p \to \mathbb{R} \cup \{+\infty\}$ $(i = 1, \cdots, n)$ are smooth and possibly nonconvex;
- $g : \mathbb{R}^p \to \mathbb{R}$ is a convex and possibly nonsmooth;
- Define $f(x) := \frac{1}{n} \sum_{i=1}^{n} f_i(x)$ as a **finite-sum function**.

Optimization Model of Federated Learning

Composite Optimization Model in Federated Learning (FL)

$$F^\star = \min_{x\in\mathbb{R}^p}\left\{F(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x) + g(x)\right\}, \tag{1}$$

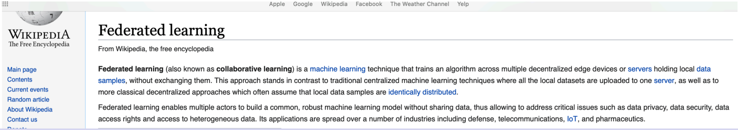
- $f_i : \mathbb{R}^p \to \mathbb{R}\cup\{+\infty\}$ $(i = 1,\cdots,n)$ are smooth and possibly nonconvex;
- $g : \mathbb{R}^p \to \mathbb{R}$ is a convex and possibly nonsmooth;
- Define $f(x) := \frac{1}{n}\sum_{i=1}^{n} f_i(x)$ as a **finite-sum function**.

Assumption 1 (Model Assumptions)

- *f_i is smooth and possibly nonconvex, and g is convex and possibly nonsmooth.*
- *The **domain** of F dom$F := \{x\in\mathbb{R}^p : F(x) < +\infty\}$ is nonempty.*
- *There exists a [first-order] **stationary point** of (1), i.e., $0 \in \nabla f(x^*) + \partial g(x^*)$.*
- ***Boundedness from below:** $F^\star := \inf_{x\in\mathbb{R}^p} F(x) > -\infty$.*
- *All functions $f_i(\cdot)$ for $i\in[n] := \{1,\cdots,n\}$ are L-smooth, i.e.:*

$$\|\nabla f_i(x) - \nabla f_i(y)\| \le L\|x-y\|, \quad \forall x,y \in \mathrm{dom} f_i. \tag{2}$$

Federated Learning and Challenges


Apple Google Wikipedia Facebook The Weather Channel Yelp

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Current events
Random article
About Wikipedia
Contact us

Federated learning

From Wikipedia, the free encyclopedia

Federated learning (also known as **collaborative learning**) is a machine learning technique that trains an algorithm across multiple decentralized edge devices or servers holding local data samples, without exchanging them. This approach stands in contrast to traditional centralized machine learning techniques where all the local datasets are uploaded to one server, as well as to more classical decentralized approaches which often assume that local data samples are identically distributed.

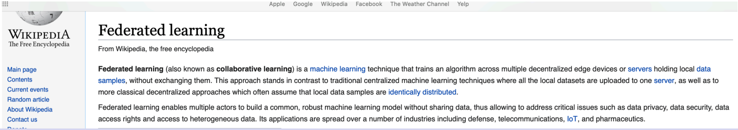
Federated learning enables multiple actors to build a common, robust machine learning model without sharing data, thus allowing to address critical issues such as data privacy, data security, data access rights and access to heterogeneous data. Its applications are spread over a number of industries including defense, telecommunications, IoT, and pharmaceutics.

What is FL?

FL is a **machine learning technique** that

- trains an algorithm across multiple decentralized edge devices/users
- local devices/users hold data samples locally without exchanging them.

Federated Learning and Challenges


Apple Google Wikipedia Facebook The Weather Channel Yelp

WIKIPEDIA
The Free Encyclopedia

Main page
Contents
Current events
Random article
About Wikipedia
Contact us

Federated learning

From Wikipedia, the free encyclopedia

Federated learning (also known as **collaborative learning**) is a machine learning technique that trains an algorithm across multiple decentralized edge devices or servers holding local data samples, without exchanging them. This approach stands in contrast to traditional centralized machine learning techniques where all the local datasets are uploaded to one server, as well as to more classical decentralized approaches which often assume that local data samples are identically distributed.

Federated learning enables multiple actors to build a common, robust machine learning model without sharing data, thus allowing to address critical issues such as data privacy, data security, data access rights and access to heterogeneous data. Its applications are spread over a number of industries including defense, telecommunications, IoT, and pharmaceutics.

What is FL?

FL is a **machine learning technique** that

- trains an algorithm across multiple decentralized edge devices/users
- local devices/users hold data samples locally without exchanging them.

Main Challenges of FL

- **Communication bottleneck:** If the number of users n is large, it creates *communication bottleneck* during **model exchange process between server and users**.
- **Data or statistical heterogeneity:** The local data in each user may be different in terms of sizes and distribution.
- **System heterogeneity:** The variety of users with different local storage, computational power, and network connectivity also creates a major challenge.
- **Privacy concern**: Accessing and sharing local raw data is not permitted in FL.

Our Approach and Contribution

Our approach

- **Our goal:** Simultaneously address fundamental challenges through **two new algorithms** for ***FL composite nonconvex optimization model*** (1).
- **Our approach:** Rely on a **novel combination** between randomized block-coordinate strategy, nonconvex Douglas-Rachford (DR) splitting, and asynchronous variant.

Our Approach and Contribution

Our approach

- **Our goal:** Simultaneously address fundamental challenges through **two new algorithms** for ***FL composite nonconvex optimization model*** (1).
- **Our approach:** Rely on a **novel combination** between randomized block-coordinate strategy, nonconvex Douglas-Rachford (DR) splitting, and asynchronous variant.

Our contribution

(a) **New Federated Douglas-Rachford algorithm – FedDR**

- Combining DR splitting technique and randomized block-coordinate strategy for **nonconvex composite optimization problem** in FL.
- Can handle nonsmooth convex regularizers and inexact evaluation of prox operations.
- Achieves the best known $\mathcal{O}\left(\varepsilon^{-2}\right)$ communication complexity for finding a **stationary point** under standard assumptions.
- Does not require all users to participate in each communication round.

Our Approach and Contribution

Our approach

- **Our goal:** Simultaneously address fundamental challenges through **two new algorithms** for ***FL composite nonconvex optimization model*** (1).
- **Our approach:** Rely on a **novel combination** between randomized block-coordinate strategy, nonconvex Douglas-Rachford (DR) splitting, and asynchronous variant.

Our contribution

(a) **New Federated Douglas-Rachford algorithm – FedDR**

- Combining DR splitting technique and randomized block-coordinate strategy for **nonconvex composite optimization problem** in FL.
- Can handle nonsmooth convex regularizers and inexact evaluation of prox operations.
- Achieves the best known $\mathcal{O}\left(\varepsilon^{-2}\right)$ communication complexity for finding a **stationary point** under standard assumptions.
- Does not require all users to participate in each communication round.

(b) **New asynchronous FL Douglas-Rachford algorithm – asyncFedDR**

- Each user can asynchronously perform local update and periodically send the update to the server for **aggregation**.
- Achieves the same $\mathcal{O}\left(\varepsilon^{-2}\right)$ communication complexity as **FedDR** (up to a constant factor) under standard assumptions.

Some Related Work in FL

Notable results closely related to our work

- **FedAvg:** ***Federated Averaging*** (**FedAvg**) is one of the first and popular methods for FL [Konevcny et al (2016), McMahan et al (2017)].

  One of the early attempts to show the convergence of **FedAvg** is [Stich (2018)].

Some Related Work in FL

Notable results closely related to our work

- **FedAvg:** ***Federated Averaging*** (**FedAvg**) is one of the first and popular methods for FL [Konevcny et al (2016), McMahan et al (2017)].

  One of the early attempts to show the convergence of **FedAvg** is [Stich (2018)].

- **FedProx:** [Li et al (2020)] is an extension of **FedAvg**, which deals with heterogeneity in federated networks by introducing a proximal term.

  FedProx has been shown to achieve better performance than **FedAvg** in heterogeneous setting.

Some Related Work in FL

Notable results closely related to our work

- **FedAvg:** ***Federated Averaging*** (**FedAvg**) is one of the first and popular methods for FL [Konevcny et al (2016), McMahan et al (2017)].

  One of the early attempts to show the convergence of **FedAvg** is [Stich (2018)].

- **FedProx:** [Li et al (2020)] is an extension of **FedAvg**, which deals with heterogeneity in federated networks by introducing a proximal term.

  FedProx has been shown to achieve better performance than **FedAvg** in heterogeneous setting.

