NC STATE UNIVERSITY

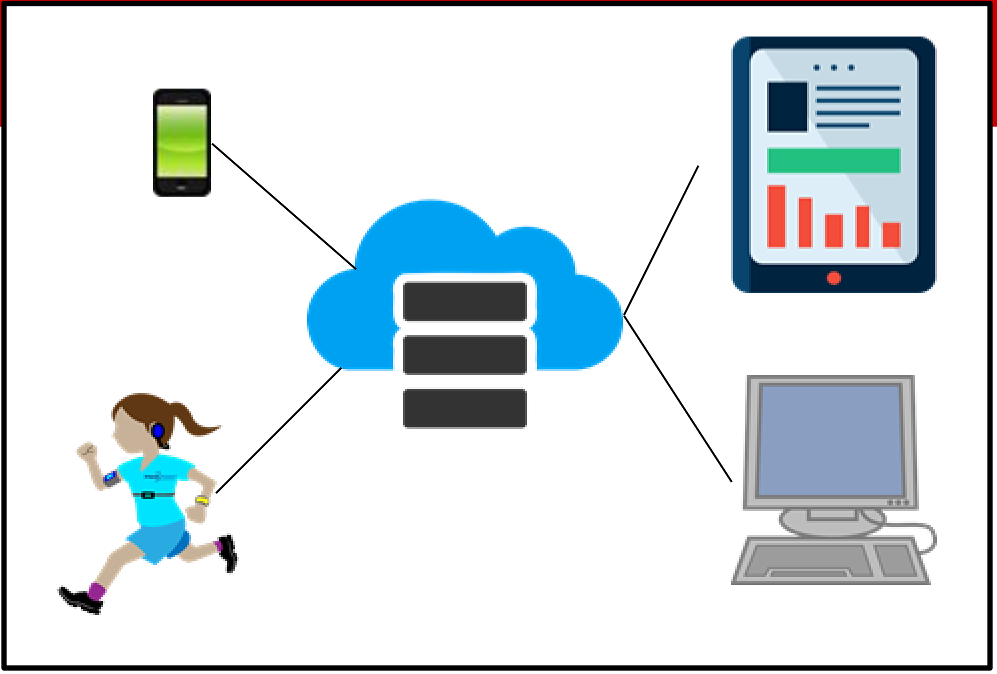
Federated Learning for Private and Efficient Computation

