INTRODUCTION TO FEDERATED LEARNING

Aurélien Bellet (Inria)

Hi! PARIS Summer School on AI & Data for Science, Business and Society
July 8, 2021

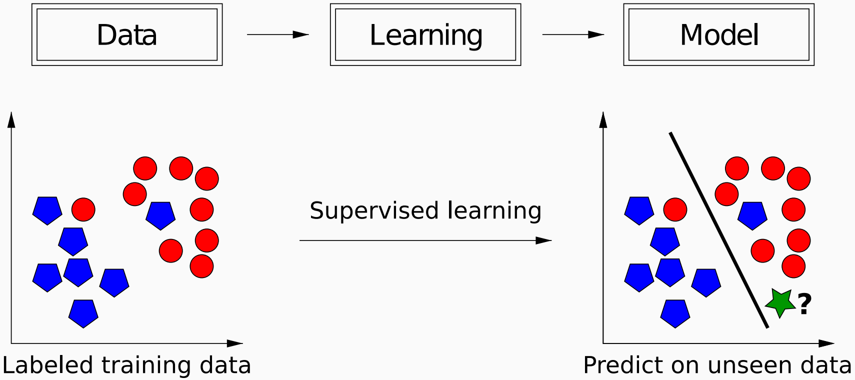
OUTLINE OF THE TALK

1. What is Federated Learning?
2. A baseline algorithm: FedAvg
3. Challenge 1: Dealing with non-IID data

   Zoom on learning personalized models via task relationships
4. Challenge 2: Preserving privacy

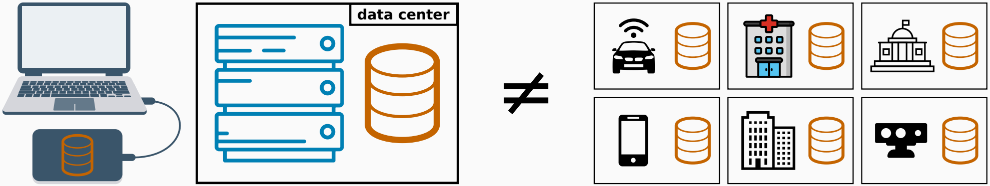
   Zoom on an accurate and scalable protocol for private aggregation
5. Wrapping up

What is Federated Learning?


Data
Learning
Model
Supervised learning
?
Labeled training data
Predict on unseen data

A SHIFT OF PARADIGM: FROM CENTRALIZED TO DECENTRALIZED DATA

- The standard setting in Machine Learning (ML) considers a centralized dataset processed in a tightly integrated system
- But in the real world data is often decentralized across many parties

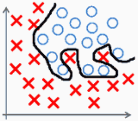


WHY CAN'T WE JUST CENTRALIZE THE DATA?

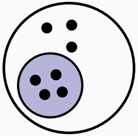
1. Sending the data may be too costly
    - Self-driving cars are expected to generate several TBs of data a day 
    - Some wireless devices have limited bandwidth/power 
2. Data may be considered too sensitive
    - We see a growing public awareness and regulations on data privacy 
    - Keeping control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

HOW ABOUT EACH PARTY LEARNING ON ITS OWN?

1. The local dataset may be too small
   - Sub-par predictive performance (e.g., due to overfitting)
   - Non-statistically significant results (e.g., medical studies)

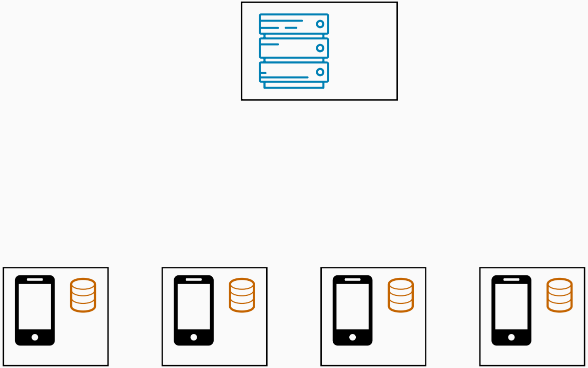


2. The local dataset may be biased
   - Not representative of the target distribution



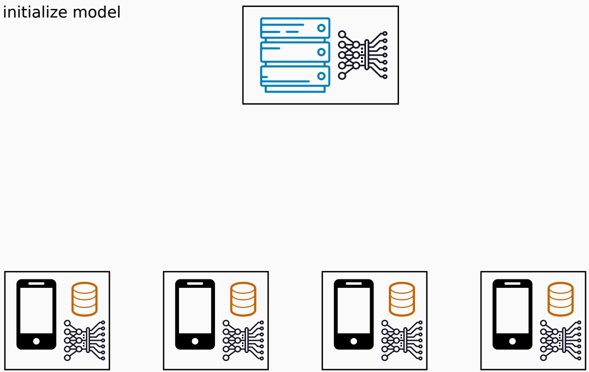
A BROAD DEFINITION OF FEDERATED LEARNING

