Federated Learning Algorithms

From easy to more complicated

Diego Klabjan

Northwestern | ENGINEERING

Acknowledgements

- PhD students of mine
  - Tom Overman, ESAM
  - Siqiao Mu, ESAM
  - Haemin Park, IEMS
  - Jay Xu, CS (graduated, at Meta)

Agenda

1. Introduction, Motivation and Setting

   Horizontal, Vertical, Hybrid Federated Learning

2. Federated Average (FedAvg)

   Variants to cope with non-IID

3. Vertical Setting

4. Hybrid Setting

INTRODUCTION

WHY CAN'T WE JUST CENTRALIZE THE DATA?

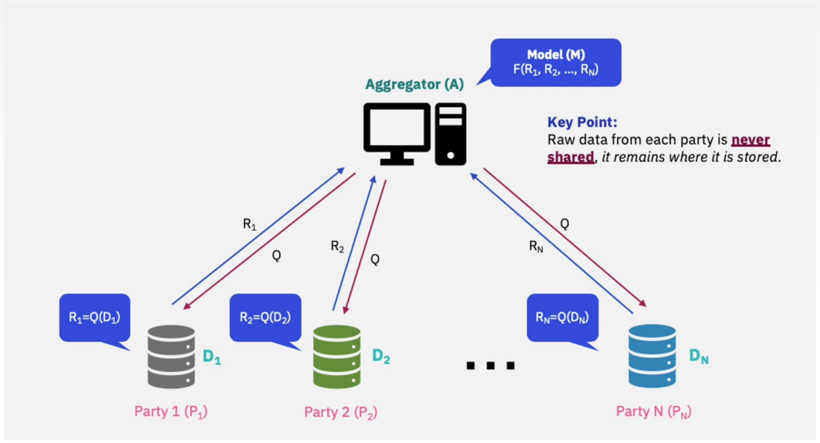
- Sending the data may be too costly
  - Self-driving cars are expected to generate several TBs of data a day
  - Some wireless devices have limited bandwidth/power
- Data may be considered too sensitive
  - We see a growing public awareness and regulations on data privacy
  - Keeping control of data can give a competitive advantage in business

HOW ABOUT EACH PARTY LEARNING ON ITS OWN?

- The local dataset may be too small
  - Sub-par predictive performance (e.g., due to overfitting)
  - Non-statistically significant results (e.g., medical studies)
- The local dataset may be biased
  - Not representative of the target distribution

Federated Learning

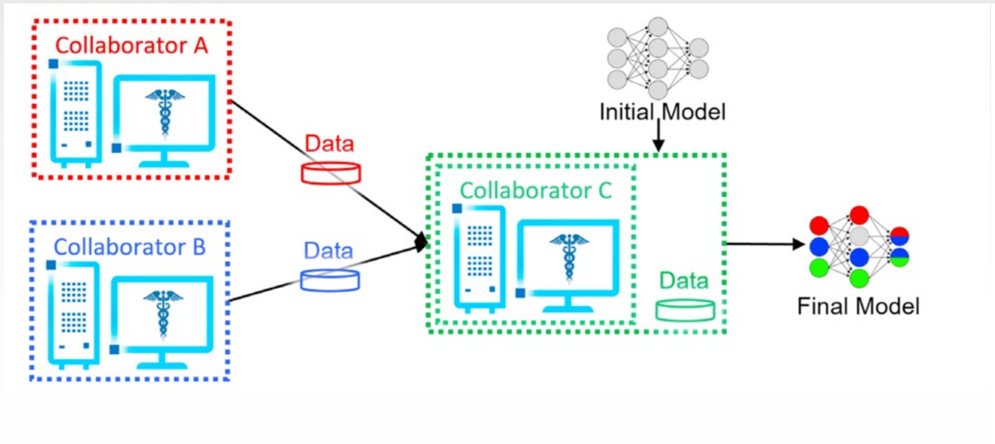
- Data is held on group of clients which can communicate with central server
- Build central model on server without ever accessing data from clients
  - Only send model weights
    - Other aspects that cannot be used to reconstruct data
- Goals of FL
  - Communication efficient
  - Robust to partial device participation
  - Resistant to model degradation with heterogeneous (non-IID) data

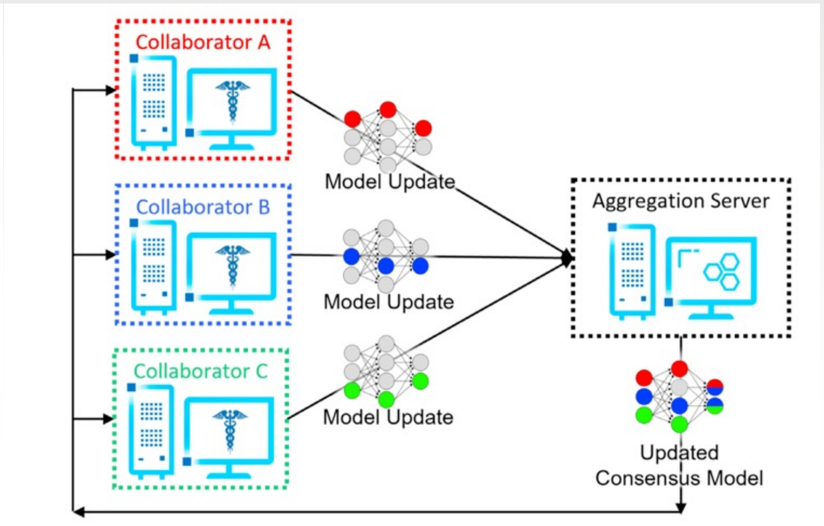


Why Federated Learning?

A consortium of hospitals	Shared patients
Telecommunication network providers	Stations do not share data Customers roam
Smart phones of a provider	User data not to be shared
Data from cars of drivers	Car manufacturers Insurance companies

Federated Learning





Centralized Learning

Centralized learning is a traditional machine learning approach where all data is collected and stored in a central repository or server.

Federated Learning

Learning technique that trains an algorithm via multiple independent sessions, each using its own dataset.

	Centralized Learning	Federated Learning
Data Storage	All training data is collected and stored in a central repository or server. Model training occurs on this central server using the entire dataset.	Data remains decentralized and is stored locally on edge devices or in various locations. Model training is performed locally on the edge devices without sharing raw data with a central server.
Data Privacy	The central server has access to the entire dataset, which can pose privacy and security concerns, especially when dealing with sensitive or private data.	Data remains on the local device and is not directly shared with a central server. Only model updates (gradients) are transmitted, and they are often anonymized and encrypted.
Scalability	Centralized learning may not be well-suited for edge computing environments, as it typically requires significant computational resources on the central server.	Federated learning is more suitable for edge computing, as it distributes the computation to the edge devices, reducing the central server's computational load.
Regulatory Compliance	Centralized learning may face more regulatory challenges related to data privacy, particularly in regions with strict data protection laws.	Federated learning can help organizations comply with data privacy regulations more effectively by keeping data local and minimizing data transmission.