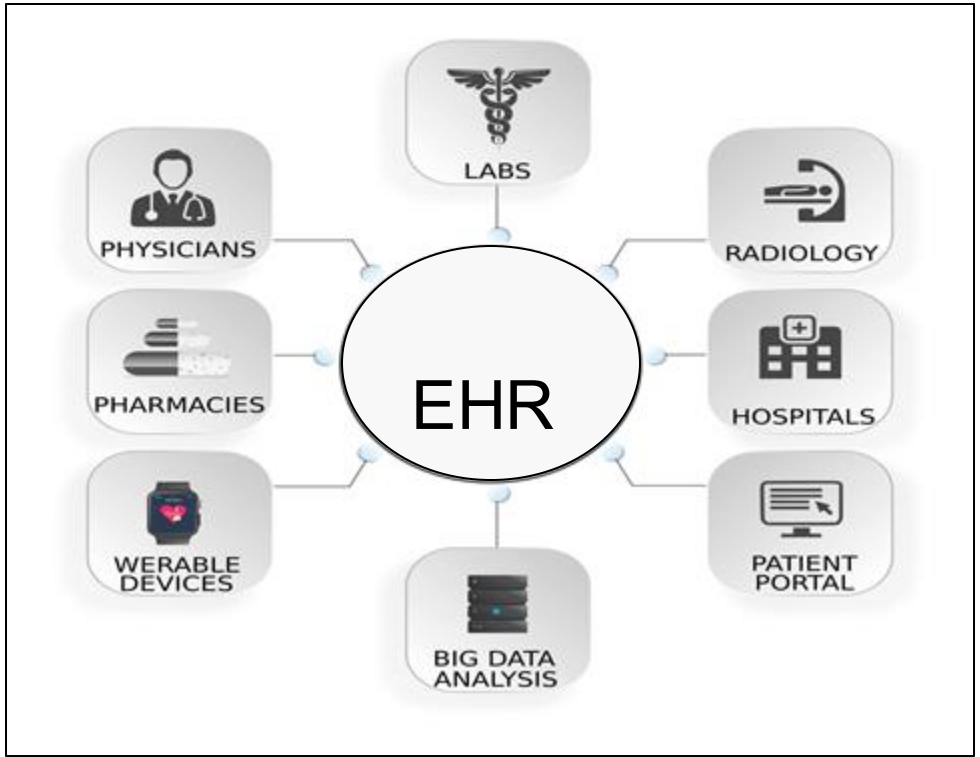
Yaqin Si

Bhaskar Ray

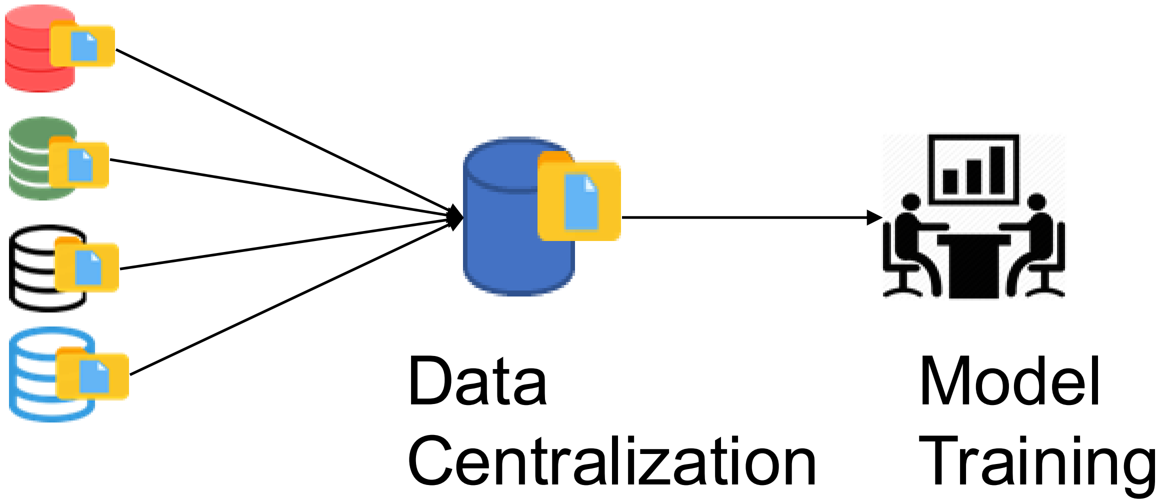
Motivation

- Data is ubiquitous and valuable
- Data is sensitive and vulnerable
    - Mobile devices
    - Electronic Health Records