- **Federated Learning (FL)** aims to collaboratively train a ML model while keeping the data decentralized



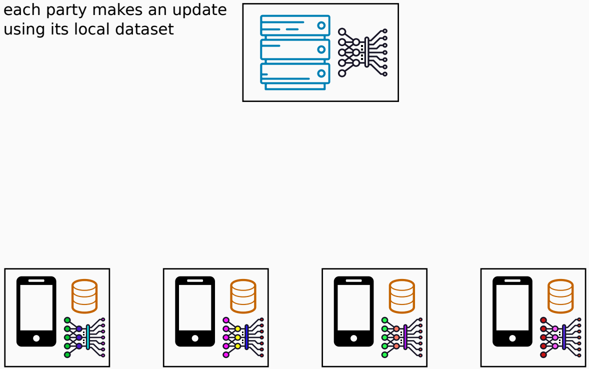
A BROAD DEFINITION OF FEDERATED LEARNING

- **Federated Learning (FL)** aims to collaboratively train a ML model while keeping the data decentralized

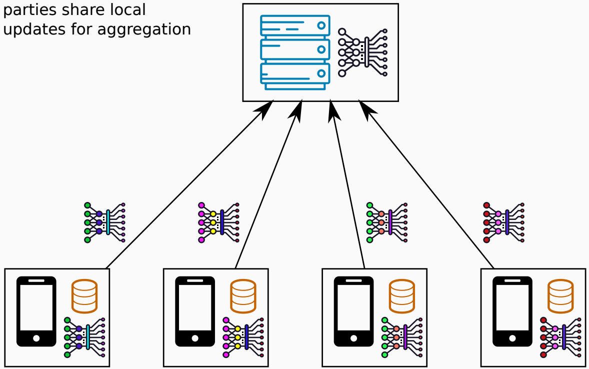


A BROAD DEFINITION OF FEDERATED LEARNING

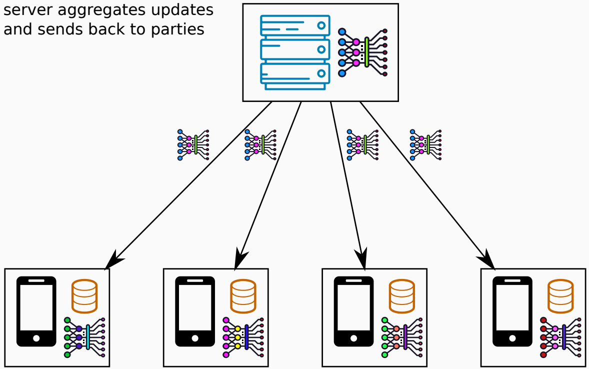
- **Federated Learning (FL)** aims to collaboratively train a ML model while keeping the data decentralized



- **Federated Learning (FL)** aims to collaboratively train a ML model while keeping the data decentralized

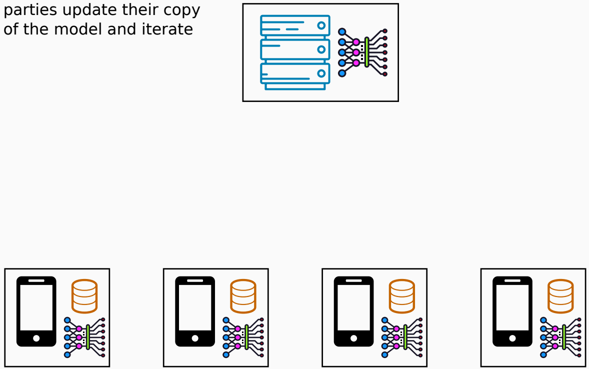


- **Federated Learning (FL)** aims to collaboratively train a ML model while keeping the data decentralized



A BROAD DEFINITION OF FEDERATED LEARNING

- **Federated Learning (FL)** aims to collaboratively train a ML model while keeping the data decentralized



- We would like the final model to be as good as the centralized solution (ideally), or at least better than what each party can learn on its own