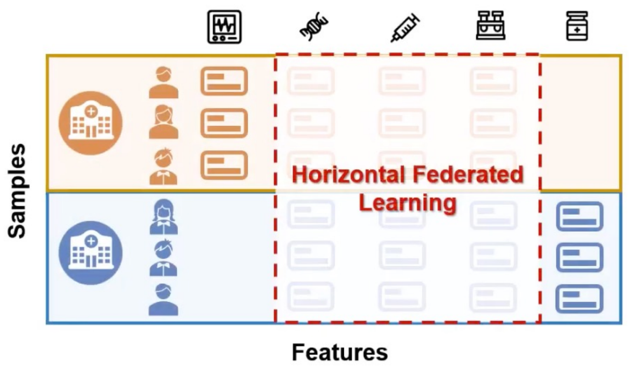
Horizontal & Vertical & Hybrid Federated Learning

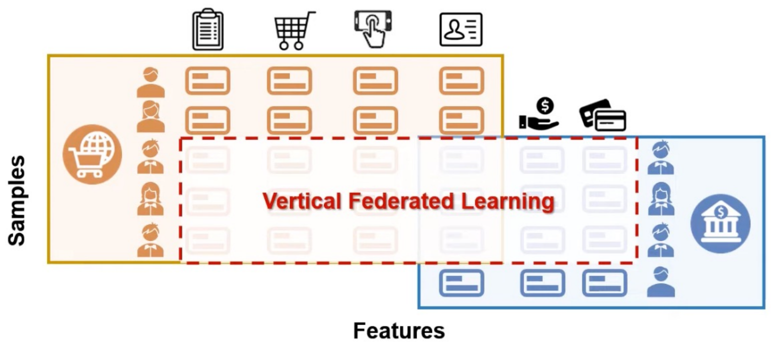
Dataset has samples and feature spaces

- **Sample** (Row) : single, individual data point or observation in a dataset
- **Feature** (Column) : characteristic or property of a data sample that is used to describe the data sample (also known as an attribute)

Horizontal Federated Learning : Intersecting features, different samples (Different hospitals have different patients but they may collect same information from their patients; Images of the same resolution collected at different places)

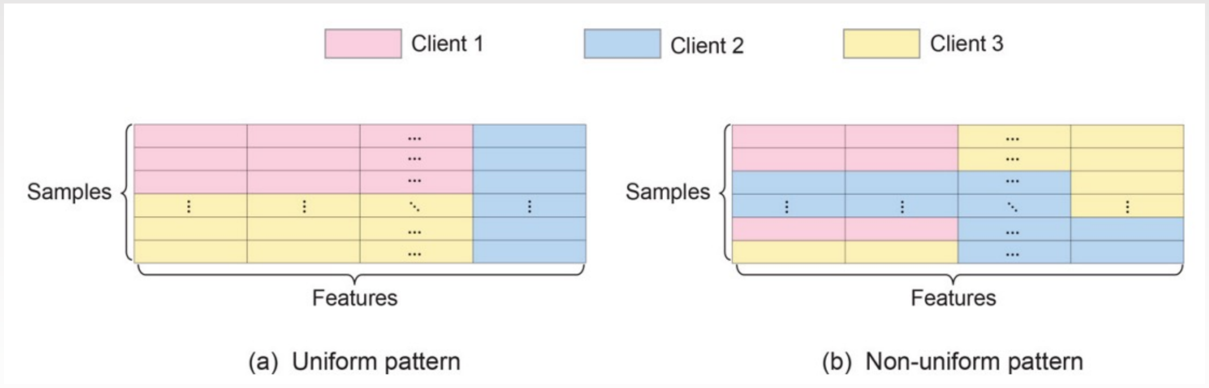
Vertical Federated Learning : Different features, intersecting samples (Shopping mall and bank may have same customers but they collect different information; Chips goes through different processes, each process captures its own features)





[9B] Comprehensive Analysis of Privacy Leakage in Vertical Federated Learning During Prediction (https://www.youtube.com/watch?v=WHZ_oKAUoXI)

Horizontal & Vertical & Hybrid Federated Learning



Haiqing Gao, Songyang Ge, Tsung-Hui Chang, FedHD: Communication-efficient federated learning from hybrid data, Journal of the Franklin Institute, nVolume 360, Issue 12, 2023,

Hybrid Federated Learning : A client may observe only part of the data samples with only part of the data features.

Example 1: Joint financial prediction(they want to predict the customers' purchasing capacities of financial products) of a customer by 1)**banks**, 2)**trust and invest cooperation**, and 3)**stock services**. The three entities respectively serve just a fraction of the customers and have only their partial records.

Example 2 : Different types of chips can go through partially same processes(same wafer fabrication, but different lithography…). Furthermore, there will be multiple rooms for same process, which also capture the same features.

Horizontal/vertical/hybrid?

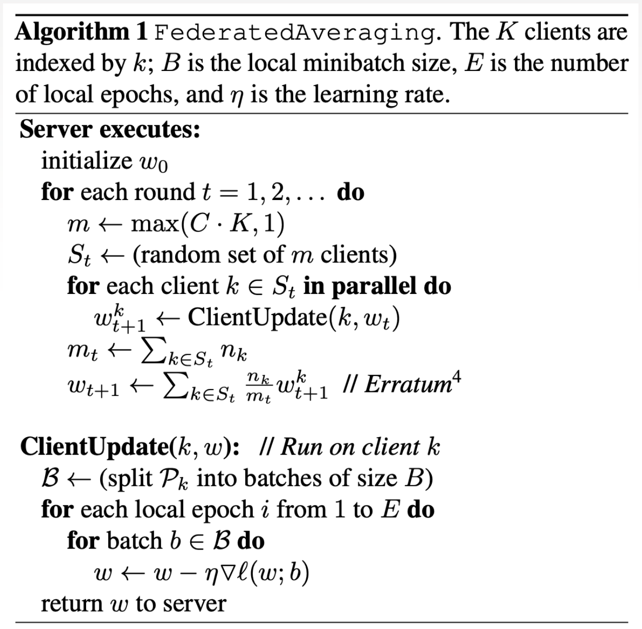
A consortium of hospitals	Shared patients
Telecommunication network providers	Stations do not share data Customers roam
Smart phones of a provider	User data not to be shared
Data from cars of drivers	Car manufacturers Insurance companies

ALGORITHMS FOR IID HORIZONTAL FL

Federated Learning Algorithm : FedAvg(Federated Averaging)

Northwestern | ENGINEERING

FedAvg(Federated Averaging) is a fundamental algorithm in federated learning that allows multiple devices or edge servers to collaboratively train a global machine learning model while keeping their data decentralized and private.



Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

```
Server executes:
  initialize $w_0$
  for each round $t = 1, 2, \dots$ do
    $m \leftarrow \max(C \cdot K, 1)$
    $S_t \leftarrow$ (random set of $m$ clients)
    for each client $k \in S_t$ in parallel do
      $w_{t+1}^k \leftarrow$ ClientUpdate$(k, w_t)$
    $m_t \leftarrow \sum_{k \in S_t} n_k$
    $w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$   // Erratum[4]

ClientUpdate$(k, w)$:   // Run on client $k$
  $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size $B$)
  for each local epoch $i$ from 1 to $E$ do
    for batch $b \in \mathcal{B}$ do
      $w \leftarrow w - \eta \nabla \ell(w; b)$
  return $w$ to server
```

Initialization: Initially, there is a global model that starts as an arbitrary model. Each participating device or edge server has its own local copy of this global model.

Local Model Training: Each device or edge server independently trains its local model using its own private data. This training occurs locally and does not involve sharing the raw data with a central server.