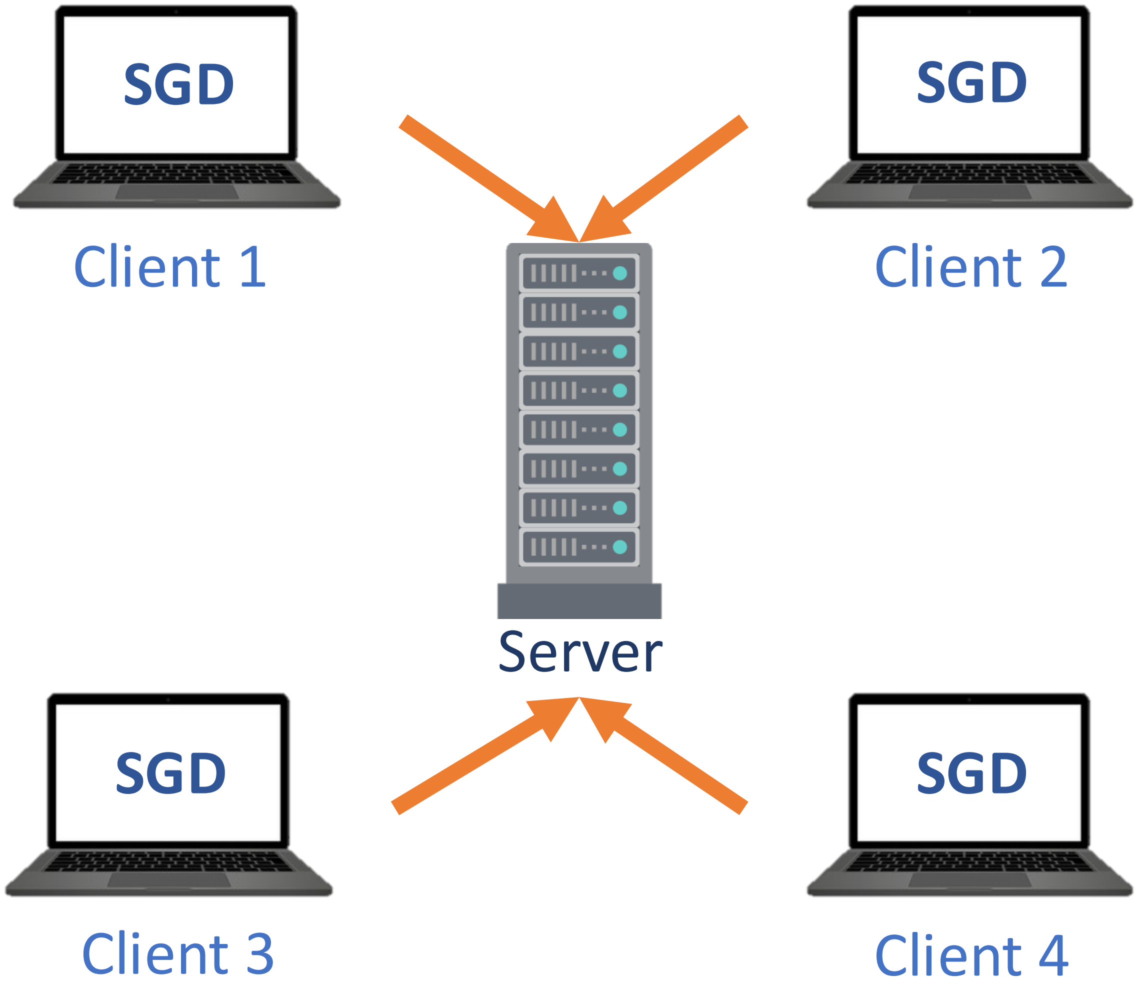
- Standard Machine learning



- Federated Learning:
  - Protect privacy
  - Apply to increasing data volume

The "FederatedAveraging" Algorithm

Communication-Efficient Learning of Deep Networks from Decentralized Data
- Brendan McMahan et al.



- ❑Centralized System
- ❑SGD on Individual Client
- ❑Aggregation at Server

Pooled Data: $x_1, \dots, x_n$

Model Parameter: β

Loss Function: $\ell(x;\beta)$

Overall Average Loss: $f(\beta) = \frac{1}{n}\sum_{i=1}^{n} \ell(x_i;\beta) \rightarrow \sum_{k=1}^{K} \underbrace{\sum_{i=1}^{n_k} \ell(x_{ki};\beta)}_{F_k(\beta)}$

Client k

Data (D_k): $x_{k1},\ x_{k2}, \dots, x_{kn_k}$

SGD based on β from server:

- Training in batches
- For batch b, update:

$$\beta = \beta - \eta \nabla \ell(\beta; b)$$

Iterate over all batches

$\times\ E$

K Clients

β_{t+1}^{k}

Server

Initialize Parameter: β_0

At round t: β_t

Aggregating all K client outputs:

$$\beta_{t+1} = \sum_{k=1}^{K} \beta_{t+1}^{k}$$

Technical Considerations

➢ Validity of delegation

Considering IID setup, D_k is distributed as a random partition of entire dataset.

$$\mathbb{E}_{D_k}\left(F_k(\beta)\right) = f(\beta)$$

For non-IID cases, experiments by the authors depict lower accuracy and slower convergence to higher accuracy values

➢ Initialization of β

Based on experiments by the authors, a random initialization at the server level leads to significant reduction in loss.

➢ Choice of objective function

Convex function desirable for averaging.