- **SCAFFOLD:** [Karimireddy et al (2020)] use a control variate to correct the "client-drift" in local update of **FedAvg**.

  MIME: [Karimireddy et al (2020)] propose **MIME** another framework that uses control variate to improve **FedAvg** for heterogeneous settings.

Some Related Work in FL

Notable results closely related to our work

- **FedAvg:** ***Federated Averaging*** (**FedAvg**) is one of the first and popular methods for FL [Konevcny et al (2016), McMahan et al (2017)].

  One of the early attempts to show the convergence of **FedAvg** is [Stich (2018)].

- **FedProx:** [Li et al (2020)] is an extension of **FedAvg**, which deals with heterogeneity in federated networks by introducing a proximal term.

  FedProx has been shown to achieve better performance than **FedAvg** in heterogeneous setting.

- **SCAFFOLD:** [Karimireddy et al (2020)] use a control variate to correct the "client-drift" in local update of **FedAvg**.

  MIME: [Karimireddy et al (2020)] propose **MIME** another framework that uses control variate to improve **FedAvg** for heterogeneous settings.

- **FedSplit:** [Pathak & Wainwright (2020)] instead employs a **Peaceman-Rachford splitting** scheme to solve a constrained reformulation of the original problem.

Some Related Work in FL

Notable results closely related to our work

- **FedAvg:** ***Federated Averaging*** (**FedAvg**) is one of the first and popular methods for FL [Konevcny et al (2016), McMahan et al (2017)].

  One of the early attempts to show the convergence of **FedAvg** is [Stich (2018)].

- **FedProx:** [Li et al (2020)] is an extension of **FedAvg**, which deals with heterogeneity in federated networks by introducing a proximal term.

  FedProx has been shown to achieve better performance than **FedAvg** in heterogeneous setting.

- **SCAFFOLD:** [Karimireddy et al (2020)] use a control variate to correct the "client-drift" in local update of **FedAvg**.

  MIME: [Karimireddy et al (2020)] propose **MIME** another framework that uses control variate to improve **FedAvg** for heterogeneous settings.

- **FedSplit:** [Pathak & Wainwright (2020)] instead employs a **Peaceman-Rachford splitting** scheme to solve a constrained reformulation of the original problem.

- **FedPD:** [Zhang et al (2020)] propose **FedPD**, which is essentially a variant of the standard augmented Lagrangian method in nonlinear optimization.

Proximal Operator and Gradient Mapping

Proximal operators and evaluation

- **Our methods** make use of prox of both f_i and g though f_i is nonconvex.
- We define the **proximal operator** of f_i as

$$\mathrm{prox}_{\eta f_i}(x) := \arg\min_{y} \left\{ f_i(y) + \frac{1}{2\eta}\|y - x\|^2 \right\}, \qquad (\eta > 0). \tag{3}$$

- Even f_i is nonconvex, under **Assumption 1**, if we choose $0 < \eta < \frac{1}{L}$, then $\mathrm{prox}_{\eta f_i}$ is well-defined and single-valued.
- Evaluating $\mathrm{prox}_{\eta f_i}$ requires to solve a strongly convex program.

Proximal Operator and Gradient Mapping

Proximal operators and evaluation

- **Our methods** make use of prox of both f_i and g though f_i is nonconvex.
- We define the **proximal operator** of f_i as

$$\mathrm{prox}_{\eta f_i}(x) := \arg\min_{y} \left\{ f_i(y) + \tfrac{1}{2\eta}\|y - x\|^2 \right\}, \qquad (\eta > 0). \tag{3}$$

- Even f_i is nonconvex, under **Assumption 1**, if we choose $0 < \eta < \frac{1}{L}$, then $\mathrm{prox}_{\eta f_i}$ is well-defined and single-valued.
- Evaluating $\mathrm{prox}_{\eta f_i}$ requires to solve a strongly convex program.

Gradient mapping

The **gradient mapping** of F is defined as

$$\mathcal{G}_\eta(x) := \tfrac{1}{\eta}\Big(x - \mathrm{prox}_{\eta g}(x - \eta\nabla f(x))\Big), \quad \eta > 0. \tag{4}$$

The **optimality condition** $0 \in \nabla f(x^*) + \partial g(x^*)$ of (1) is equivalent to $\mathcal{G}_\eta(x^*) = 0$.

Proximal Operator and Gradient Mapping

Proximal operators and evaluation

- **Our methods** make use of prox of both f_i and g though f_i is nonconvex.
- We define the **proximal operator** of f_i as

$$\mathrm{prox}_{\eta f_i}(x) := \arg\min_y \left\{ f_i(y) + \tfrac{1}{2\eta}\|y - x\|^2 \right\}, \qquad (\eta > 0). \tag{3}$$

- Even f_i is nonconvex, under **Assumption 1**, if we choose $0 < \eta < \frac{1}{L}$, then $\mathrm{prox}_{\eta f_i}$ is well-defined and single-valued.
- Evaluating $\mathrm{prox}_{\eta f_i}$ requires to solve a strongly convex program.

Gradient mapping

The **gradient mapping** of F is defined as

$$\mathcal{G}_\eta(x) := \tfrac{1}{\eta}\Big(x - \mathrm{prox}_{\eta g}(x - \eta\nabla f(x))\Big), \quad \eta > 0. \tag{4}$$

The **optimality condition** $0 \in \nabla f(x^*) + \partial g(x^*)$ of (1) is equivalent to $\mathcal{G}_\eta(x^*) = 0$.

Definition 1 (Approximate Stationary Point)

If $\tilde{x} \in \mathrm{dom}F$ satisfies $\mathbb{E}\left[\|\mathcal{G}_\eta(\tilde{x})\|^2\right] \leq \varepsilon^2$, then $\tilde{x}$ is called an ε-**stationary point** of (1).

The Derivation of The Douglas-Rachford Splitting Scheme

Consider a convex minimization problem: $\min_{\mathbf{x}} \left\{ F(\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{x}) \right\}$.

The Derivation of The Douglas-Rachford Splitting Scheme

Consider a convex minimization problem: $\min_{\mathbf{x}} \left\{ F(\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{x}) \right\}$.

The derivation of DR splitting scheme

- Starting from the optimality condition $\nabla f(\mathbf{x}^\star) + \nabla g(\mathbf{x}^\star) = 0$, we write it as

$$\mathbf{x}^\star + \eta \nabla g(\mathbf{x}^\star) = 2\mathbf{x}^\star - \left[\mathbf{x}^\star + \eta \nabla f(\mathbf{x}^\star) \right], \qquad \eta > 0.$$

The Derivation of The Douglas-Rachford Splitting Scheme

Consider a convex minimization problem: $\min_{\mathbf{x}} \left\{ F(\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{x}) \right\}$.

The derivation of DR splitting scheme

- Starting from the optimality condition $\nabla f(\mathbf{x}^\star) + \nabla g(\mathbf{x}^\star) = 0$, we write it as

$$\mathbf{x}^\star + \eta \nabla g(\mathbf{x}^\star) = 2\mathbf{x}^\star - \left[\mathbf{x}^\star + \eta \nabla f(\mathbf{x}^\star) \right], \qquad \eta > 0.$$

- Define $\mathbf{y}^\star := \mathbf{x}^\star + \eta \nabla f(\mathbf{x}^\star) = (\mathbb{I} + \eta \nabla f)(\mathbf{x}^\star)$. Taking the inverse, we have

$$\mathbf{x}^\star = (\mathbb{I} + \eta \nabla f)^{-1}(\mathbf{y}^\star) = \mathrm{prox}_{\eta f}(\mathbf{y}^\star).$$

The Derivation of The Douglas-Rachford Splitting Scheme

Consider a convex minimization problem: $\min_{\mathbf{x}} \left\{ F(\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{x}) \right\}$.

The derivation of DR splitting scheme

- Starting from the optimality condition $\nabla f(\mathbf{x}^\star) + \nabla g(\mathbf{x}^\star) = 0$, we write it as

$$\mathbf{x}^\star + \eta \nabla g(\mathbf{x}^\star) = 2\mathbf{x}^\star - \left[\mathbf{x}^\star + \eta \nabla f(\mathbf{x}^\star) \right], \qquad \eta > 0.$$

- Define $\mathbf{y}^\star := \mathbf{x}^\star + \eta \nabla f(\mathbf{x}^\star) = (\mathbb{I} + \eta \nabla f)(\mathbf{x}^\star)$. Taking the inverse, we have

$$\mathbf{x}^\star = (\mathbb{I} + \eta \nabla f)^{-1}(\mathbf{y}^\star) = \mathrm{prox}_{\eta f}(\mathbf{y}^\star).$$

- From the above expression, we have $(\mathbb{I} + \eta \nabla g)(\mathbf{x}^\star) = 2\mathbf{x}^\star - \mathbf{y}^\star$, equivalently to

$$\mathbf{x}^\star = (\mathbb{I} + \eta \nabla g)^{-1}(2\mathbf{x}^\star - \mathbf{y}^\star) = \mathrm{prox}_{\eta g}(2\mathbf{x}^\star - \mathbf{y}^\star) = \mathrm{prox}_{\eta g}\left(2 \cdot \mathrm{prox}_{\eta f}(\mathbf{y}^\star) - \mathbf{y}^\star \right).$$

The Derivation of The Douglas-Rachford Splitting Scheme

Consider a convex minimization problem: $\min_{\mathbf{x}} \left\{ F(\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{x}) \right\}$.