KEY DIFFERENCES WITH DISTRIBUTED LEARNING

Data distribution

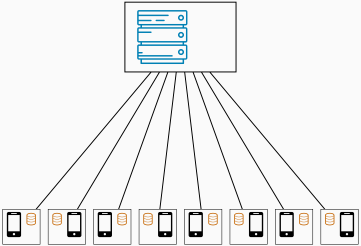
- In distributed learning, data is centrally stored (e.g., in a data center)
  - The main goal is just to train faster
  - We control how data is distributed across workers: usually, it is distributed uniformly at random across workers
- In FL, data is naturally distributed and generated locally
  - Data is **not** independent and identically distributed (non-IID), and it is imbalanced

Additional challenges that arise in FL

- Dealing with the possibly limited reliability/availability of participants
- Enforcing privacy constraints
- Achieving robustness against malicious parties
- ...

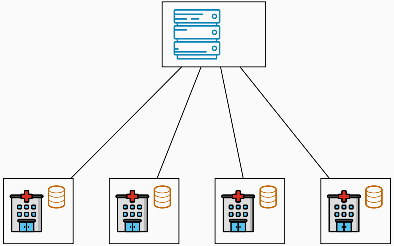
CROSS-DEVICE VS. CROSS-SILO FL

Cross-device FL



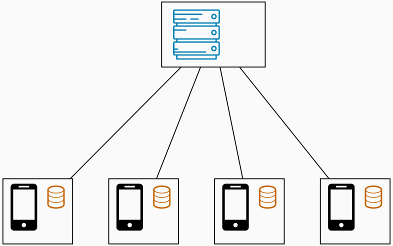
- Massive number of parties (up to 10^{10})
- Small dataset per party (could be size 1)
- Limited availability and reliability
- Some parties may be malicious

Cross-silo FL



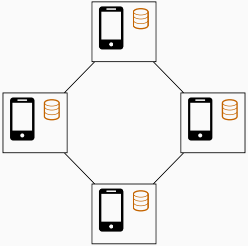
- 2-100 parties
- Medium to large dataset per party
- Reliable parties, almost always available
- Parties are typically honest

Server-orchestrated FL



- Server-client communication
- Global coordination, global aggregation
- Server is a single point of failure and may become a bottleneck

Fully decentralized FL



- Device-to-device communication
- No global coordination, local aggregation
- Naturally scales to a large number of devices

FEDERATED LEARNING IS A BOOMING TOPIC

- 2016: the term FL is first coined by Google researchers; 2020: more than 1,000 papers on FL in the first half of the year (compared to just 180 in 2018)[1]
- We have already seen some real-world deployments by companies and researchers
- Several open-source libraries are under development: PySyft, TensorFlow Federated, FATE, Flower, Substra...
- FL is highly multidisciplinary: it involves machine learning, numerical optimization, privacy & security, networks, systems, hardware...

This is all a bit hard to keep up with!

[1] https://www.forbes.com/sites/robtoews/2020/10/12/the-next-generation-of-artificial-intelligence/

A baseline algorithm: FedAvg

- We consider a set of K parties (also called users or clients)
- Each party k holds a dataset $\mathcal{D}_k$ of n_k points
- Let $\mathcal{D} = \mathcal{D}_1 \cup \cdots \cup \mathcal{D}_K$ be the joint dataset and $n = \sum_k n_k$ the total number of points
- We want to solve problems of the form $\min_{\theta \in \mathbb{R}^p} F(\theta; \mathcal{D})$ where:

$$F(\theta; \mathcal{D}) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(\theta; \mathcal{D}_k) \quad \text{and} \quad F_k(\theta; \mathcal{D}_k) = \frac{1}{n_k} \sum_{d \in \mathcal{D}_k} f(\theta; d)$$

- $\theta \in \mathbb{R}^p$ are model parameters (e.g., weights of a logistic regression or neural network)
- This covers a broad class of ML problems formulated as empirical risk minimization

FEDAVG (AKA LOCAL SGD) [McMahan et al., 2017]

Algorithm FedAvg (server-side)

Parameters: client sampling rate ρ

initialize θ
for each round $t = 0, 1, \dots$ **do**
  $\mathcal{S}_t \leftarrow$ random set of $m = \lceil \rho K \rceil$ clients
  for each client $k \in \mathcal{S}_t$ in parallel **do**
    $\theta_k \leftarrow \text{ClientUpdate}(k, \theta)$
  $\theta \leftarrow \sum_{k \in \mathcal{S}_t} \frac{n_k}{n} \theta_k$