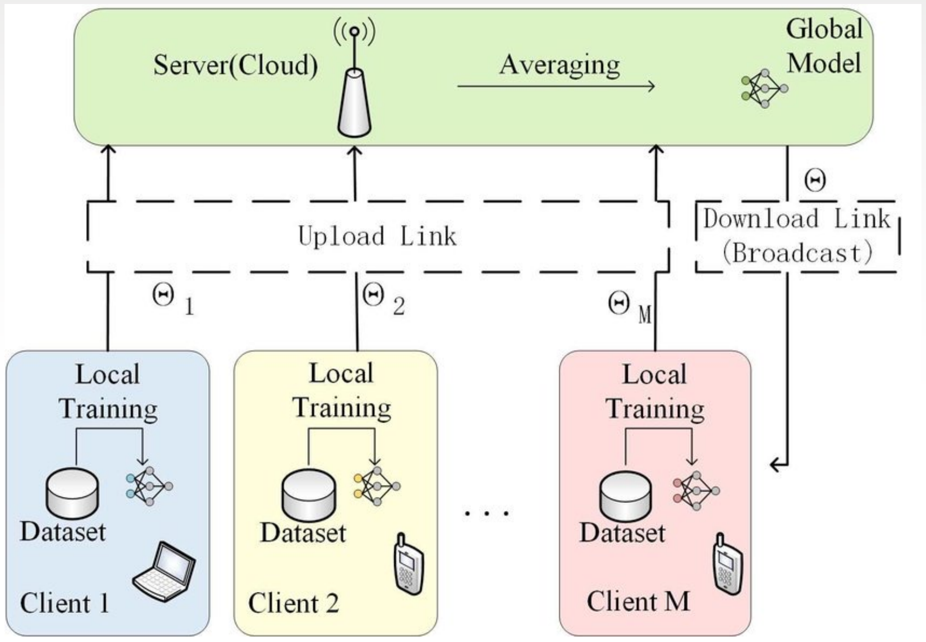
Model Update: After local training, each device sends the model to the server.

Aggregation: The models from all participating devices are securely transmitted to a central server. The central server aggregates these models by averaging them.

Global Model Update: The averaged model becomes the updated global model. The global model now incorporates the collective knowledge from all the devices without any individual device sharing its private data.

Haiqing Gao, Songyang Ge, Tsung-Hui Chang, FedHD: Communication-efficient federated learning from hybrid data, Journal of the Franklin Institute, nVolume 360, Issue 12, 2023,

Federated Learning Algorithm



Yang, Y., Zhang, Z., & Yang, Q. (2021). Communication-efficient federated learning with binary neural networks. *IEEE Journal on Selected Areas in Communications*, *39*(12), 3836-3850.

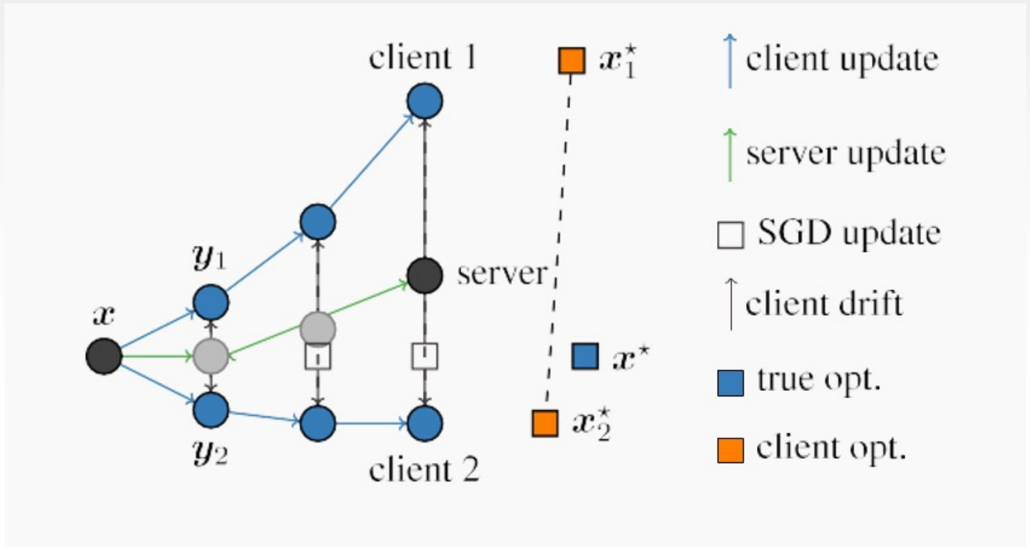
Local Model Training: Which local training algorithm to pick? How many rounds of local training do we force the local clients to do before the model aggregation?

Model Update: How to express the local updates? (Privacy Issue)

Aggregation: Which aggregation to pick?

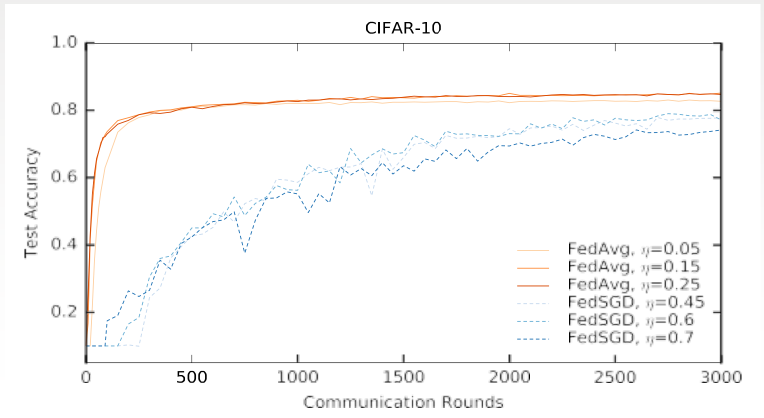
What setting of FL is FedAvg amenable to?

Non-IID



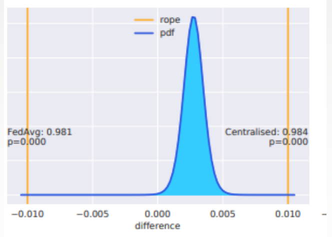
[Karimireddy et al., 2020])

FedAvg [MCMAHAN ET AL., 2017]

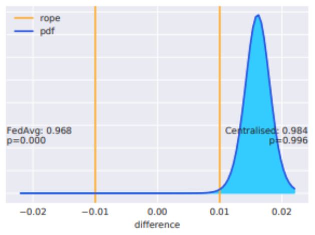


- FedAvg with $E > 1$ allows to reduce the number of communication rounds
  - Often the bottleneck in FL (especially in the cross-device setting)
- Empirically achieves better generalization than parallel SGD with large mini-batch
- Convergence to the optimal model can be guaranteed for i.i.d. data [Stich, 2019] [Woodworth et al., 2020]
  - Issues arise in strongly non-i.i.d. case

FedAvg vs Centralized



(d) FedAvg vs. Centr. i.i.d.



(j) FedAvg vs. Centr. non-i.i.d.

- Several variants of MNIST
- Posterior based on t-test
- p is area outside of rope
  - Probability the algorithm is better
- FedAvg good proxy in iid
  - Performance degrades in non-iid

A Performance Evaluation of Federated Learning Algorithms,
Adrian Nilsson, Simon Smith, Gregor Ulm, Emil Gustavsson, Mats Jirstrand

Setting

- IID vs non-IID
  - Distribution of data on clients – features and labels
- $w_{t+1} = \frac{1}{K} \sum_k w_t^k$
  - Poor performance in non-IID cases
- $w_{t+1} = \frac{1}{n} \sum_k n_k w_t^k$
  - n_k= number of samples at client k
  - Is it available?