The derivation of DR splitting scheme

- Starting from the optimality condition $\nabla f(\mathbf{x}^\star) + \nabla g(\mathbf{x}^\star) = 0$, we write it as

$$\mathbf{x}^\star + \eta\nabla g(\mathbf{x}^\star) = 2\mathbf{x}^\star - \left[\mathbf{x}^\star + \eta\nabla f(\mathbf{x}^\star)\right], \qquad \eta > 0.$$

- Define $\mathbf{y}^\star := \mathbf{x}^\star + \eta\nabla f(\mathbf{x}^\star) = (\mathbb{I} + \eta\nabla f)(\mathbf{x}^\star)$. Taking the inverse, we have

$$\mathbf{x}^\star = (\mathbb{I} + \eta\nabla f)^{-1}(\mathbf{y}^\star) = \mathrm{prox}_{\eta f}(\mathbf{y}^\star).$$

- From the above expression, we have $(\mathbb{I} + \eta\nabla g)(\mathbf{x}^\star) = 2\mathbf{x}^\star - \mathbf{y}^\star$, equivalently to

$$\mathbf{x}^\star = (\mathbb{I}+\eta\nabla g)^{-1}(2\mathbf{x}^\star - \mathbf{y}^\star) = \mathrm{prox}_{\eta g}(2\mathbf{x}^\star - \mathbf{y}^\star) = \mathrm{prox}_{\eta g}\left(2\cdot\mathrm{prox}_{\eta f}(\mathbf{y}^\star) - \mathbf{y}^\star\right).$$

- Using $\mathbf{x}^\star = \mathrm{prox}_{\eta f}(\mathbf{y}^\star)$, this is equivalent to

$$\mathrm{prox}_{\eta f}(\mathbf{y}^\star) = \mathrm{prox}_{\eta g}\left(2\cdot\mathrm{prox}_{\eta f}(\mathbf{y}^\star) - \mathbf{y}^\star\right).$$

The Derivation of The Douglas-Rachford Splitting Scheme

Consider a convex minimization problem: $\min_{\mathbf{x}} \left\{ F(\mathbf{x}) = f(\mathbf{x}) + g(\mathbf{x}) \right\}$.

The derivation of DR splitting scheme

- Starting from the optimality condition $\nabla f(\mathbf{x}^\star) + \nabla g(\mathbf{x}^\star) = 0$, we write it as

$$\mathbf{x}^\star + \eta \nabla g(\mathbf{x}^\star) = 2\mathbf{x}^\star - \left[\mathbf{x}^\star + \eta \nabla f(\mathbf{x}^\star)\right], \qquad \eta > 0.$$

- Define $\mathbf{y}^\star := \mathbf{x}^\star + \eta \nabla f(\mathbf{x}^\star) = (\mathbb{I} + \eta \nabla f)(\mathbf{x}^\star)$. Taking the inverse, we have

$$\mathbf{x}^\star = (\mathbb{I} + \eta \nabla f)^{-1}(\mathbf{y}^\star) = \mathrm{prox}_{\eta f}(\mathbf{y}^\star).$$

- From the above expression, we have $(\mathbb{I} + \eta \nabla g)(\mathbf{x}^\star) = 2\mathbf{x}^\star - \mathbf{y}^\star$, equivalently to

$$\mathbf{x}^\star = (\mathbb{I} + \eta \nabla g)^{-1}(2\mathbf{x}^\star - \mathbf{y}^\star) = \mathrm{prox}_{\eta g}(2\mathbf{x}^\star - \mathbf{y}^\star) = \mathrm{prox}_{\eta g}\left(2 \cdot \mathrm{prox}_{\eta f}(\mathbf{y}^\star) - \mathbf{y}^\star\right).$$

- Using $\mathbf{x}^\star = \mathrm{prox}_{\eta f}(\mathbf{y}^\star)$, this is equivalent to

$$\mathrm{prox}_{\eta f}(\mathbf{y}^\star) = \mathrm{prox}_{\eta g}\left(2 \cdot \mathrm{prox}_{\eta f}(\mathbf{y}^\star) - \mathbf{y}^\star\right).$$

- We can write this equivalently to

$$\mathbf{y}^\star = \mathbf{y}^\star + \alpha\left[\mathrm{prox}_{\eta g}\left(2 \cdot \mathrm{prox}_{\eta f}(\mathbf{y}^\star) - \mathbf{y}^\star\right) - \mathrm{prox}_{\eta f}(\mathbf{y}^\star) = \mathcal{T}_{\mathrm{DR}}(\mathbf{y}^\star)\right].$$

- $\mathbf{y}^\star$ is a fixed-point of $\mathcal{T}_{\mathrm{DR}}(\mathbf{y}) := \mathbf{y} + \alpha[\mathrm{prox}_{\eta g}(2 \cdot \mathrm{prox}_{\eta f}(\mathbf{y}) - \mathbf{y}) - \mathrm{prox}_{\eta f}(\mathbf{y})]$.

The Douglas-Rachford Splitting Method

The fixed-point iteration of DR splitting method

Starting from $\mathbf{y}_0$, compute a sequence $\{\mathbf{y}_k\}$ by

$$\mathbf{y}_{k+1} := \mathcal{T}_{\mathrm{DR}}(\mathbf{y}_k).$$

Then $\mathbf{x}_k = \mathrm{prox}_{\eta f}(\mathbf{y}_k)$ is an approximate solution of the problem.

Note: The fixed-point $\mathbf{y}^\star$ of $T_{\mathrm{DR}}(\cdot)$ is not a solution of our problem, but $\mathbf{x}^\star = \mathrm{prox}_{\eta f}(\mathbf{y}^\star)$ is.

The Douglas-Rachford Splitting Method

The fixed-point iteration of DR splitting method

Starting from $\mathbf{y}_0$, compute a sequence $\{\mathbf{y}_k\}$ by

$$\mathbf{y}_{k+1} := \mathcal{T}_{\mathrm{DR}}(\mathbf{y}_k).$$

Then $\mathbf{x}_k = \mathrm{prox}_{\eta f}(\mathbf{y}_k)$ is an approximate solution of the problem.

Note: The fixed-point $\mathbf{y}^\star$ of $T_{\mathrm{DR}}(\cdot)$ is not a solution of our problem, but $\mathbf{x}^\star = \mathrm{prox}_{\eta f}(\mathbf{y}^\star)$ is.

The implementation of DR splitting method

- Starting from $\mathbf{y}_0$, update

$$\begin{cases} \mathbf{w}_k & := & \mathrm{prox}_{\eta f}(\mathbf{y}_k), \\ \mathbf{v}_k & := & \mathrm{prox}_{\eta g}(2\mathbf{w}_k - \mathbf{y}_k), \\ \mathbf{y}_{k+1} & := & \mathbf{y}_k + \alpha(\mathbf{v}_k - \mathbf{w}_k). \end{cases}$$

- Finally, compute $\mathbf{x}_k = \mathrm{prox}_{\eta f}(\mathbf{y}_k)$.
- We can circulate these three steps to get different updating orders.
- We can use a change of variable to get different interpretation.

Outline

Problem Statement, Motivation, and Contribution

Federated Learning with Randomized DR – FedDR

Federated Learning with Asynchronous DR – assyncFedDR

Numerical Examples

Conclusions and Future Research

Equivalent Reformulations of FL Optimization Model

Constrained reformulation – Duplicating variables

$$\begin{cases} \min\limits_{x_1,\cdots,x_n} & \left\{ F(\mathbf{x}) := f(\mathbf{x}) + g(\mathbf{x}) \equiv \frac{1}{n}\sum_{i=1}^{n} f_i(x_i) + g(x_1) \right\} \\ \text{s.t.} & x_2 = x_1,\ x_3 = x_1,\ \cdots, x_n = x_1. \end{cases} \tag{5}$$

where $\mathbf{x} := [x_1, x_2, \cdots, x_n]$ **concatenates** n variables x_i $(i \in [n])$.