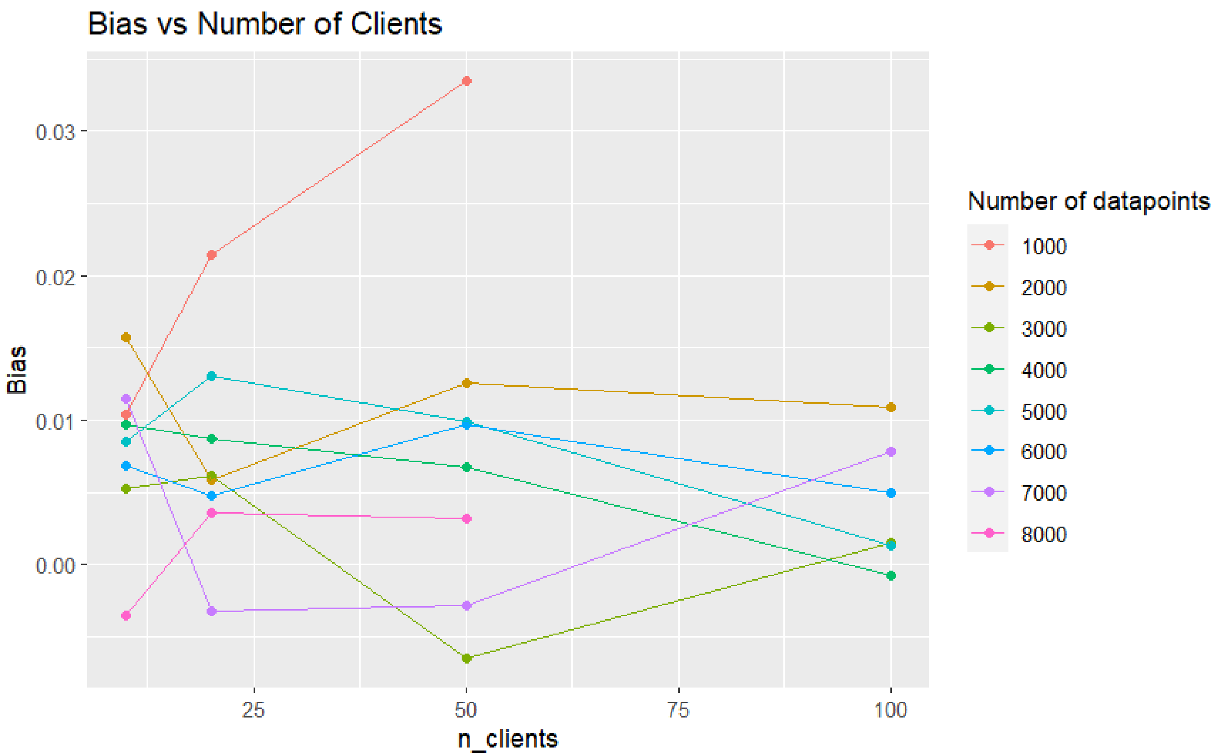
→depends on $\ell(\cdot;\cdot)$

Simulation Study

$$X_i = (Z_i, Y_i)$$

$$Z_i \sim N(1,2)\ IID \qquad Y_i \sim \text{Ber}\left(\frac{1}{1+\exp[-\beta x]}\right) \text{ indep}$$

- Total no. of datapoints $n = 1000, 2000, \ldots, 10000$
- No. of Clients $K = 10, 20, 50, 100$
- $n_k = \frac{n}{K}$
- No. of batches, $B = \lfloor n_k \rfloor$
- $E = 1$
- $\eta = 0.1$



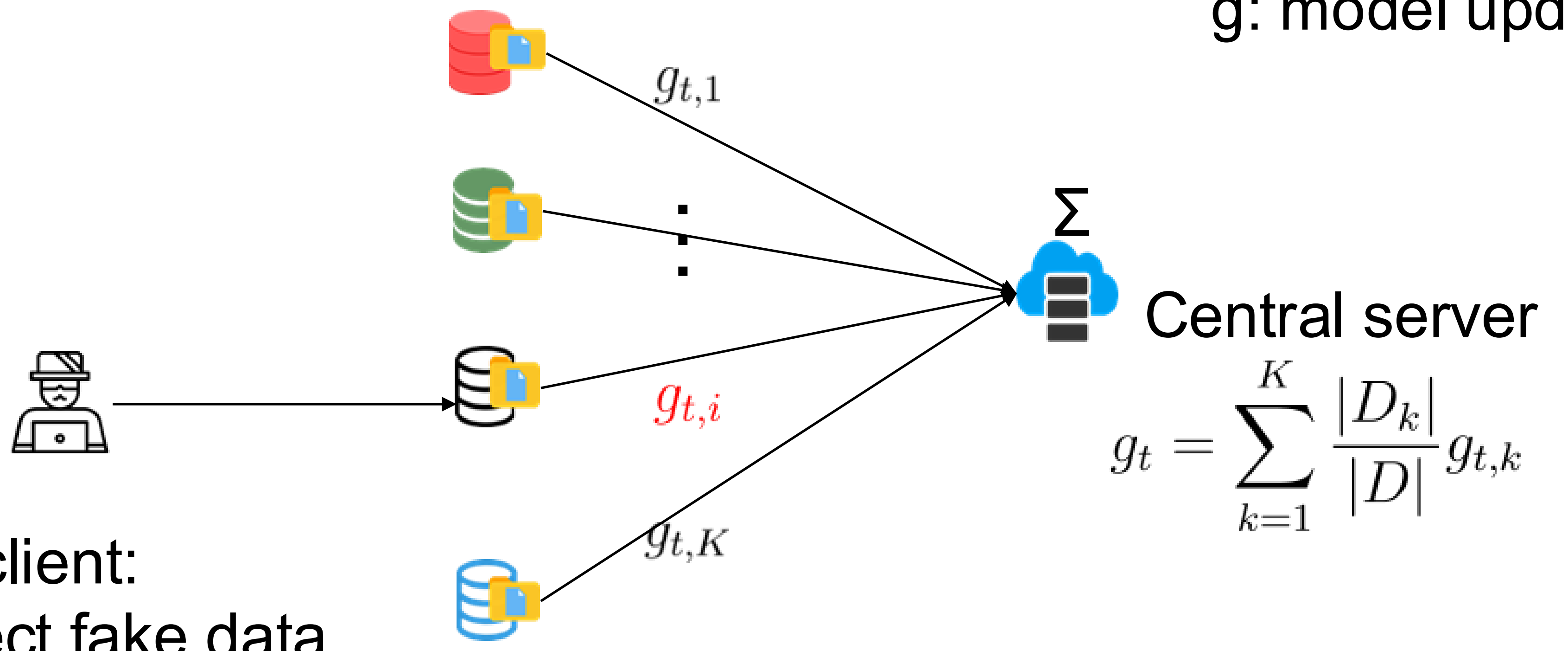
Future Prospects

- More extensive simulation studies for properties.
- Use of *Momentum* instead of *simple* Stochastic Gradient Descent.

Robust Federated Learning to Deal with Malicious Clients

FedAvg under Attack

t: Iteration
1,…,K: indicator of clients
g: model updates



$$g_t = \sum_{k=1}^{K} \frac{|D_k|}{|D|} g_{t,k}$$

Attack client:
- Inject fake data,
- Manipulate model updates directly

Model update with problematic direction and magnitude.