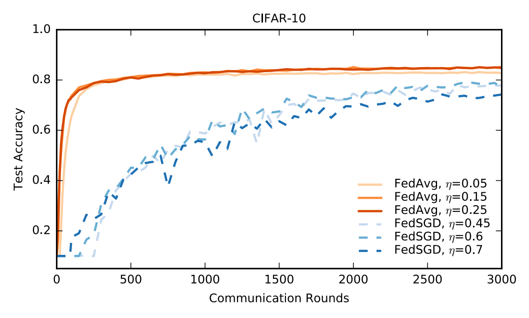
Algorithm ClientUpdate(k, θ)

Parameters: batch size B, number of local steps L, learning rate η

for each local step $1, \dots, L$ **do**
  $\mathcal{B} \leftarrow$ mini-batch of B examples from $\mathcal{D}_k$
  $\theta \leftarrow \theta - \frac{1}{B} \eta \sum_{d \in \mathcal{B}} \nabla f(\theta; d)$
send θ to server

- For $L = 1$ and $\rho = 1$, it is equivalent to classic parallel SGD: updates are aggregated and the model synchronized at each step
- For $L > 1$: each client performs multiple local SGD steps before communicating

FEDAVG (AKA LOCAL SGD) [McMahan et al., 2017]



- FedAvg with $L > 1$ allows to reduce the number of communication rounds, which is often the bottleneck in FL (especially in the cross-device setting)
- It empirically achieves better generalization than parallel SGD with large mini-batch
- Convergence to the optimal model can be guaranteed for IID data [Stich, 2019] [Woodworth et al., 2020] but issues arise in strongly non-IID case (more on this later)

FULLY DECENTRALIZED SETTING

- We can derive algorithms similar to FedAvg for the fully decentralized setting, where parties do not rely on a server for aggregating updates
- Let $G = (\{1, \dots, K\}, E)$ be a connected undirected graph where nodes are parties and an edge $\{k, l\} \in E$ indicates that k and l can exchange messages
- Let $W \in [0, 1]^{K \times K}$ be a symmetric, doubly stochastic matrix such that $W_{k,l} = 0$ if and only if $\{k, l\} \notin E$
- Given models $\Theta = [\theta_1, \dots, \theta_K]$ for each party, $W\Theta$ corresponds to a weighted aggregation among neighboring nodes in G:

$$[W\Theta]_k = \sum_{l \in \mathcal{N}_k} W_{k,l}\theta_l, \quad \text{where } \mathcal{N}_k = \{l : \{k, l\} \in E\}$$

Algorithm Fully decentralized SGD (run by party k)

Parameters: batch size B, learning rate η, sequence of matrices $W^{(t)}$

initialize $\theta_k^{(0)}$

for each round $t = 0, 1, \dots$ **do**

$\mathcal{B} \leftarrow$ mini-batch of B examples from $\mathcal{D}_k$

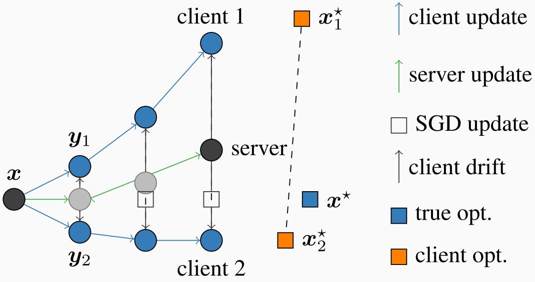
$\theta_k^{(t+\frac{1}{2})} \leftarrow \theta_k^{(t)} - \frac{1}{B}\eta \sum_{d\in\mathcal{B}} \nabla f(\theta_k^{(t)}; d)$

$\theta_k^{(t+1)} \leftarrow \sum_{l\in\mathcal{N}_k^{(t)}} W_{k,l}^{(t)} \theta_l^{(t+\frac{1}{2})}$

- Decentralized SGD alternates between local updates and local aggregation
- Doing multiple local steps is equivalent to choosing $W^{(t)} = I_n$ in some of the rounds
- The convergence rate depends on the topology (the more connected, the faster)

Challenge 1: Dealing with non-iid data

CLIENT DRIFT IN FEDAVG



(Figure taken from [Karimireddy et al., 2020])

- When local datasets are non-IID, FedAvg suffers from client drift
- To avoid this drift, one must use fewer local updates and/or smaller learning rates, which hurts convergence