Define a **linear subspace**:

$$\mathcal{L} := \{\mathbf{x} \in \mathbb{R}^{np} : x_2 = x_1,\ x_3 = x_1, \cdots, x_n = x_1\} \subset \mathbb{R}^{np}.$$

Equivalent Reformulations of FL Optimization Model

Constrained reformulation – Duplicating variables

$$\left\{ \begin{array}{ll} \displaystyle\min_{x_1,\cdots,x_n} & \left\{ F(\mathbf{x}) := f(\mathbf{x}) + g(\mathbf{x}) \equiv \frac{1}{n}\sum_{i=1}^{n} f_i(x_i) + g(x_1) \right\} \\ \text{s.t.} & x_2 = x_1,\ x_3 = x_1,\ \cdots, x_n = x_1. \end{array} \right. \quad (5)$$

where $\mathbf{x} := [x_1, x_2, \cdots, x_n]$ **concatenates** n variables x_i $(i \in [n])$.

Define a **linear subspace**:

$$\mathcal{L} := \{\mathbf{x} \in \mathbb{R}^{np} : x_2 = x_1,\ x_3 = x_1, \cdots, x_n = x_1\} \subset \mathbb{R}^{np}.$$

Unconstrained reformulation – Handling constraints

Let $\delta_{\mathcal{L}}$ be the **indicator function** of $\mathcal{L}$. We can rewrite (5) as

$$\min_{\mathbf{x} \in \mathbb{R}^{np}} \left\{ F(\mathbf{x}) := f(\mathbf{x}) + g(\mathbf{x}) + \delta_{\mathcal{L}}(\mathbf{x}) \equiv \frac{1}{n}\sum_{i=1}^{n} f_i(x_i) + g(x_1) + \delta_{\mathcal{L}}(\mathbf{x}) \right\}. \quad (6)$$

Equivalent Reformulations of FL Optimization Model

Constrained reformulation – Duplicating variables

$$\begin{cases} \displaystyle\min_{x_1,\cdots,x_n} & \left\{ F(\mathbf{x}) := f(\mathbf{x}) + g(\mathbf{x}) \equiv \frac{1}{n}\sum_{i=1}^n f_i(x_i) + g(x_1) \right\} \\ \text{s.t.} & x_2 = x_1,\ x_3 = x_1,\ \cdots, x_n = x_1. \end{cases} \tag{5}$$

where $\mathbf{x} := [x_1, x_2, \cdots, x_n]$ **concatenates** n variables x_i $(i \in [n])$.

Define a **linear subspace**:

$$\mathcal{L} := \{\mathbf{x} \in \mathbb{R}^{np} : x_2 = x_1,\ x_3 = x_1, \cdots, x_n = x_1\} \subset \mathbb{R}^{np}.$$

Unconstrained reformulation – Handling constraints

Let $\delta_{\mathcal{L}}$ be the **indicator function** of $\mathcal{L}$. We can rewrite (5) as

$$\min_{\mathbf{x}\in\mathbb{R}^{np}} \left\{ F(\mathbf{x}) := f(\mathbf{x}) + g(\mathbf{x}) + \delta_{\mathcal{L}}(\mathbf{x}) \equiv \frac{1}{n}\sum_{i=1}^n f_i(x_i) + g(x_1) + \delta_{\mathcal{L}}(\mathbf{x}) \right\}. \tag{6}$$

Equivalence between (6) and (1)

A stationary point $\mathbf{x}^\star$ of (6) $\Leftrightarrow$ $x_1^\star$ is a stationary point of (1).

Derivation of Douglas-Rachford Splitting Scheme for FL

Douglas-Rachford splitting steps for (6)

$$\begin{cases} \mathbf{y}^{k+1} & := & \mathbf{x}^k + \alpha(\bar{\mathbf{x}}^k - \mathbf{x}^k), \\ \mathbf{x}^{k+1} & := & \mathrm{prox}_{n\eta f}(\mathbf{y}^{k+1}), \\ \bar{\mathbf{x}}^{k+1} & := & \mathrm{prox}_{n\eta(g+\delta_{\mathcal{L}})}(2\mathbf{x}^{k+1} - \mathbf{y}^{k+1}), \end{cases} \tag{7}$$

where $\eta > 0$ is a given **step-size** and $\alpha \in (0, 2]$ is a **relaxation parameter**.

Derivation of Douglas-Rachford Splitting Scheme for FL

Douglas-Rachford splitting steps for (6)

$$\begin{cases} \mathbf{y}^{k+1} & := & \mathbf{x}^k + \alpha(\bar{\mathbf{x}}^k - \mathbf{x}^k), \\ \mathbf{x}^{k+1} & := & \mathrm{prox}_{n\eta f}(\mathbf{y}^{k+1}), \\ \bar{\mathbf{x}}^{k+1} & := & \mathrm{prox}_{n\eta(g+\delta_{\mathcal{L}})}(2\mathbf{x}^{k+1} - \mathbf{y}^{k+1}), \end{cases} \tag{7}$$

where $\eta > 0$ is a given **step-size** and $\alpha \in (0, 2]$ is a **relaxation parameter**.

Application of DR to FL

- **Step 1:** Decompose $\mathbf{x}^{k+1} := \mathrm{prox}_{n\eta f}(\mathbf{y}^{k+1})$ into $x_i^{k+1} := \mathrm{prox}_{\eta f_i}(y_i^{k+1})$ for all $i \in [n]$.
- **Step 2:** Introduce $\hat{x}_i^{k+1} := 2x_i^{k+1} - y_i^{k+1}$ for all $i \in [n]$.
- **Step 3:** Line 3 of (7) $\bar{\mathbf{x}}^{k+1} := \mathrm{prox}_{n\eta(g+\delta_{\mathcal{L}})}(\hat{\mathbf{x}}^{k+1})$ can be rewritten as

$$\bar{\mathbf{x}}^{k+1} := \mathrm{prox}_{n\eta(g+\delta_{\mathcal{L}})}(\hat{\mathbf{x}}^{k+1}) = \begin{cases} \arg\min_{\mathbf{x}} \left\{ g(x_1) + \frac{1}{2n\eta}\|\mathbf{x}_i - \hat{\mathbf{x}}_i^{k+1}\|^2 \right\} \\ \text{s.t.} \quad x_i = x_1, \quad \text{for all } i = 2, \cdots, n. \end{cases}$$

Explicitly solve this problem to get a **closed-form update** for $\bar{\mathbf{x}}^{k+1}$.

Parallel DR Splitting Scheme for Solving (1)

From a full parallel to randomized block-coordinate DR splitting scheme

The full parallel DR splitting scheme:

$$\begin{cases} y_i^{k+1} & := & y_i^k + \alpha(\bar{x}^k - x_i^k), \quad \forall i \in [n] \\ x_i^{k+1} & := & \mathrm{prox}_{\eta f_i}(y_i^{k+1}), \quad \forall i \in [n] \\ \hat{x}_i^{k+1} & := & 2x_i^{k+1} - y_i^{k+1}, \quad \forall i \in [n] \\ \tilde{x}^{k+1} & := & \frac{1}{n}\sum_{i=1}^n \hat{x}_i^{k+1}, \\ \bar{x}^{k+1} & := & \mathrm{prox}_{\eta g}\left(\tilde{x}^{k+1}\right). \end{cases} \tag{8}$$

The randomized block-coordinate DR splitting scheme:

- **Randomly sample** a subset $\mathcal{S}_k$ of users in $\{1, 2, \cdots, n\}$.
- Update y_i^{k+1}, x_i^{k+1}, and $\hat{x}_i^{k+1}$ for $i \in \mathcal{S}_k$, while keeping other users unchanged.

Parallel DR Splitting Scheme for Solving (1)

From a full parallel to randomized block-coordinate DR splitting scheme

The full parallel DR splitting scheme:

$$\begin{cases} y_i^{k+1} & := & y_i^k + \alpha(\bar{x}^k - x_i^k), \quad \forall i \in [n] \\ x_i^{k+1} & := & \mathrm{prox}_{\eta f_i}(y_i^{k+1}), \quad \forall i \in [n] \\ \hat{x}_i^{k+1} & := & 2x_i^{k+1} - y_i^{k+1}, \quad \forall i \in [n] \\ \tilde{x}^{k+1} & := & \frac{1}{n}\sum_{i=1}^n \hat{x}_i^{k+1}, \\ \bar{x}^{k+1} & := & \mathrm{prox}_{\eta g}\left(\tilde{x}^{k+1}\right). \end{cases} \tag{8}$$

The randomized block-coordinate DR splitting scheme:

- **Randomly sample** a subset $\mathcal{S}_k$ of users in $\{1, 2, \cdots, n\}$.
- Update y_i^{k+1}, x_i^{k+1}, and $\hat{x}_i^{k+1}$ for $i \in \mathcal{S}_k$, while keeping other users unchanged.