Robust Aggregation Rules

t: iteration
1,…,K: indicator of clients
g: model updates
m: number of malicious clients
Γ: a subset

- Coordinate-wise adjustment
  - Median

$$g_t = \mathrm{Median}\{g_{t,1}, \cdots, g_{t,K}\}$$

  - Trimmed mean

$$g_t = \frac{1}{K-2m} \sum_{g_{t,k} \in \Gamma_{t,K-2m}} g_{t,k}$$

- Distance-based score adjustment
  - Krum

$$Score_{ED}(k) = \sum_{g_{t,i} \in \Gamma_{t,K-m-2}} \|g_{t,i} - g_{t,k}\|_2^2$$

$$g_t = g_{t,k} \text{ where client } k \text{ has } \min(\mathrm{Score}_{ED})$$

Robust Aggregation rules

- Coordinate-wise adjustment
  - Median
  - Trimmed mean

Less than half of the clients are malicious

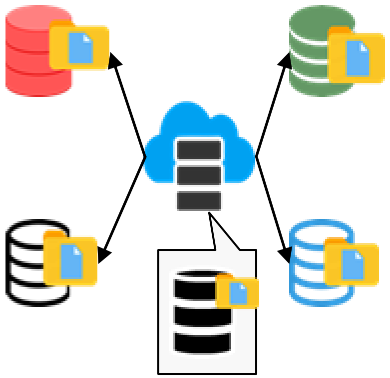
- Distance-based adjustment
  - Krum

High computation cost

Limitation:

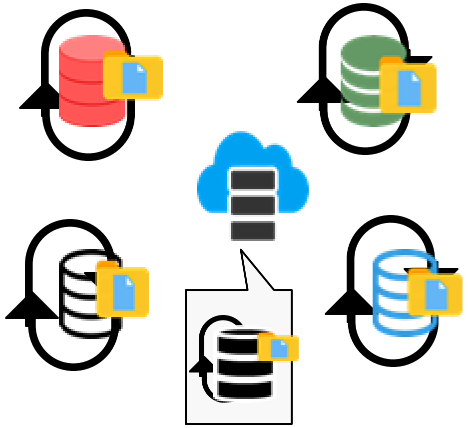
- ➢ Information loss
- ➢ No ground truth

FLTrust Algorithm



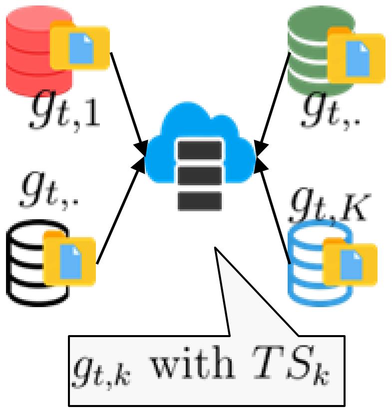
Collect "root data set"

Distribute Global model



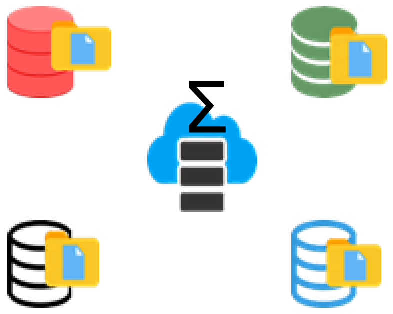
Train "server model"

Train local model



Send local models to server

Bootstrap trust score (TS)



Aggregate model according to trust score

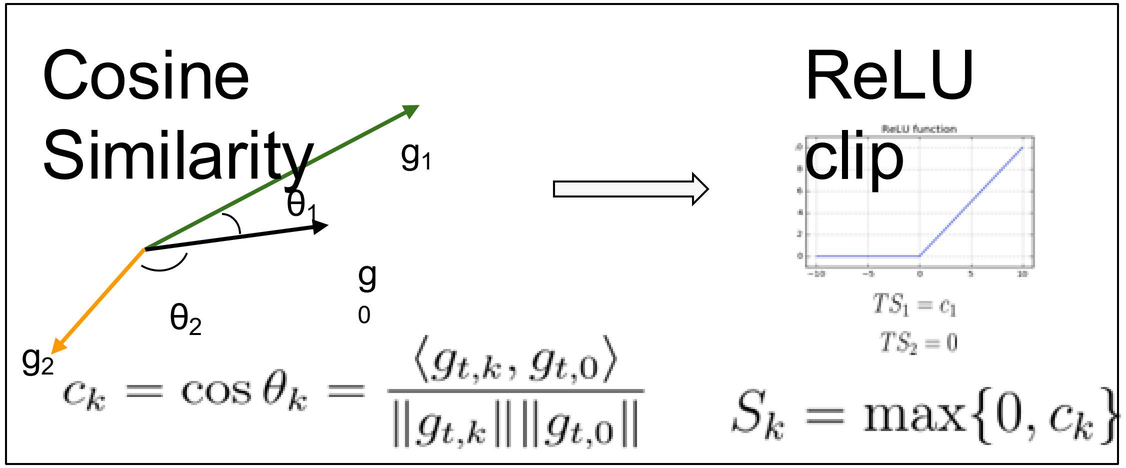
Repeat until training complete

FLTrust algorithm

t: iteration
1,…,K: indicator of clients
g_0: server model updates
g_k: client model updates