- Analyzing the convergence rate of FL algorithms on non-IID data involves some assumption about how the local cost functions $F_1, \dots, F_k$ are related
- For instance, one can assume that there exists constants $G \geq 0$ and $B \geq 1$ such that

$$\forall \theta : \quad \frac{1}{K} \sum_{k=1}^{K} \|\nabla F_k(\theta; \mathcal{D}_k)\|^2 \leq G^2 + B^2 \|\nabla F(\theta; \mathcal{D})\|^2$$

- FedAvg without client sampling reaches ϵ accuracy with $O(\frac{1}{KL\epsilon^2} + \frac{G}{\epsilon^{3/2}} + \frac{B^2}{\epsilon})$, which is slower than the $O(\frac{1}{KL\epsilon^2} + \frac{1}{\epsilon})$ of parallel SGD with large batch [Karimireddy et al., 2020]

Algorithm Scaffold (server-side)

Parameters: client sampling rate ρ, global learning rate η_g

initialize θ, $c = c_1, \ldots, c_K = 0$

for each round $t = 0, 1, \ldots$ **do**

  $\mathcal{S}_t \leftarrow$ random set of $m = \lceil \rho K \rceil$ clients

  for each client $k \in \mathcal{S}_t$ in parallel **do**

    $(\Delta\theta_k, \Delta c_k) \leftarrow$ ClientUpdate (k, θ, c)

  $\theta \leftarrow \theta + \frac{\eta_g}{m} \sum_{k \in \mathcal{S}_t} \Delta\theta_k$

  $c \leftarrow c + \frac{1}{K} \sum_{k \in \mathcal{S}_t} \Delta c_k$

Algorithm ClientUpdate(k, θ, c)

Parameters: batch size B, # of local steps L, local learning rate η_l

Initialize $\theta_k \leftarrow \theta$

for each local step $1, \ldots, L$ **do**

  $\mathcal{B} \leftarrow$ mini-batch of B examples from $\mathcal{D}_k$

  $\theta_k \leftarrow \theta_k - \eta_l(\frac{1}{B} \sum_{d \in \mathcal{B}} \nabla f(\theta; d) - c_k + c)$

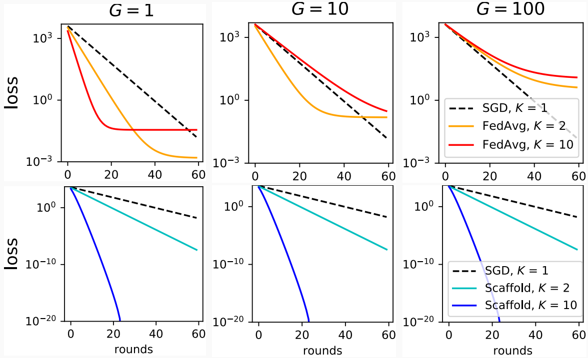
$c_k^+ \leftarrow c_k - c + \frac{1}{L\eta_l}(\theta - \theta_k)$

send $(\theta_k - \theta, c_k^+ - c_k)$ to server

$c_k \leftarrow c_k^+$

- Correction terms $c_1, \ldots, c_K$ are a form of variance reduction (cf Aymeric's tutorial)
- Can show convergence rates which beat parallel SGD

SCAFFOLD: CORRECTING LOCAL UPDATES [Karimireddy et al., 2020]



- FedAvg becomes slower than parallel SGD for strongly non-IID data (large G)
- Scaffold can often do better in such settings
- Other relevant approach: FedProx [Li et al., 2020b]

- Learning from non-IID data is difficult/slow because each party wants the model to go in a particular direction
- If data distributions are very different, learning a single model which performs well for all parties may require a very large number of parameters
- Another direction to deal with non-IID data is thus to lift the requirement that the learned model should be the same for all parties ("one size fits all")
- Instead, we can allow each party k to learn a (potentially simpler) personalized model θ_k but design the objective so as to enforce some kind of collaboration

PERSONALIZED MODELS FROM A "META" MODEL

- [Hanzely et al., 2020] propose to regularize personalized models to their mean:

$$F(\theta_1, \ldots, \theta_K; \mathcal{D}) = \frac{1}{K} \sum_{k=1}^{K} F_k(\theta_k; \mathcal{D}_k) + \frac{\lambda}{2K} \sum_{k=1}^{K} \left\| \theta_k - \frac{1}{K} \sum_{l=1}^{K} \theta_l \right\|^2$$

- Inspired by meta-learning, [Fallah et al., 2020] propose to learn a global model which easily adapts to each party:

$$F(\theta; \mathcal{D}) = \frac{1}{K} \sum_{k=1}^{K} F_k(\theta - \alpha \nabla F_k(\theta); \mathcal{D}_k)$$