Challenges in analysis of our block-coordinate DR variant

- Three randomized block-coordinate steps y_i^{k+1}, x_i^{k+1}, and $\hat{x}_i^{k+1}$ are updated sequentially.
- Cannot switch $\mathrm{prox}_{\eta f_i}$ and $\mathrm{prox}_{\eta g}$ as in the convex case.

Classical Block-Coordinate Fixed-point Scheme

Classical block-coordinate fixed-point scheme

Given $\mathbf{y}^0$, iterate

$$\mathbf{y}_i^{k+1} = \begin{cases} \mathcal{T}_i(\mathbf{y}^k) & \text{if } i = i_k \\ \mathbf{y}_i^k & \text{otherwise} \end{cases} \tag{9}$$

where $\mathcal{T}$ is a **fixed-point mapping**, and i_k is "randomly sampled" from $[n]$ blocks.

Example: **ARock** from [Peng et al (2016)] is an instance of this general scheme.

We find that most existing block-coordinate-based methods rely on this principle.

Classical Block-Coordinate Fixed-point Scheme

Classical block-coordinate fixed-point scheme

Given $\mathbf{y}^0$, iterate

$$\mathbf{y}_i^{k+1} = \begin{cases} \mathcal{T}_i(\mathbf{y}^k) & \text{if } i = i_k \\ \mathbf{y}_i^k & \text{otherwise} \end{cases} \tag{9}$$

where $\mathcal{T}$ is a **fixed-point mapping**, and i_k is "randomly sampled" from $[n]$ blocks.

Example: **ARock** from [Peng et al (2016)] is an instance of this general scheme.

We find that most existing block-coordinate-based methods rely on this principle.

The Douglas-Rachford fixed-point interpretation

From (7), define the following Douglas-Rachford mapping:

$$\mathcal{T}_{\mathrm{DR}}(\mathbf{y}) = \mathrm{prox}_{n\eta f}(\mathbf{y}) + \alpha \cdot \left(\mathrm{prox}_{n\eta(g+\delta_{\mathcal{L}})}\left(2 \cdot \mathrm{prox}_{n\eta f}(\mathbf{y}) - \mathbf{y}\right)\right). \tag{10}$$

Then, (7) can be written as

$$\mathbf{y}^{k+1} = \mathcal{T}_{\mathrm{DR}}(\mathbf{y}^k).$$

Finally, $\bar{\mathbf{x}}^{k+1} = \mathrm{prox}_{n\eta(g+\delta_{\mathcal{L}})}\left(2 \cdot \mathrm{prox}_{n\eta f}(\mathbf{y}^k) - \mathbf{y}^k\right)$ as the output.

- If we apply the **block-coordinate scheme** (9) to (16), then we need to compute full blocks of $\mathrm{prox}_{\eta f_i}$ for $i \in [n]$.
- The output is $\bar{\mathbf{x}}^k$ in our analysis, not $\mathbf{y}^k$ as in the scheme (9).

The Complete FedDR Algorithm

Algorithm 1 (Federated Learning with Randomized DR (**FedDR**))

Initialization:

Take $x^0 \in \mathrm{dom} F$. Choose $\eta > 0$ and $\alpha > 0$, and accuracies $\epsilon_{i,0} \geq 0$ $(i \in [n])$.

Initialize the server with $\bar{x}^0 := x^0$ and $\tilde{x}^0 := x^0$.

Initialize all users with $y_i^0 := x^0$, $x_i^0 :\approx \mathrm{prox}_{\eta f_i}(y_i^0)$, and $\hat{x}_i^0 := 2x_i^0 - y_i^0$.

For $k := 0, \cdots, K$ **do**

[*Active users*] Generate a proper realization $\mathcal{S}_k \subseteq [n]$ of $\hat{\mathcal{S}}$.

[*Communication*] Each user $i \in \mathcal{S}_k$ **receives** $\bar{x}^k$ from the server.

[*Local update*] **For each user** $i \in \mathcal{S}_k$ **do**: Choose $\epsilon_{i,k+1} \geq 0$ and update

$$\begin{cases} y_i^{k+1} & := & y_i^k + \alpha(\bar{x}^k - x_i^k), \\ x_i^{k+1} & :\approx & \mathrm{prox}_{\eta f_i}(y_i^{k+1}), \\ \hat{x}_i^{k+1} & := & 2x_i^{k+1} - y_i^{k+1}. \end{cases}$$

[*Communication*] Each user $i \in \mathcal{S}_k$ **sends** $\Delta\hat{x}_i^k := \hat{x}_i^{k+1} - \hat{x}_i^k$ back to the server.

[*Sever aggregation*] Aggregate $\tilde{x}^{k+1} := \tilde{x}^k + \frac{1}{n}\sum_{i \in \mathcal{S}_k} \Delta\hat{x}_i^k$.

[*Sever update*] Update $\bar{x}^{k+1} := \mathrm{prox}_{\eta g}\left(\tilde{x}^{k+1}\right)$.

End For

Sampling Scheme and Technical Assumption

Sample scheme for users

- Consider a proper sampling scheme $\hat{\mathcal{S}}$ of $[n]$, which is a random set-valued mapping with values in $2^{[n]}$, the collection of all subsets of $[n]$.
- Let $\mathcal{S}_k$ be an iid realization of $\hat{\mathcal{S}}$ and $\mathcal{F}_k := \sigma(\mathcal{S}_0, \cdots, \mathcal{S}_k)$ be the σ-algebra generated by $\mathcal{S}_0, \cdots, \mathcal{S}_k$.

Sampling Scheme and Technical Assumption

Sample scheme for users

- Consider a proper sampling scheme $\hat{\mathcal{S}}$ of $[n]$, which is a random set-valued mapping with values in $2^{[n]}$, the collection of all subsets of $[n]$.
- Let $\mathcal{S}_k$ be an iid realization of $\hat{\mathcal{S}}$ and $\mathcal{F}_k := \sigma(\mathcal{S}_0, \cdots, \mathcal{S}_k)$ be the σ-algebra generated by $\mathcal{S}_0, \cdots, \mathcal{S}_k$.

Assumption 2

There exist $\mathbf{p}_1, \cdots, \mathbf{p}_n > 0$ *such that*

$$\mathbb{P}\left(i \in \hat{\mathcal{S}}\right) = \mathbf{p}_i > 0$$

for all $i \in [n]$.

Main Result 1: Convergence and Communication Complexity

Theorem 2 (Convergence of **FedDR**)

Suppose that:

- ***Assumptions 1 and 2*** *hold.*
- *Let* $\{(x_i^k, y_i^k, \hat{x}_i^k, \bar{x}^k)\}$ *be generated by the exact variant of* ***Algorithm 1***
- *The conditions* $0 < \alpha < 2$ *and* $0 < \eta < \frac{2-\alpha}{2L}$ *hold.*

Main Result 1: Convergence and Communication Complexity

Theorem 2 (Convergence of **FedDR**)

Suppose that:

- ***Assumptions 1 and 2*** *hold.*
- *Let* $\{(x_i^k, y_i^k, \hat{x}_i^k, \bar{x}^k)\}$ *be generated by the exact variant of* ***Algorithm 1***
- *The conditions* $0 < \alpha < 2$ *and* $0 < \eta < \frac{2-\alpha}{2L}$ *hold.*

Conclusions:

- *We have*

$$\frac{1}{K+1}\sum_{k=0}^{K}\mathbb{E}\left[\|\mathcal{G}_\eta(\bar{x}^k)\|^2\right] \leq \frac{C[F(x^0) - F^\star]}{K+1}, \tag{11}$$

  where $C := \frac{4(1+\eta L)^2(1+L^2\eta^2)}{\hat{\mathbf{p}}\eta\alpha(2-\alpha(L\eta+1)-2L^2\eta^2)} > 0$.