- Trust score
  - ReLU-Clipped Cosine Similarity: correct direction

$$S_k = ReLU\left(\frac{\langle g_{t,k}, g_{t,0}\rangle}{\|g_{t,k}\|\,\|g_{t,0}\|}\right)$$

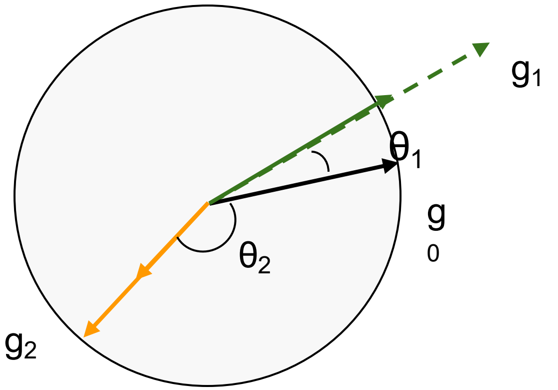


- Trust score based aggression
  - Normalization: correct magnitude
  - Aggregation

$$g_t = \frac{1}{\sum_{k=1}^{N} S_k} \sum_{k=1}^{K} S_k \tilde{g}_{t,k}$$

Normalization

$$\bar{g}_{t,k} = \frac{\|g_{t,0}\|_2}{\|g_{t,k}\|_2} g_{t,k}$$



FLTrust Full Algorithm

∇f Evaluated on random batches

Algorithm 2 FLTrust Algorithm

Require: global learning rate $\alpha \geq 0$, local learning rate $\eta \geq 0$, communication round T

Initialization: $\beta_0 =$ random initial

for $t = 1, 2, ..., T$ **do** ▷ Outer iteration for communication

  for $r = \{1, 2, \cdots, R\}$ **do**

    Initialize $\beta_{t,0,0} = \beta_{t-1}$

    $\beta_{t,0,r} = \beta_{t,0,r-1} - \eta \nabla f_{(r-1)}$ and use $\beta_{t,0} = \beta_{t,0,R}$ as server model

    Server model updates $g_{t,0} = \beta_{t,0} - \beta_{t-1,0}$

  end for

  for $k = \{1, 2, \cdots, K\}$ **do**

    for $r = \{1, 2, \cdots, R\}$ **do**

      Initialize $\beta_{t,k,r} = \beta_{t-1}$

      $\beta_{t,k,r} = \beta_{t,k,r-1} - \eta \nabla f_{(r-1)}$ and use $\beta_{t,k} = \beta_{t,k,R}$ as local model

      Local model update $g_{t,k} = \beta_{t,k} - \beta_{t-1,k}$

    end for

    $S_k = ReLU(\frac{\langle g_{t,k}, g_{t,0} \rangle}{\|g_{t,k}\| \|g_{t,0}\|}))$ ▷ Trust score

    $\tilde{g}_{t,k} = \frac{\|g_{t,0}\|_2}{\|g_{t,k}\|_2} g_{t,k}$ ▷ Normalize local model updates

  end for

  $g_t = \frac{1}{\sum_{k=1}^{N} S_k} \sum_{k=1}^{K} S_k \tilde{g}_{t,k}$

  Update global model as $\beta_t = \beta_{t-1} + \alpha g_t$

end for

FLTrust Properties

- Fidelity under no attack
  Low error rate in experiments compared to trimmed mean, median, krum; similar to FedAvg.
- Robustness under attack
  Experimental evidence.
- Efficiency
  Extra tasks compared to FedAvg: maintain server model, Trust score, and normalization. (Linear in number of clients, not quadratic)
- Security
  Bounded error between the global minimum and the FLTrust solution under assumptions (convex, differentiable, Lipschitz continuous gradient, independent local and root data, bounded local and global variance)

Decentralized Federated Learning Without Central Server

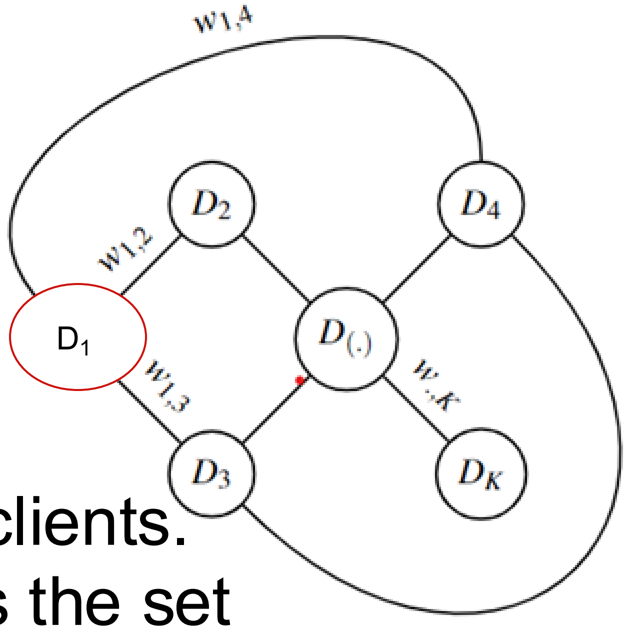
Problems

- Massive communication in FedAvg
  - Central server -- All clients

- Attack on the central server

Decentralized FedAvg with Momentum

- Graph-guided communication
  - No central server
  - Communicate to neighbors only



Graph: communication between K clients.