- These formulations are actually related to each other (and to the FedAvg algorithm)
- Other formulations exist, see e.g., the bilevel approach of [Dinh et al., 2020]

Challenge 1: Dealing with non-iid data

Zoom on learning personalized models via task relationships

- Inspired by multi-task learning, [Smith et al., 2017, Vanhaesebrouck et al., 2017] propose to regularize personalized models using (learned) relationships between tasks
- Learn personalized models $\Theta \in \mathbb{R}^{K \times p}$ and graph weights $w \in \mathbb{R}_{\geq 0}^{K(K-1)/2}$ as solutions to

$$\min_{\Theta \in \mathbb{R}^{K \times p}, w \in \mathbb{R}_{\geq 0}^{K(K-1)/2}} J(\Theta, w) = \sum_{k=1}^{K} d_k c_k F_k(\theta_k; \mathcal{D}_k) + \frac{\mu}{2} \sum_{k<l} w_{kl} \|\theta_k - \theta_l\|^2 + \lambda g(w),$$

- Trade-off between learning accurate models on local data and learning similar models for similar parties
- $c_k \in (0, 1] \propto n_k/n$: confidence of party k, $d_k = \sum_{l \neq k} w_{kl}$: degree of k
- Graph regularizer $g(w)$: avoid trivial graph, encourage sparsity
- Flexible relationships: hyperparameter $\mu \geq 0$ interpolates between learning purely local models and a shared model per connected component

ALGORITHM BASED ON ALTERNATING OPTIMIZATION

We design an alternating optimization procedure over Θ and w:

1. A federated algorithm to learn the models given the graph
2. A federated algorithm to learn a graph given the models

FULLY DECENTRALIZED SETTING

- Asynchronous time model: each party becomes active at random times, asynchronously and in parallel (we use global counter t to denote the t-th activation)
- Communication model: all parties can exchange messages, but we want to restrict communication to pairs of most similar parties
- We use the (current) relationship graph as a communication overlay: party k only send messages to her neighbors $\mathcal{N}(k) = \{l : w_{kl} > 0\}$

- Initialize models $\Theta(0) \in \mathbb{R}^{K \times p}$, choose learning rate $\alpha \in (0, 1)$
- At step $t \geq 0$, a random party k becomes active:
  1. party k updates its model based on its local dataset $\mathcal{D}_k$ and neighbors' models:

$$\theta_k(t+1) = (1-\alpha)\theta_k(t) + \alpha\Big(\sum_{l \in \mathcal{N}(k)} \frac{W_{kl}}{d_k}\theta_l(t) - \frac{c_k}{\mu}\nabla F_k(\theta_k(t); \mathcal{D}_k)\Big)$$

  2. party k sends its updated model $\theta_k(t+1)$ to its neighborhood $\mathcal{N}(k)$
- The update is a combination of a local gradient descent step and a weighted average of neighbors' models

Proposition ([Bellet et al., 2018])

For any $T > 0$, let $(\Theta(t))_{t=1}^{T}$ be the sequence of iterates generated by the algorithm running for T iterations from an initial point $\Theta(0)$. When the local losses $F_1, \ldots, F_K$ are strongly convex, for appropriate choice of α, we have:

$$\mathbb{E}\left[f(\Theta(T)) - f^\star\right] \leq \left(1 - \frac{\sigma}{KL_{max}}\right)^{T} \left(f(\Theta(0)) - f^\star\right).$$

where L_{max} and σ are global smoothness and strong convexity parameters.

- Optimality gap decreases exponentially fast with T
- Constant number of per-party updates $\rightarrow$ optimality gap roughly constant in K
- Note: can prove $O(1/T)$ convergence for the standard convex case

CHOICE OF GRAPH REGULARIZER

$$\min_{\Theta \in \mathbb{R}^{K \times p}, w \in \mathbb{R}_{\geq 0}^{K(K-1)/2}} J(\Theta, w) = \sum_{k=1}^{K} d_k c_k F_k(\theta_k; \mathcal{D}_k) + \frac{\mu}{2} \sum_{k<l} w_{kl} \|\theta_k - \theta_l\|^2 + \lambda g(w),$$

- Our algorithm can deal with a large family of functions g
- Inspired by [Kalofolias, 2016], we can use

$$g(w) = \beta \|w\|^2 - 1^T \log(d + \delta) \quad \text{(with } \delta \text{ small constant)}$$

- Log barrier on the degree vector d to avoid isolated parties and L_2 penalty on weights to control the graph sparsity
- The resulting objective h in w is strongly convex