- *Let* $\tilde{x}^K$ *be selected uniformly at random from* $\{\bar{x}^0, \cdots, \bar{x}^K\}$ *as the* ***output*** *of* ***Algorithm 1****. Then, after at most*

$$K := \left\lfloor \frac{C[F(x^0) - F^\star]}{\varepsilon^2} \right\rfloor \equiv \mathcal{O}\left(\frac{1}{\varepsilon^2}\right)$$

  iterations, we obtain $\tilde{x}^K$ *as an* ε***-stationary point*** *of* (1) *as in* ***Definition 1****.*

Outline

Motivation of assyncFedDR

Motivation

- In FL, it is critical to account for **system heterogeneity** of local users.
- Requiring **synchronous aggregation** at the end of each communication round may lead to slow down in training.
- It is more practical to have **asynchronous update** from local users.

Motivation of assyncFedDR

Motivation

- In FL, it is critical to account for **system heterogeneity** of local users.
- Requiring **synchronous aggregation** at the end of each communication round may lead to slow down in training.
- It is more practical to have **asynchronous update** from local users.

Main idea of **asyncFedDR**

- At each iteration k, each user receives a **delay copy** $\bar{x}^{k-d_{i_k}^k}$ of $\bar{x}^k$ from the server with a **delay** $d_{i_k}^k$.
- The **active user** i_k will update its own **local model** $(y_i^k, x_i^k, \hat{x}_i^k)$ in an asynchronous mode without waiting for others to complete.
- Once completing its update, user i_k just sends an **increment** $\Delta\hat{x}_{i_k}^k$ to the server to update the **global model**, while others may be still reading.

In our analysis, a **transition of iteration** from k to $k+1$ is triggered whenever a user completes its update.

The Asynchronous FedDR – asyncFedDR

Algorithm 2 (Asynchronous FedDR (**asyncFedDR**))

Initialization:

Take $x^0 \in \mathrm{dom}F$ and choose $\eta > 0$ and $\alpha > 0$.

Initialize the server with $\bar{x}^0 := x^0$ and $\tilde{x}^0 := 0$.

Initialize each user $i \in [n]$ with $y_i^0 := x^0$, $x_i^0 := \mathrm{prox}_{\eta f_i}(y_i^0)$, and $\hat{x}_i^0 := 2x_i^0 - y_i^0$.

For $k := 0, \cdots, K$ **do**

Select i_k such that (i_k, d^k) is a **realization** of $(\hat{i}_k, \hat{d}^k)$.

[*Communication*] User i_k **receives** $\bar{x}^{k-d_{i_k}^k}$, a **delayed version** of $\bar{x}^k$ with delay $d_{i_k}^k$.

[*Local update*] User i_k updates

$$\begin{cases} y_{i_k}^{k+1} & := & y_{i_k}^k + \alpha(\bar{x}^{k-d_{i_k}^k} - x_{i_k}^k), \\ x_{i_k}^{k+1} & := & \mathrm{prox}_{\eta f_{i_k}}(y_{i_k}^{k+1}), \\ \hat{x}_{i_k}^{k+1} & := & 2x_{i_k}^{k+1} - y_{i_k}^{k+1}. \end{cases}$$

Other users **maintain** $y_i^{k+1} := y_i^k$, $x_i^{k+1} := x_i^k$, and $\hat{x}_i^{k+1} := \hat{x}_i^k$ for $i \neq i_k$.

[*Communication*] User i_k **sends** $\Delta_{i_k}^k := \hat{x}_{i_k}^{k+1} - \hat{x}_{i_k}^k$ back to the server.

[*Sever aggregation*] Aggregate $\tilde{x}^{k+1} := \tilde{x}^k + \frac{1}{n}\Delta_{i_k}^k$.

[*Sever update*] Update $\bar{x}^{k+1} := \mathrm{prox}_{\eta g}\left(\tilde{x}^{k+1}\right)$.

End For

Joint Probabilistic Model for Users and Delays

Probabilistic model

- Introduce $\hat{\xi}^k := (\hat{i}_k, \hat{d}^k)$ as a **joint random variable** containing the **user index** $\hat{i}_k \in [n]$ and the **delay vector** $\hat{d}^k \in \mathcal{D} := \{0, 1, \cdots, \tau\}^n$ at iteration k.
- Let $\xi^k := (i_k, d^k)$ be a realization of a random vector.
- Introduce a **random vector** $\hat{\xi}^{0:k} := (\hat{\xi}^0, \cdots, \hat{\xi}^k)$ and its possible values $\xi^{0:k} = (\xi^0, \xi^1, \cdots, \xi^k)$.
- Let Ω be the **sample space** of all **sequences** $\omega := \{(i_k, d^k)\}_{k \geq 0}$.
- Assume that $\mathbf{p}(\xi^{0:k}) := \mathbb{P}(\hat{\xi}^{0:k} = \xi^{0:k}) > 0$.

Joint Probabilistic Model for Users and Delays

Probabilistic model

- Introduce $\hat{\xi}^k := (\hat{i}_k, \hat{d}^k)$ as a **joint random variable** containing the **user index** $\hat{i}_k \in [n]$ and the **delay vector** $\hat{d}^k \in \mathcal{D} := \{0, 1, \cdots, \tau\}^n$ at iteration k.
- Let $\xi^k := (i_k, d^k)$ be a realization of a random vector.
- Introduce a **random vector** $\hat{\xi}^{0:k} := (\hat{\xi}^0, \cdots, \hat{\xi}^k)$ and its possible values $\xi^{0:k} = (\xi^0, \xi^1, \cdots, \xi^k)$.
- Let Ω be the **sample space** of all **sequences** $\omega := \{(i_k, d^k)\}_{k \geq 0}$.
- Assume that $\mathbf{p}(\xi^{0:k}) := \mathbb{P}(\hat{\xi}^{0:k} = \xi^{0:k}) > 0$.

Assumption 3 (Positive probability for updates and bounded delay)

- *For all $i \in [n]$ and $\omega \in \Omega$, $\exists$ at least one $t \in \{0, 1, \cdots, T\}$ with $T > 0$, such that*

$$\sum_{d \in \mathcal{D}} \mathbf{p}((i, d) \mid \xi^{0:k+t-1}) \geq \hat{\mathbf{p}} \quad \textit{if } \mathbf{p}(\xi^{0:k}) > 0, \tag{12}$$

for a given $\hat{\mathbf{p}} > 0$ and any $k \geq 0$.
- *Assume also that $d_i^k \leq \tau$ and $d_{i_k}^k = 0$ for all $k \geq 0$ and $i, i_k \in [n]$.*

Technical Parameters for Complexity Bound

- **Step 1:** Choose $0 < \alpha < \bar{\alpha}$ and $0 < \eta < \bar{\eta}$ in **Algorithm 2**, where $c := \frac{2\tau^2 - n}{n^2}$ is given, and $\bar{\alpha} > 0$ and $\bar{\eta} > 0$ are respectively computed by

$$\bar{\alpha} := \begin{cases} 1 & \text{if } 2\tau^2 \le n, \\ \frac{2}{2+c} & \text{otherwise,} \end{cases} \quad \text{and } \bar{\eta} := \begin{cases} \frac{\sqrt{16 - 8\alpha - 7\alpha^2} - \alpha}{2L(2+\alpha)} & \text{if } 2\tau^2 \le n, \\ \frac{\sqrt{16 - 8\alpha - (7 + 4c + 4c^2)\alpha^2} - \alpha}{2L[2 + (1+c)\alpha]} & \text{otherwise.} \end{cases}$$

- **Step 2:** Introduce the following parameters:

$$\rho := \begin{cases} \frac{2(1-\alpha) - (2+\alpha)L^2\eta^2 - L\alpha\eta}{\alpha\eta n} & \text{if } \quad 2\tau^2 \le n, \\ \frac{n^2[2(1-\alpha) - (2+\alpha)L^2\eta^2 - L\alpha\eta] - \alpha(1+\eta^2L^2)(2\tau^2 - n)}{\alpha\eta n^3} & \text{otherwise.} \end{cases}$$

$$D := \frac{8\alpha^2(1 + L^2\eta^2)(\tau^2 + 2Tn\hat{\mathbf{p}}) \ + \ 8n^2(1 + L^2\eta^2 + T\alpha^2\hat{\mathbf{p}})}{\hat{\mathbf{p}}\alpha^2 n^2}.$$

Both ρ and D are positive.

Remark: When the delay τ satisfies $\tau \le \sqrt{\frac{n}{2}}$, we can use large stepsizes α and η. Otherwise, we need to choose smaller stepsizes α and η.