Convergence

- Clients optimally solve
  - Works for non-convex problems
  - Convex case makes sense only if clients stop before convergence
    - Fix number of epochs
- General
  - Perform a fix number of epochs at clients
  - Convergence to optimality in the convex case under assumptions
    - Not trivial to establish
    - IID and non-IID cases

Convergence of FedAvg with Non-IID data

- Despite client drift, FedAvg is guaranteed to converge to the optimum for convex functions and the right choice of decreasing learning rate

$$\mathbb{E}\left[F(\mathbf{w}_T)\right] - F^* \le \frac{\kappa}{\gamma + T - 1}\left(\frac{2B}{\mu} + \frac{\mu\gamma}{2}\mathbb{E}\|\mathbf{w}_1 - \mathbf{w}^*\|^2\right), \tag{4}$$

where

$$B = \sum_{k=1}^{N} p_k^2 \sigma_k^2 + 6L\Gamma + 8(E-1)^2 G^2. \tag{5}$$

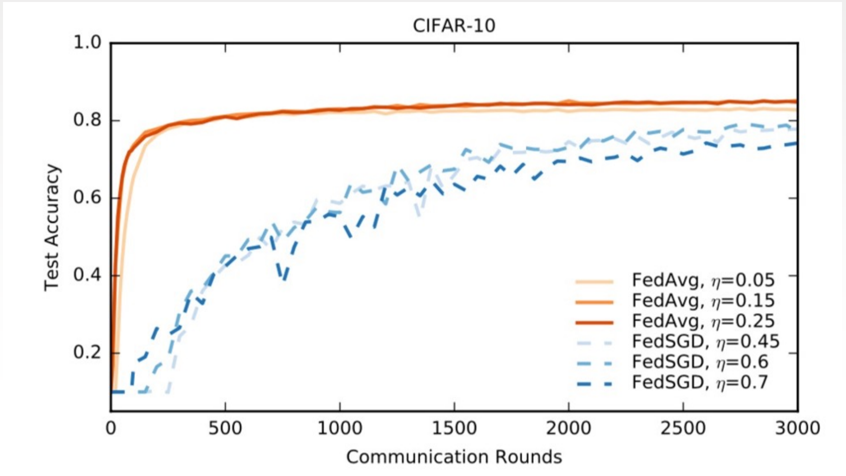
- Also guaranteed stationary point for smooth, non-convex objectives

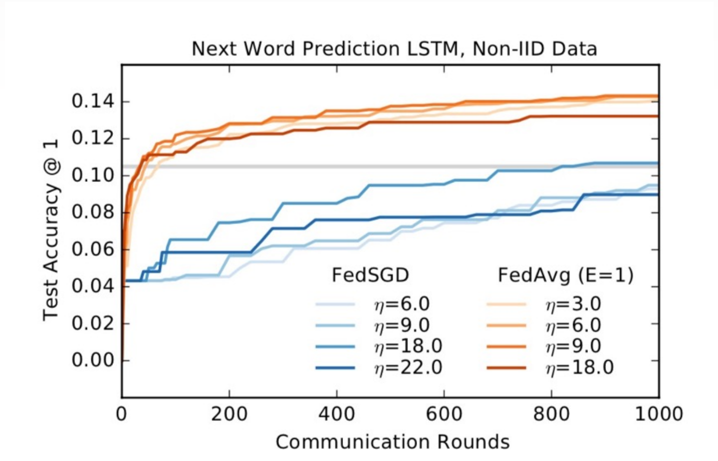
FedSGD

- Apply distributed gradient computation
  - Instead of models getting communicated, gradients are communicated
- Loop
  - For each client k
    - Select mini batch B_k
    - Compute $\omega_k = \frac{1}{|B_k|} \nabla f_k(w; B_k)$
    - Send to server ω_k
  - Server
    - Receive all ω_k
    - $w = w - \eta \sum_k \omega_k$
    - Send w to each client

- More communication than in FedAvg
- Privacy
  - Gradients reveal more information
  - More susceptible to attacks

Comparison


CIFAR-10
Test Accuracy
Communication Rounds
FedAvg, η=0.05
FedAvg, η=0.15
FedAvg, η=0.25
FedSGD, η=0.45
FedSGD, η=0.6
FedSGD, η=0.7


Next Word Prediction LSTM, Non-IID Data
Test Accuracy @ 1
Communication Rounds
FedSGD
η=6.0
η=9.0
η=18.0
η=22.0
FedAvg (E=1)
η=3.0
η=6.0
η=9.0
η=18.0

Other Aspects of FL

- How to select validation/test data
  - Particularly in non-iid
- Normalization/standardization
- Full client participation vs partial

- Descriptive statistics sharing

ALGORITHMS FOR NON-IID HORIZONTAL FL

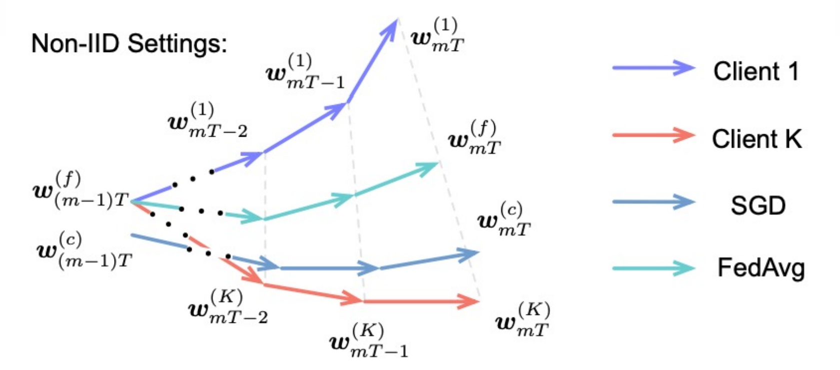
Ye Xue, Diego Klabjan, Yuan Luo

Northwestern | ENGINEERING

Motivation

- How non-IID challenges FedAvg's aggregation
  - $w_t = \sum_{k \in \mathcal{K}} \frac{n_k}{n} w_t^k$
    - Client updates w_t^k are learned from non-IID data, thus can be highly skewed. Averaging updates does not yield a similar model as training on global data. Weighted averaging can make it worse if a highly skewed client is large

- Question
  - Can each **local model** be trained on data that are **representative** of the global distribution at the time of aggregation, without sharing data?



Non-IID Settings:
$\boldsymbol{w}_{mT}^{(1)}$
$\boldsymbol{w}_{mT-1}^{(1)}$
$\boldsymbol{w}_{mT-2}^{(1)}$
$\boldsymbol{w}_{mT}^{(f)}$
$\boldsymbol{w}_{(m-1)T}^{(f)}$
$\boldsymbol{w}_{mT}^{(c)}$
$\boldsymbol{w}_{(m-1)T}^{(c)}$
$\boldsymbol{w}_{mT-2}^{(K)}$
$\boldsymbol{w}_{mT-1}^{(K)}$
$\boldsymbol{w}_{mT}^{(K)}$
Client 1
Client K
SGD
FedAvg

Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., & Chandra, V. (2018). Federated learning with non-iid data. arXiv preprint arXiv:1806.00582.