LEARNING THE GRAPH GIVEN THE MODELS

- We rely on decentralized peer sampling [Jelasity et al., 2007] to allow parties to communicate with a set of κ random peers
- Initialize weights $w(0)$, choose parameter $\kappa \in \{1, \dots, K-1\}$
- At each step $t \geq 0$, a random party k becomes active:
  1. Draw a set $\mathcal{K}$ of κ parties and request their model, loss and degree
  2. Update the associated weights $w(t+1)_{k,\mathcal{K}}$ via a gradient update
  3. Send each updated weight $w(t+1)_{kl}$ to the associated party $l \in \mathcal{K}$

Theorem ([Zantedeschi et al., 2020])

For any $T > 0$, let $(w(t))_{t=1}^{T}$ be the sequence of iterates generated by the algorithm running for T iterations from an initial point $w(0)$. We have:

$$\mathbb{E}[h(w^{(T)}) - h^*] \leq \rho^T (h(w^{(0)}) - h^*), \quad \textit{where } \rho = 1 - \frac{4}{K(K-1)} \frac{\kappa\beta\delta^2}{\kappa + 1 + 2\beta\delta^2}$$

- κ can be used to trade-off between communication cost and convergence speed

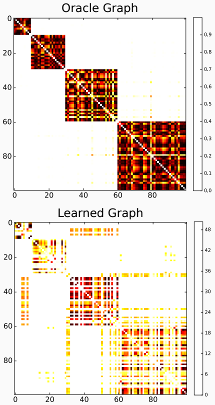
- Consider a set of base models $h_1, \ldots, h_M$ (e.g., pre-trained on proxy data)
- Find personalized ensemble models $x \mapsto \text{sign}(\sum_{m=1}^{M} [\theta_k]_m h_m(x))$ as solutions to:

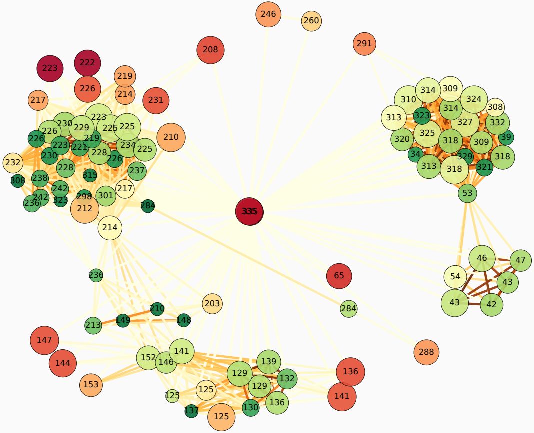
$$\min_{\substack{\|\theta_1\|_1 \leq \beta, \ldots, \|\theta_K\|_1 \leq \beta \\ w \in \mathbb{R}_{\geq 0}^{K(K-1)/2}}} \sum_{k=1}^{K} d_k c_k \underbrace{\log\Big(\sum_{l=1}^{n_k} \exp\big(-(A_k\theta_k)_l\big)\Big)}_{F_k(\theta_k;\mathcal{D}_k)} + \frac{\mu}{2} \sum_{k<l} w_{kl} \|\theta_k - \theta_l\|^2 + \lambda g(w)$$

- $A_k \in \mathbb{R}^{n_k \times M}$: margins of base models on each data point of party k
- Can design algorithm with communication cost logarithmic in K

ILLUSTRATION ON SYNTHETIC DATA

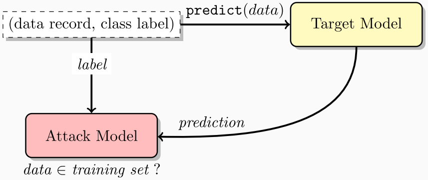
- We approximately recover the ground-truth cluster structure
- Prediction accuracy is close to that of the “oracle” graph



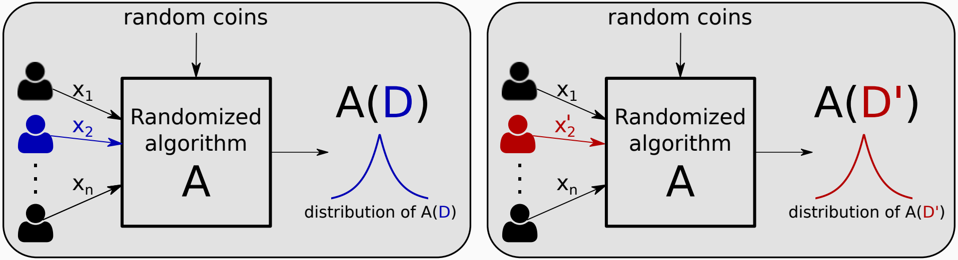


Challenge 2: Preserving privacy

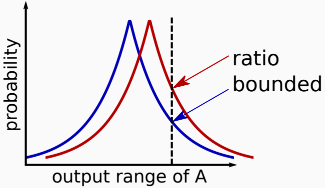
- ML models are susceptible to various attacks on data privacy
- Membership inference attacks try to infer the presence of a known individual in the training set, e.g., by exploiting the confidence in model predictions [Shokri et al., 2017]