Main Result 2: Convergence and Communication Complexity

Theorem 3 (Convergence of **asyncFedDR**)

Suppose that:

- ***Assumption 1 and 3*** *hold.*
- *Let* $\bar{\alpha}$, $\bar{\eta}$, ρ, *and* D *be given in* ***the previous slide****, respectively.*
- *Let* $\{(x_i^k, y_i^k, \bar{x}^k)\}$ *be generated by* ***Algorithm 2****.*
- *The conditions* $\alpha \in (0, \bar{\alpha})$ *and* $\eta \in (0, \bar{\eta})$ *hold.*

Main Result 2: Convergence and Communication Complexity

Theorem 3 (Convergence of **asyncFedDR**)

Suppose that:

- ***Assumption 1 and 3*** *hold.*
- *Let $\bar{\alpha}$, $\bar{\eta}$, ρ, and D be given in* ***the previous slide****, respectively.*
- *Let $\{(x_i^k, y_i^k, \bar{x}^k)\}$ be generated by* ***Algorithm 2****.*
- *The conditions $\alpha \in (0, \bar{\alpha})$ and $\eta \in (0, \bar{\eta})$ hold.*

Conclusions:

- *Then, the following bound holds:*

$$\frac{1}{K+1}\sum_{k=0}^{K}\mathbb{E}\left[\|\mathcal{G}_{\eta}(\bar{x}^k)\|^2\right] \leq \frac{\hat{C}\left[F(x^0) - F^{\star}\right]}{K+1}, \tag{13}$$

  where $\hat{C} := \frac{2(1+\eta L)^2 D}{n\eta^2\rho} > 0$ depending on $n, L, \eta, \alpha, \tau, T$, and $\hat{\mathbf{p}}$.

- *Let $\tilde{x}_K$ be selected uniformly at random from $\{\bar{x}^0, \cdots, \bar{x}^K\}$ as the* ***output*** *of* ***Algorithm 2****. Then, after at most $K := \mathcal{O}\left(\varepsilon^{-2}\right)$ iterations, $\tilde{x}^K$ is an ε-**stationary point** of (1) as in* ***Definition 1****.*

Outline

Problem Statement, Motivation, and Contribution

Federated Learning with Randomized DR – FedDR

Federated Learning with Asynchronous DR – assyncFedDR

Numerical Examples

Conclusions and Future Research

Configuration of Experiments

Experiment Configuration

- **Our methods: FedDR** and **asyncFedDR**
- **Competitors: FedAvg**, **FedProx**, and **FedPD**.
- **Optimization Models:** Neural networks.
- **Data:** Both synthetic and real datasets.
- **Comparison Metrics:** Training loss, training accuracy, and test accuracy.
- **Parameters:** Parameters are tuned to obtain the best performance in all methods.
- **Local Solvers:** Use the same local solver (SGD) for all algorithms.

Configuration of Experiments

Experiment Configuration

- **Our methods: FedDR** and **asyncFedDR**
- **Competitors: FedAvg**, **FedProx**, and **FedPD**.
- **Optimization Models:** Neural networks.
- **Data:** Both synthetic and real datasets.
- **Comparison Metrics:** Training loss, training accuracy, and test accuracy.
- **Parameters:** Parameters are tuned to obtain the best performance in all methods.
- **Local Solvers:** Use the same local solver (SGD) for all algorithms.

Implementation

- For synchronous algorithms, we reuse the implementation of **FedAvg** and **FedProx** in [Li et al (2020)] and implement **FedDR** and **FedPD** on top of it.
- For asynchronous methods, we implement **our algorithms** based on the asynchronous framework in **DistBelief** [Cai (2018)].
- All experiments are run on a **Linux-based server** with multiple nodes and configuration: 24-core 2.50GHz Intel processors, 30M cache, and 256GB RAM.

Performance of FedDR and Competitors on Synthetic Datasets

- Compare the algorithms on **synthetic dataset** with both **iid** and **non-iid** settings.
- Generate 1 iid dataset `synthetic-iid` and 3 non-iid datasets: `synthetic-`(r,s) for $(r,s) = \{(0,0),(0.5,0.5),(1,1)\}$ as in [Li et al (2020)].
- Update all users without sampling and non-composite model of (1).

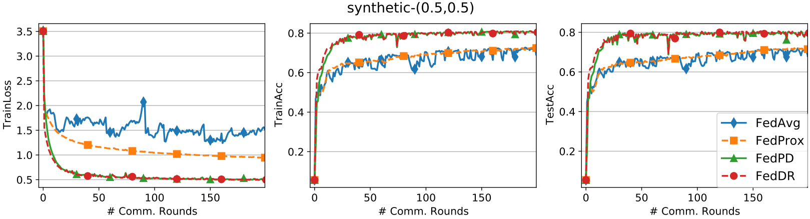


Figure: The performance of 4 algorithms on non-iid synthetic datasets without user sampling

Performance of FedDR and Competitors on Synthetic Datasets

- Compare the algorithms on **synthetic dataset** with both **iid** and **non-iid** settings.
- Generate 1 iid dataset `synthetic-iid` and 3 non-iid datasets: `synthetic-`(r,s) for $(r,s) = \{(0,0),(0.5,0.5),(1,1)\}$ as in [Li et al (2020)].
- Update all users without sampling and non-composite model of (1).

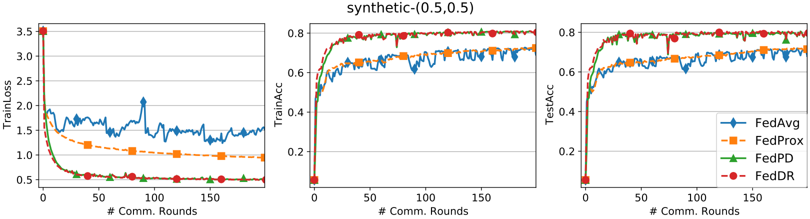


Figure: The performance of 4 algorithms on non-iid synthetic datasets without user sampling

Observation

- **FedDR** and **FedPD** are comparable in these datasets.
- They both outperform **FedProx** and **FedAvg**.
- **FedProx** works better than **FedAvg** which was observed before.
- Comparing on more datasets, **our algorithm** overall performs better than others.

Performance of FedDR and Competitors on Synthetic Datasets

- Sample of 10 users out of 30 to update at each communication round for **FedAvg**, **FedProx**, and **FedDR**.
- Use all users for **FedPD**.
- The evaluation metric is the **number of bytes** communicated between users and server at each communication round.

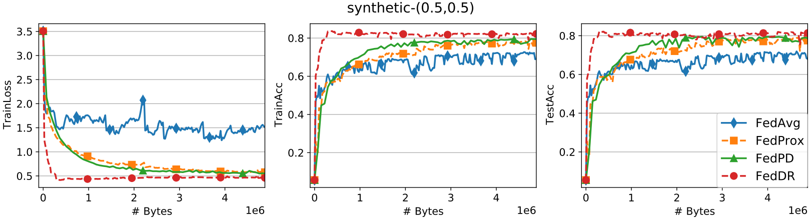


Figure: The performance of 4 algorithms with user sampling scheme on non-iid synthetic datasets.

Performance of FedDR and Competitors on Synthetic Datasets

- Sample of 10 users out of 30 to update at each communication round for **FedAvg**, **FedProx**, and **FedDR**.
- Use all users for **FedPD**.
- The evaluation metric is the **number of bytes** communicated between users and server at each communication round.

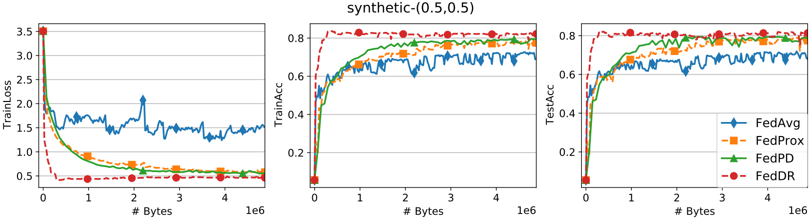


Figure: The performance of 4 algorithms with user sampling scheme on non-iid synthetic datasets.

Observation

- **FedDR** performs well compared to others.
- **FedProx** using user sampling scheme performs better and is slightly behind **FedPD**.
- **FedDR**, **FedPD**, and **FedProx** outperform **FedAvg**.

Performance of FedDR and Competitors on FEMNIST Datasets

- **FEMNIST** is an extended version of **MNIST**.
- It has a total of 62 classes (10 digits, 26 upper-case and 26 lower-case letters) with over 800,000 samples.
- There are total of 200 users and we sample 50 users to update **FedAvg**, **FedProx**, and **FedDR**, while we use all users to perform update for **FedPD**.