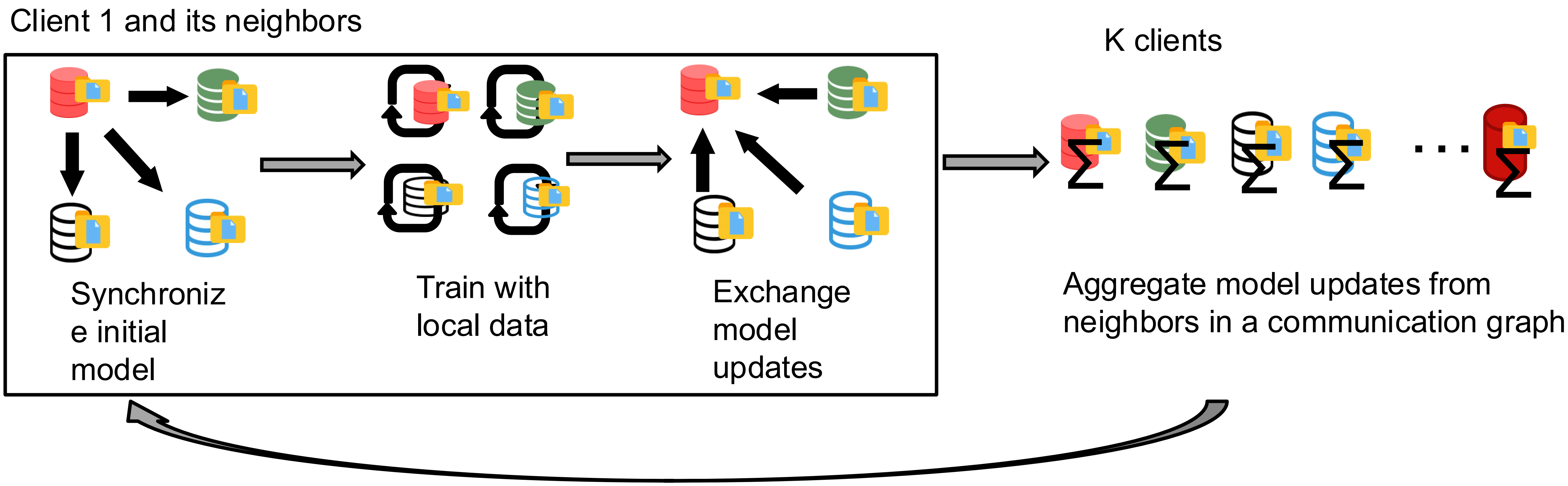
$\mathcal{N}(k)$: neighbor of client k, which is the set of nodes that directly connect to client k,

For example, client 1 (with data D_1) has 3 neighbors (D_2,D_3,and D_4)

	D_1	D_2	D_3	D_4
D_1	0	$w_{1,2}$	$w_{1,3}$	$w_{1,4}$
D_2	$w_{1,2}$	0	$w_{2,3}$	$w_{2,4}$
D_3	$w_{1,3}$	$w_{2,3}$	0	$w_{3,4}$
D_4	$w_{1,4}$	$w_{2,4}$	$w_{3,4}$	0

Mixing matrix for client 1 and its neighbors

Decentralized Federated Learning



DFedAvgM Full Algorithm

Algorithm 3 Decentralized FedAvg with Momentum (DFedAvgM)

Require: Aggregation learning rate $\alpha \geq 0$, local learning rate $\eta \geq 0$, communication round T, inner iteration $R \geq 1$, momentum $0 \leq \theta < 1$

Initialization: $\beta^{(0)} = 0$

for $t = \{1, 2, ..., T\}$ **do** ▷ Outer iteration for communication

  for $k = \{1, 2, \cdots, K\}$ **do**

    for $r = \{1, 2, \cdots, R\}$ **do** ▷ Inner iteration for local update

      Initialize $\beta_{t,k,0} = \beta_{t,k,-1} = \beta_{t-1}$

      $\beta_{t,k,r} = \beta_{t,k,r-1} - \eta \nabla f_{k,r-1} + \theta(\beta_{t,k,r-1} - \beta_{t,k,r-2})$ and use $\beta_{t,k} = \beta_{t,k,R}$ as local model