RADFed – Randomized Aggregation Delayed

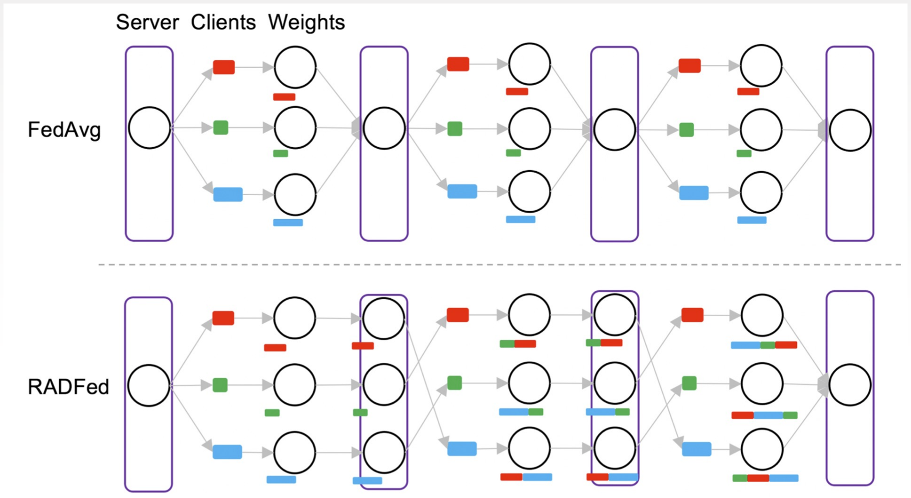


Fig. 1: A comparison of aggregation between RADFed and FedAvg. Colored blocks represent the data on clients. Each color represents a class. The length of each block shows the size of the data. Each circle represents a model's weights. Colored blocks under each circle show the data used to train a certain set of weights.

- FedAvg:
  - 3 aggregations
  - At each aggregation, each set of local model weights is trained on data of a different class
- RADFed:
  - 1 aggregation
  - Each set of local model weights is trained on data of all three classes (i.e., more representative of the global distribution)
  - Bonus: same (or similar) number of samples trained at aggregation
  - Cost: fewer aggregations but the same total number of communication rounds.

RADFed – Randomized Aggregation Delayed

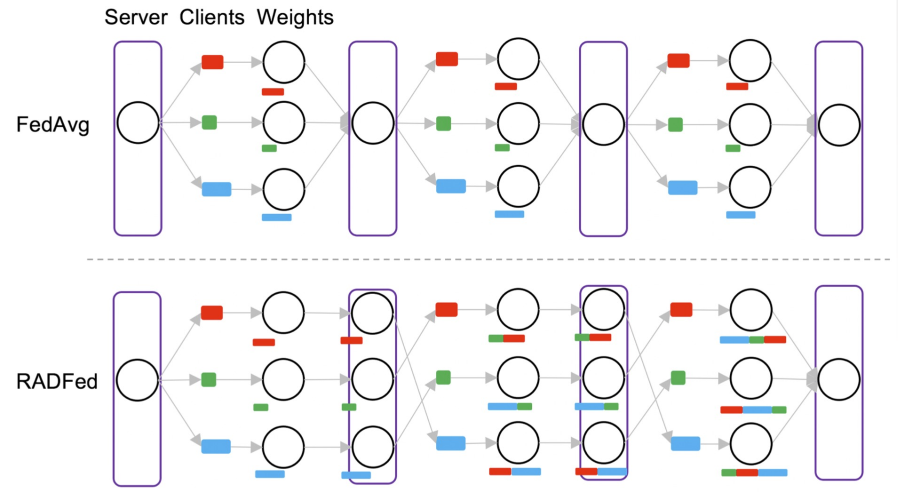
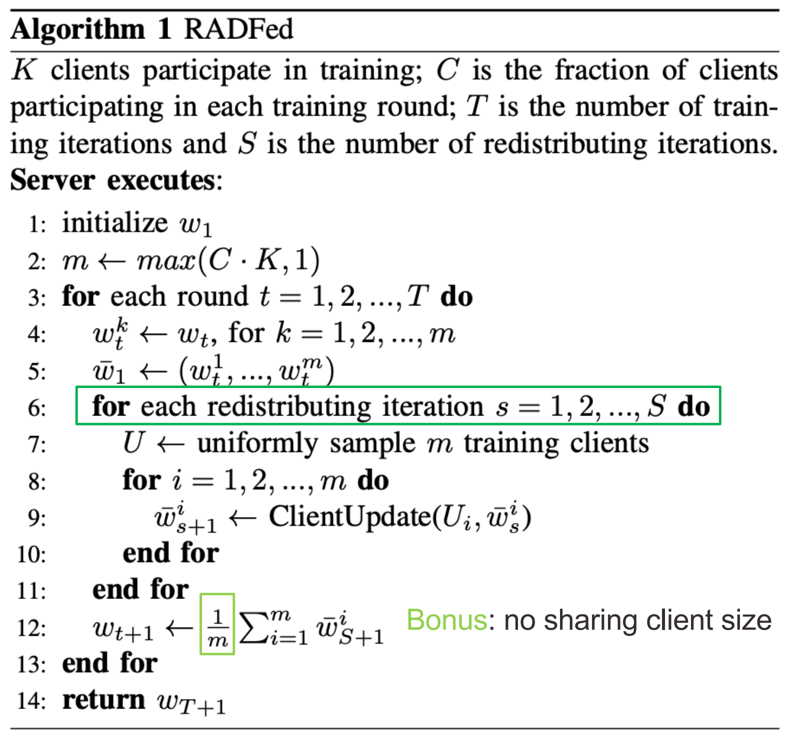


Fig. 1: A comparison of aggregation between RADFed and FedAvg. Colored blocks represent the data on clients. Each color represents a class. The length of each block shows the size of the data. Each circle represents a model's weights. Colored blocks under each circle show the data used to train a certain set of weights.



Algorithm 1 RADFed

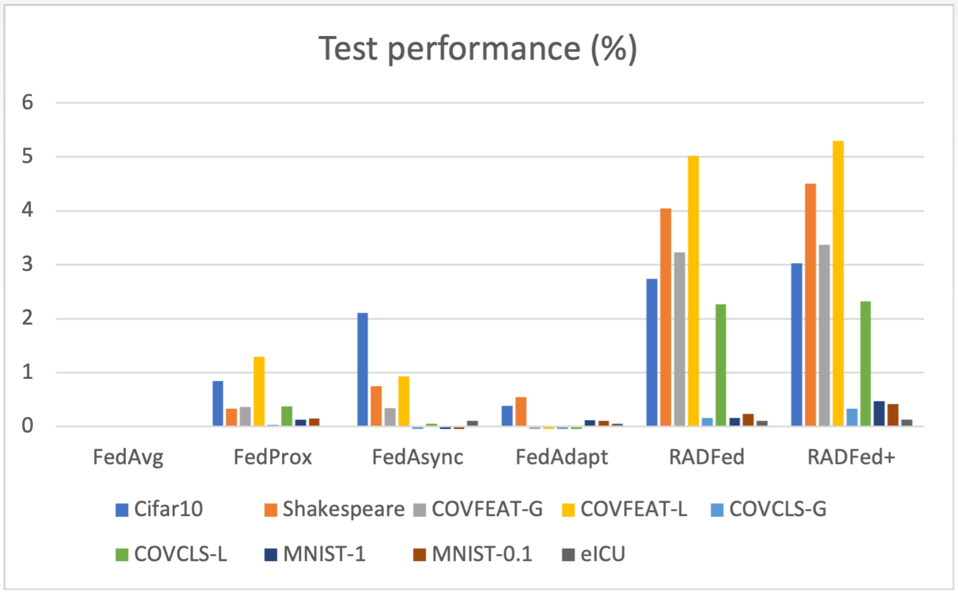
K clients participate in training; C is the fraction of clients participating in each training round; T is the number of training iterations and S is the number of redistributing iterations.
Server executes:

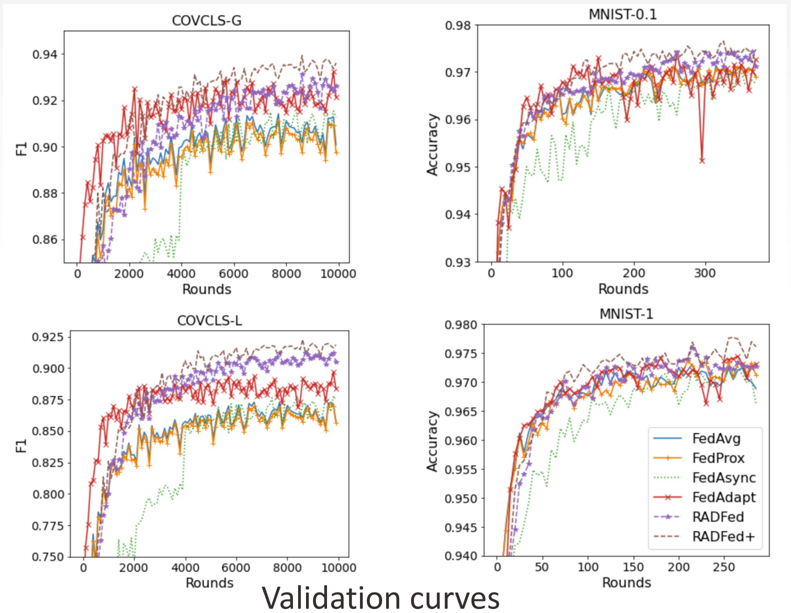
```
1:  initialize $w_1$
2:  $m \leftarrow max(C \cdot K, 1)$
3:  for each round $t = 1, 2, ..., T$ do
4:      $w_t^k \leftarrow w_t$, for $k = 1, 2, ..., m$
5:      $\bar{w}_1 \leftarrow (w_t^1, ..., w_t^m)$
6:      $\boxed{\text{for each redistributing iteration } s = 1, 2, ..., S \text{ do}}$
7:          $U \leftarrow$ uniformly sample $m$ training clients
8:          for $i = 1, 2, ..., m$ do
9:              $\bar{w}_{s+1}^i \leftarrow \text{ClientUpdate}(U_i, \bar{w}_s^i)$
10:         end for
11:     end for
12:     $w_{t+1} \leftarrow \boxed{\frac{1}{m}} \sum_{i=1}^{m} \bar{w}_{S+1}^i$
13: end for
14: return $w_{T+1}$
```

Bonus: no sharing client size

Performance

- Using Ray to simulate distributed training for over 100 clients

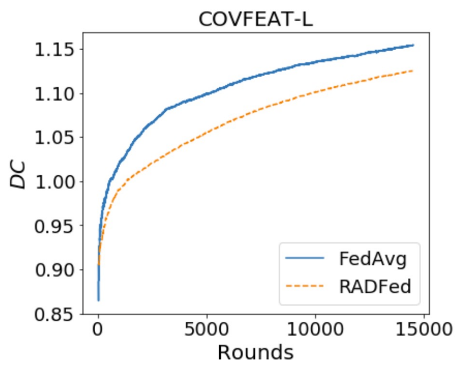




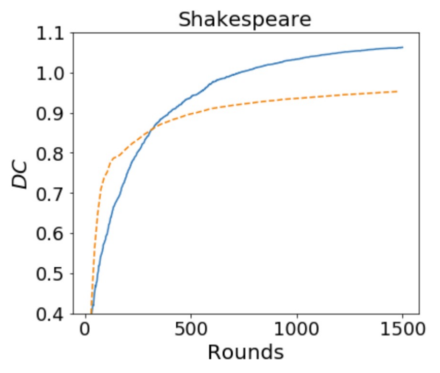
Validation curves

Lower Divergence

- Divergence between the Centralized and federated models (DC)[5]
  - $DC(t) = \frac{||w_{FL}^{t} - w_{C}^{t}||}{||\ w_{C}^{t}||}$
    - w_{FL}^{t}: the weights of the global model in federated training at the t-th round
    - w_{C}^{t}: the weights in centralized training



(a)

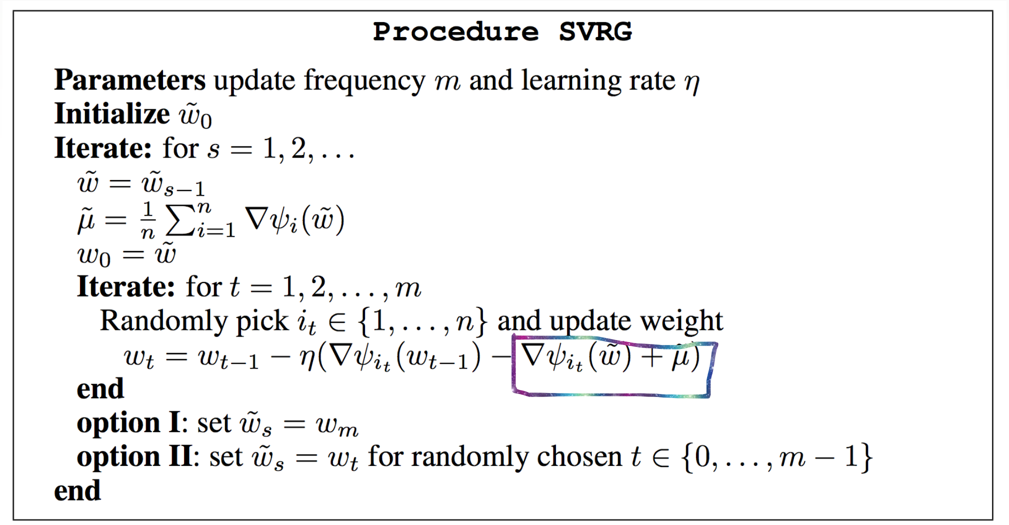


(b)

[5] Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., & Chandra, V. (2018). Federated learning with non-iid data. arXiv preprint arXiv:1806.00582.

Two Ideas from Standard Optimization

- Stochastic hedging
  - Force optimization variables/parameters to agree
- Variance reduction (SVRG)



Procedure SVRG

```
Parameters update frequency $m$ and learning rate $\eta$
Initialize $\tilde{w}_0$
Iterate: for $s = 1, 2, \ldots$
  $\tilde{w} = \tilde{w}_{s-1}$
  $\tilde{\mu} = \frac{1}{n}\sum_{i=1}^{n} \nabla\psi_i(\tilde{w})$
  $w_0 = \tilde{w}$
  Iterate: for $t = 1, 2, \ldots, m$
    Randomly pick $i_t \in \{1, \ldots, n\}$ and update weight
      $w_t = w_{t-1} - \eta(\nabla\psi_{i_t}(w_{t-1}) - \boxed{\nabla\psi_{i_t}(\tilde{w}) + \tilde{\mu})}$
  end
  option I: set $\tilde{w}_s = w_m$
  option II: set $\tilde{w}_s = w_t$ for randomly chosen $t \in \{0, \ldots, m-1\}$
end
```

Johnson, Rie, and Tong Zhang.
"Accelerating stochastic gradient descent using predictive variance reduction."
In Advances in Neural Information Processing Systems, pp. 315-323. 2013