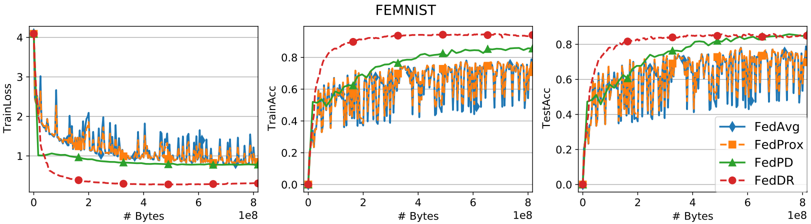


Figure: The performance of 4 algorithms on the FEMNIST dataset.

Performance of FedDR and Competitors on FEMNIST Datasets

- **FEMNIST** is an extended version of **MNIST**.
- It has a total of 62 classes (10 digits, 26 upper-case and 26 lower-case letters) with over 800,000 samples.
- There are total of 200 users and we sample 50 users to update **FedAvg**, **FedProx**, and **FedDR**, while we use all users to perform update for **FedPD**.

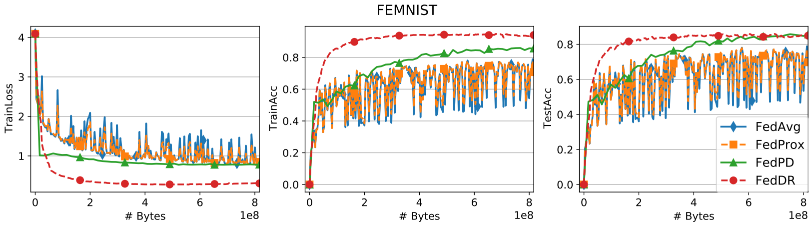


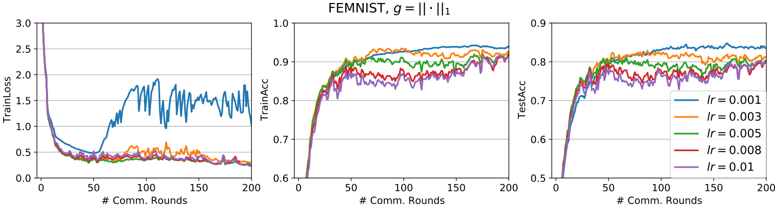
Figure: The performance of 4 algorithms on the FEMNIST dataset.

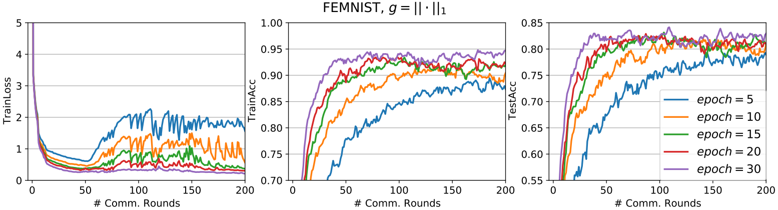
Observation

- **FedDR** can achieve lower loss and higher training accuracy than **other algorithms**.
- **FedPD** can reach the same test accuracy as ours at the end.
- Overall, **FedDR** seems working better than other algorithms in this test.

The Composite Case with ℓ_1-Norm Regularizer

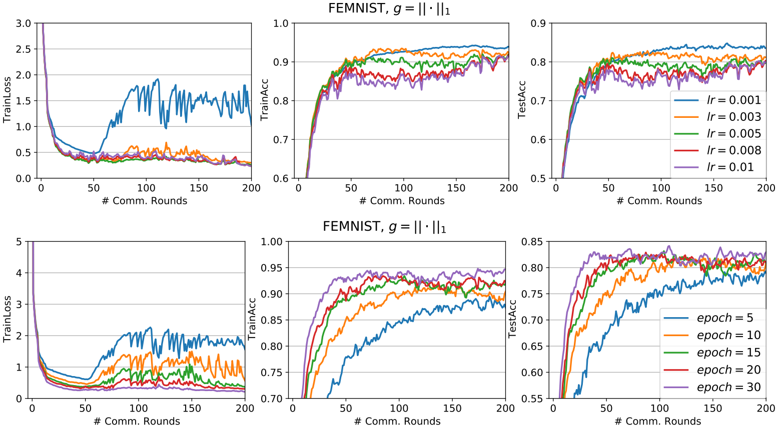
- Choose $g(x) := 0.01\,\|x\|_1$ and **different inexactness levels** $\epsilon_{i,k}$.
- Run **Algorithm 1** on the **FEMNIST** dataset.





The Composite Case with ℓ_1-Norm Regularizer

- Choose $g(x) := 0.01 \|x\|_1$ and **different inexactness levels** $\epsilon_{i,k}$.
- Run **Algorithm 1** on the **FEMNIST** dataset.



Observation

- **Algorithm 1** works best when **local learning rate** is 0.003.
- It also performs better when we decrease $\epsilon_{i,k}$ by increasing the **number of epochs**.

Performance of asyncFedDR over FedDR

- Illustrate the advantages of **asyncFedDR** over **FedDR**.
- Use **MNIST** dataset with a sample of 20 users per round.
- Since the computing nodes have identical configurations, we add **variable delay** to users to simulate a **computing power discrepancy**.

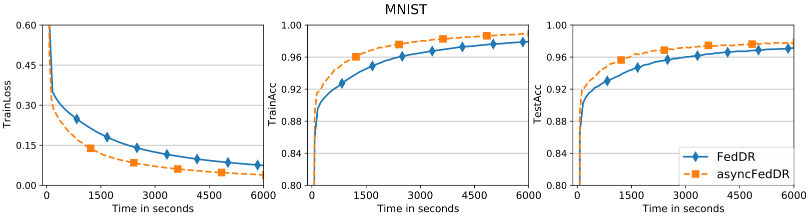


Figure: The performance of **FedDR** and **asyncFedDR** on the MNIST dataset.

Performance of asyncFedDR over FedDR

- Illustrate the advantages of **asyncFedDR** over **FedDR**.
- Use **MNIST** dataset with a sample of 20 users per round.
- Since the computing nodes have identical configurations, we add **variable delay** to users to simulate a **computing power discrepancy**.

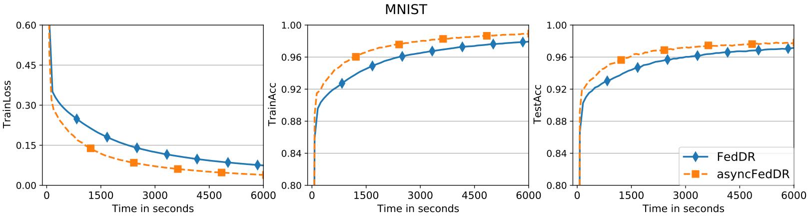


Figure: The performance of **FedDR** and **asyncFedDR** on the MNIST dataset.

Observation

- **asyncFedDR** can achieve better performance than **FedDR** in terms of training time.
- This illustrate the advantage of asynchronous update in **heterogeneous systems**.

Outline

Problem Statement, Motivation, and Contribution

Federated Learning with Randomized DR – FedDR

Federated Learning with Asynchronous DR – assyncFedDR

Numerical Examples

Conclusions and Future Research

Conclusions and Future Research

Conclusions

- Develop **two new algorithms** for FL using **randomized DR splitting idea**.
- The **new algorithms** have several advantages: subset of users per round, asynchronous implementation, inexact computation, composite form, etc.
- Prove the best-known complexity for communication under **standard assumptions**.
- **Numerical experiments** overall show the advantages over their **competitors**.

Conclusions and Future Research

Conclusions

- Develop **two new algorithms** for FL using **randomized DR splitting idea**.
- The **new algorithms** have several advantages: subset of users per round, asynchronous implementation, inexact computation, composite form, etc.
- Prove the best-known complexity for communication under **standard assumptions**.
- **Numerical experiments** overall show the advantages over their **competitors**.

Future research directions

- Focus on the **convex setting** and apply **compression** to improve communication.
- Study **accelerated methods** and **adaptive variants**.
- Incorporate **second-order information** to develop **second-order methods**.

Conclusions and Future Research

Conclusions

- Develop **two new algorithms** for FL using **randomized DR splitting idea**.
- The **new algorithms** have several advantages: subset of users per round, asynchronous implementation, inexact computation, composite form, etc.
- Prove the best-known complexity for communication under **standard assumptions**.
- **Numerical experiments** overall show the advantages over their **competitors**.

Future research directions

- Focus on the **convex setting** and apply **compression** to improve communication.
- Study **accelerated methods** and **adaptive variants**.
- Incorporate **second-order information** to develop **second-order methods**.

Thank you very much for your attention!