      Send $g_{t,k} = \beta_{t,k} - \beta_{t-1,k}$ to neighbors $\mathcal{N}(k)$

    end for

    model update of client k for $k \in \{1, 2, \cdots, K\}$ as $g_{t,k} = \sum_{i \in \mathcal{N}(k)} w_{i,k} g_{t,i}$

    Update client k for $k \in \{1, 2, \cdots, K\}$ as $\beta_{t,k} = \beta_{t-1,k} + \alpha g_{t,k}$

  end for

end for

DFedAvgM with quantization for communication efficiency

- Quantize a number

$$q(a) = \begin{cases} vs, w.p. 1 - \frac{a-ks}{s} \\ (v+1)s, w.p. \frac{a-ks}{s} \end{cases}$$

- Quantize a vector

$$Q(g) = \{q(g^1), \cdots, q(g^j), \cdots, q(g^d)\}$$

- Communication cost

The original system of 32 bits, and quantize to b bits.

v is a number that can be represented with b bits

$v \in \{-2^{b-1}, 2^{b-1}\}$

a: any number in system
s: as a base number
w.p.: with probability
g^j: model updates of dimension d

Q(g)	Data	Bits/vector	Total bits
Before	g^j	32d	$32d \times Deg(\mathcal{N}(j)) \times T$
After	$s+v^j$	32+bd	$32d \times Deg(\mathcal{N}(j)) \times T$

DFedAvgM with quantization

Algorithm 4 Quantized DFedAvgM

Require: Aggregation learning rate $\alpha \geq 0$, local learning rate $\eta \geq 0$, communication round T, inner iteration $R \geq 1$, momentum $0 \leq \theta < 1$

Initialization: $\beta^{(0)} = 0$

for $t = 1, 2, \ldots$ **do** ▷ Outer iteration for communication

  for $k = \{1, 2, \cdots, K\}$ **do**

    for $r = \{1, 2, \cdots, R\}$ **do** ▷ Inner iteration for local update

      Initialize $\beta_{t,k,0} = \beta_{t,k,-1} = \beta_{t-1}$

      $\beta_{t,k,r} = \beta_{t,k,r-1} - \eta \nabla f_{k,r-1} + \theta(\beta_{t,k,r-1} - \beta_{t,k,r-2})$ and use $\beta_{t,k} = \beta_{t,k,R}$ as local model

      $\boxed{\text{Send } g_{t,k} = Q(\beta_{t,k} - \beta_{t-1,k} \text{ to neighbors } \mathcal{N}(k)}$

    end for

    model update of client k for $k \in \{1, 2, \cdots, K\}$ as $g_{t,k} = \sum_{i \in \mathcal{N}(k)} w_{i,k} g_{t,i}$

    Update client k for $k \in \{1, 2, \cdots, K\}$ as $\beta_{t,k} = \beta_{t-1,k} + \alpha g_{t,k}$

  end for

end for

DFedAvgM and quantized algorithm Properties

Assumptions: differentiable, Lipschitz continuous gradient, bounded local and global variance

- With smooth loss function $T = \mathcal{O}(\frac{1}{\epsilon^2})$
  Converge at the same rate as SGD:
  DFedAvgM and QDFedAvgM
  Converge faster with more local inner iterations before communication

- With strong convexity
  Convergence rate improves: $T = \mathcal{O}(\frac{1}{\epsilon})$

Heterogeneity in Federated Learning

- Data at client affected by local environment

- Model parameter at client j:

$$\theta_i := (\beta, \gamma_i)$$

We are interested in estimating β only

- Data:

$$X_{ij} \sim f(x; \theta_i); \qquad i = 1, \dots, K; j = 1, \dots, n$$

- $\Gamma = \{\gamma_1, \dots, \gamma_K\}$ is the set of nuisance (local) parameters.

Complete Data log-likelihood

$$L(\beta, \Gamma) = \frac{1}{Kn} \sum_{i=1}^{K} \sum_{j=1}^{n} \log(f(x_{ij}; \beta, \gamma_i)) = \frac{1}{K} \sum_{i=1}^{K} L_i(\beta, \gamma_i)$$

Efficient Score Function

$$s_i(x; \beta, \gamma_i) = \nabla_\beta \log f(x; \beta, \gamma_i) - I_{\beta\gamma}^{(i)} {I_{\gamma\gamma}^{(i)}}^{-1} \nabla_\gamma \log f(x; \beta, \gamma_i)$$

$$I^{(i)} = \mathbb{E}(-\nabla^2 L_i(\theta_i^*))$$

Approach

- Use of *Surrogate Likelihood Method*.

- *Density ratio tilting* method for heterogeneity:

$$\frac{f(x_{ij};\beta,\gamma_1)}{f(x_{ij};\beta,\gamma_i)}$$

- Estimator of β is asymptotically normal.

Further Areas of Research

- ❑ Heterogenous set-ups with unbalanced data.
- ❑ Cases where data come from (possibly) different distributions.
- ❑ Decentralized set-up for heterogenous data.

Thank You