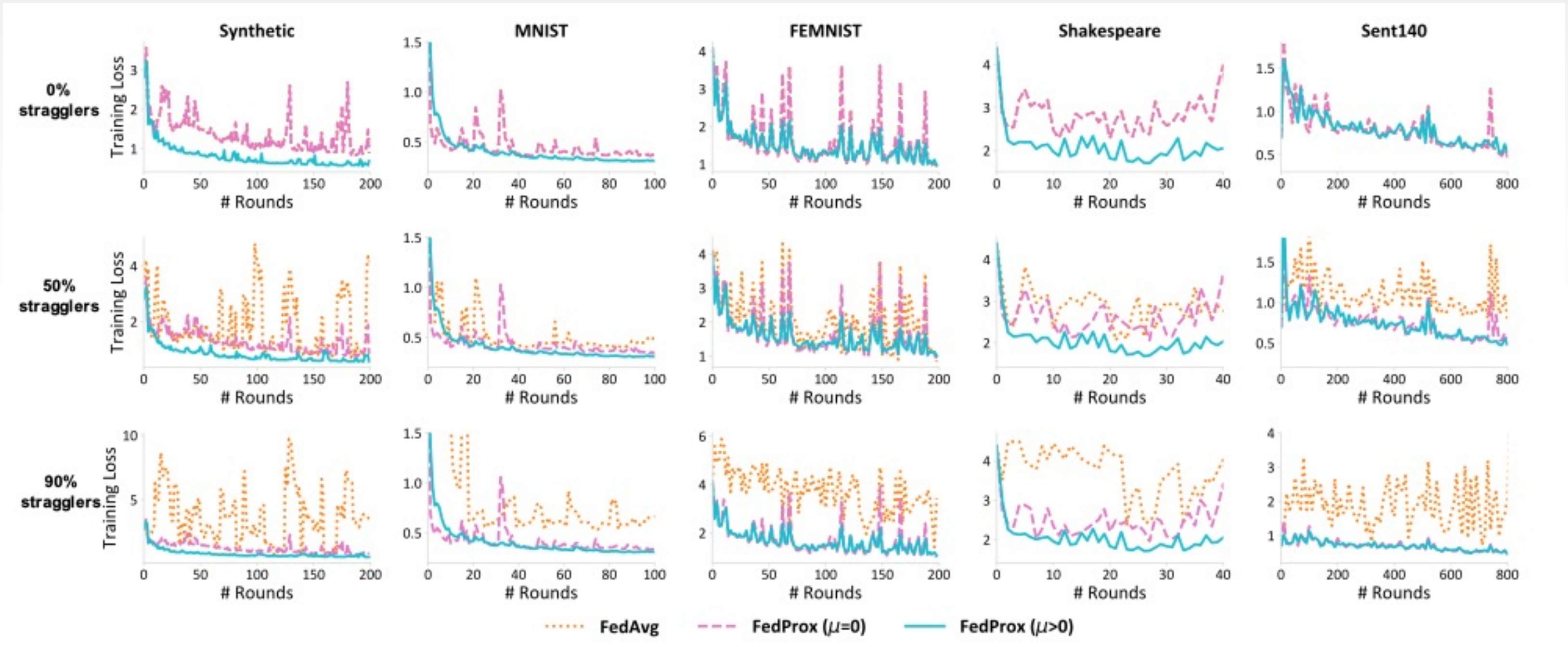
Other FedAvg-Style Algorithms

- SCAFFOLD
  - Uses control variables on clients and server to reduce the client drift
  - Improves convergence in non-IID settings
- FedProx
  - Adds a proximal term to each local objective function to be minimized that serves to minimize the drift of the local weights from the global weights
- FedSAM
  - Uses sharpness aware minimization (SAM) methods on each local optimization problem to help convergence to a wider minima on the global problem

FedProx [Li, et al. 2020]

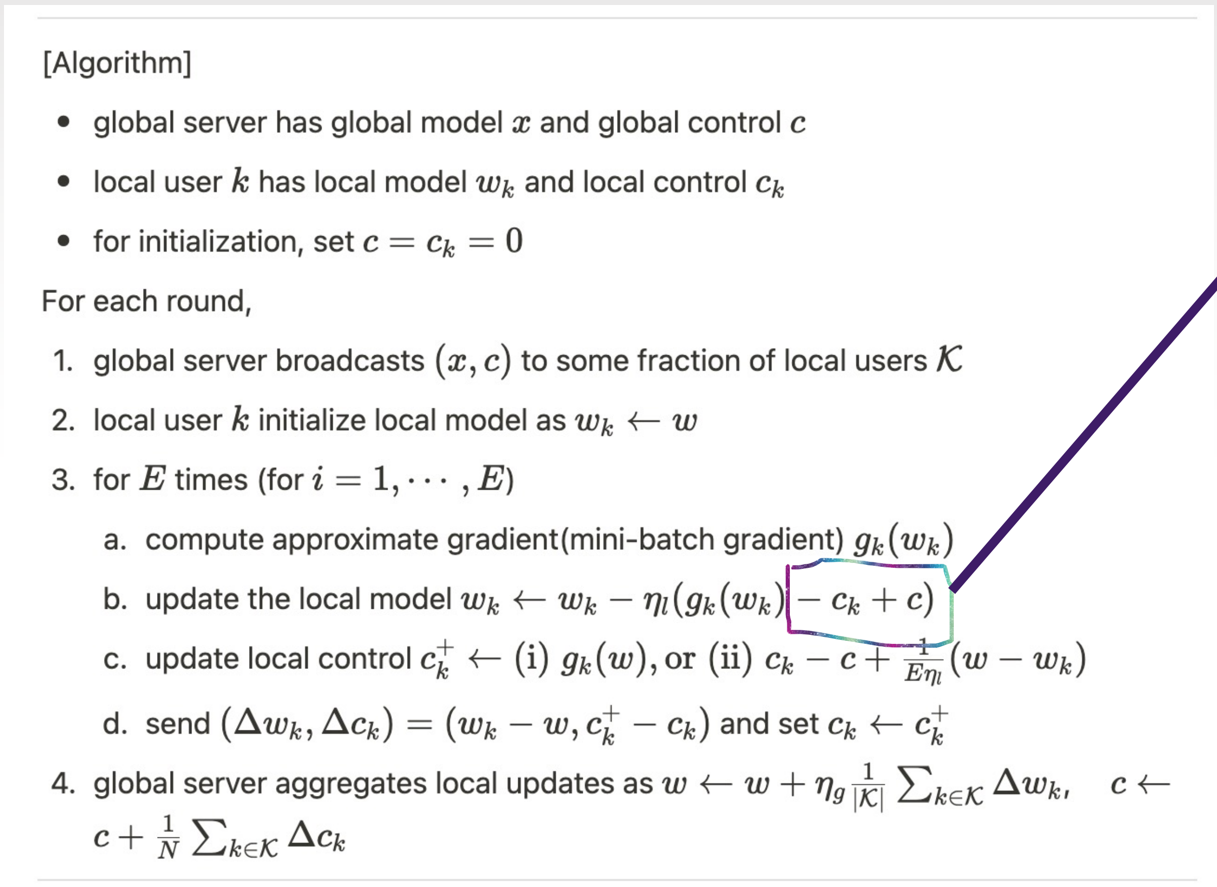
- Incentivize clients to produce the same model (weights)
- Same framework as FedAvg
- Clients solve ‘regularized’ problem
  - $loss = F_k(w) + \frac{\mu}{2}||w - w^t||^2$
  - Minimize original loss + coordination incentive
  - w^t = current global solution obtained/maintained by the server

FedProx



Synthetic
MNIST
FEMNIST
Shakespeare
Sent140
0% stragglers
50% stragglers
90% stragglers
Training Loss
# Rounds
FedAvg
FedProx (μ=0)
FedProx (μ>0)

- SCAFFOLD [Karimireddy et al, 2019]



[Algorithm]

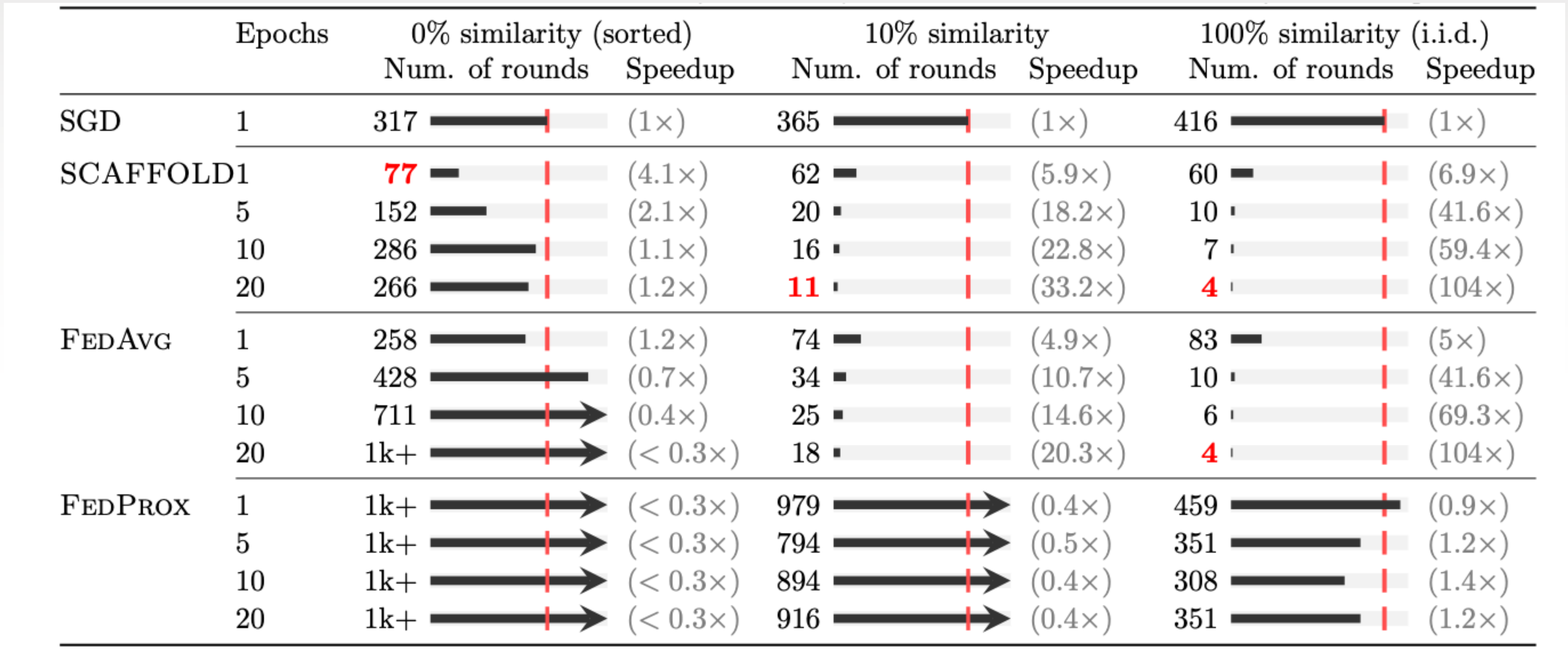
- global server has global model x and global control c
- local user k has local model w_k and local control c_k
- for initialization, set $c = c_k = 0$

For each round,

1. global server broadcasts (x, c) to some fraction of local users $\mathcal{K}$
2. local user k initialize local model as $w_k \leftarrow w$
3. for E times (for $i = 1, \cdots, E$)
   a. compute approximate gradient(mini-batch gradient) $g_k(w_k)$
   b. update the local model $w_k \leftarrow w_k - \eta_l(g_k(w_k)\boxed{- c_k + c)}$
   c. update local control $c_k^+ \leftarrow \text{(i)}\ g_k(w), \text{or (ii)}\ c_k - c + \frac{1}{E\eta_l}(w - w_k)$
   d. send $(\Delta w_k, \Delta c_k) = (w_k - w, c_k^+ - c_k)$ and set $c_k \leftarrow c_k^+$
4. global server aggregates local updates as $w \leftarrow w + \eta_g \frac{1}{|\mathcal{K}|} \sum_{k \in \mathcal{K}} \Delta w_k, \quad c \leftarrow c + \frac{1}{N} \sum_{k \in \mathcal{K}} \Delta c_k$

- First round equals FedAvg
- Borrows SVRG ideas
- Requires fewer rounds of communication
- Not affected by data heterogeneity or client sampling

SCAFFOLD



	Epochs	0% similarity (sorted)		10% similarity		100% similarity (i.i.d.)	
		Num. of rounds	Speedup	Num. of rounds	Speedup	Num. of rounds	Speedup
SGD	1	317	(1×)	365	(1×)	416	(1×)
SCAFFOLD	1	**77**	(4.1×)	62	(5.9×)	60	(6.9×)
	5	152	(2.1×)	20	(18.2×)	10	(41.6×)
	10	286	(1.1×)	16	(22.8×)	7	(59.4×)
	20	266	(1.2×)	**11**	(33.2×)	**4**	(104×)
FedAvg	1	258	(1.2×)	74	(4.9×)	83	(5×)
	5	428	(0.7×)	34	(10.7×)	10	(41.6×)
	10	711	(0.4×)	25	(14.6×)	6	(69.3×)
	20	1k+	(< 0.3×)	18	(20.3×)	**4**	(104×)
FedProx	1	1k+	(< 0.3×)	979	(0.4×)	459	(0.9×)
	5	1k+	(< 0.3×)	794	(0.5×)	351	(1.2×)
	10	1k+	(< 0.3×)	894	(0.4×)	308	(1.4×)
	20	1k+	(< 0.3×)	916	(0.4×)	351	(1.2×)

FedSAM [Caldarola et al, 2022]

- Flat minima generalize better
  - Empirical and theoretical evidence
- Flatness
  - Smallest eigenvalue of Hessian
  - Sharpness $\max_{||\epsilon||_p \leq \rho} \mathcal{L}_{\mathcal{D}}(\theta + \epsilon) - \mathcal{L}_{\mathcal{D}}(\theta)$

$$\min_{\theta \in \mathbb{R}^d} \max_{||\epsilon||_p \leq \rho} \mathcal{L}_{\mathcal{D}}(\theta + \epsilon) + \lambda ||\theta||_2^2$$

- First order Taylor approximation $\bar{\epsilon}(\theta) =$

$$\arg\max_{||\epsilon||_2 \leq \rho} \mathcal{L}_{\mathcal{D}}(\theta) + \epsilon^T \nabla_\theta \mathcal{L}_{\mathcal{D}}(\theta) = \rho \frac{\nabla_\theta \mathcal{L}_{\mathcal{D}}(\theta)}{||\nabla_\theta \mathcal{L}_{\mathcal{D}}(\theta)||_2}$$

$$\min_\theta L_D(\theta + \bar{\epsilon}(\theta))$$

- Problem?
- FedSAM: clients use such updates