- Reconstruction attacks try to infer some of the points used to train the model, e.g., by differencing attacks [Paige et al., 2020]
- Federated Learning offers an additional attack surface because the server and/or other clients observe model updates (not only the final model) [Nasr et al., 2019]



- Neighboring datasets $\mathcal{D} = \{x_1, x_2, \ldots, x_n\}$ and $\mathcal{D}' = \{x_1, x_2', x_3, \ldots, x_n\}$
- **Requirement**: $\mathcal{A}(\mathcal{D})$ and $\mathcal{A}(\mathcal{D}')$ should have "close" distribution



Definition ([Dwork et al., 2006], informal)

A randomized algorithm $\mathcal{A}$ is (ε, δ)-differentially private (DP) if for all neighboring datasets $\mathcal{D} = \{x_1, x_2, \ldots, x_n\}$ and $\mathcal{D}' = \{x_1, x_2', x_3, \ldots, x_n\}$ and all sets S:

$$\Pr[\mathcal{A}(\mathcal{D}) \in S] \leq e^{\varepsilon} \Pr[\mathcal{A}(\mathcal{D}') \in S] + \delta.$$

- First proposed in [Dwork et al., 2006] (who won the Gödel prize in 2017)
- Key principle: privacy is a property of the analysis, not of a particular output (in contrast to e.g., k-anonymity)
- For meaningful privacy guarantees, think of $\varepsilon \leq 1$ and $\delta \ll 1/n$

- DP is immune to post-processing: it is impossible to compute a function of the output of the private algorithm and make it less differentially private
- DP is robust to arbitrary auxiliary knowledge (worst-case model): the guarantee is just as strong if the adversary knows all but one record and regardless of the adversary strategy and computational power
- DP is robust under composition: if multiple analyses are performed on the same data, as long as each one satisfies DP, all the information released taken together will still satisfy DP (albeit with a degradation in the parameters)

- Consider f taking as input a dataset and returning a p-dimensional real vector

Gaussian mechanism $\mathcal{A}_{\text{Gauss}}(\mathcal{D}, f, \epsilon, \delta)$

1. Compute sensitivity $\Delta = \max_{(\mathcal{D},\mathcal{D}') \text{ are neighboring}} \|f(\mathcal{D}) - f(\mathcal{D}')\|_2$
2. For $i = 1, \dots, p$: draw $Y_i \sim \mathcal{N}(0, \sigma^2)$ independently for each i, where $\sigma = \frac{\sqrt{2\ln(1.25/\delta)}\Delta}{\varepsilon}$
3. Output $f(\mathcal{D}) + Y$, where $Y = (Y_1, \dots, Y_p) \in \mathbb{R}^p$

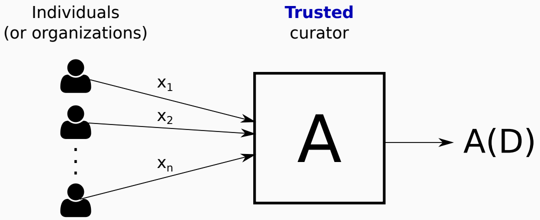
Theorem

Let $\varepsilon, \delta > 0$. The Gaussian mechanism $\mathcal{A}_{Gauss}(\cdot, f, \varepsilon, \delta)$ is (ϵ, δ)-DP.

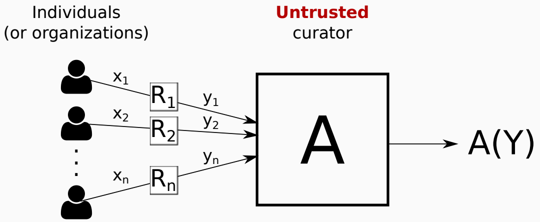
- Noise calibrated using sensitivity of f and privacy budget (ε and δ)
- Induces a clear privacy-utility trade-off

TWO SETTINGS FOR DP: CENTRALIZED VS DECENTRALIZED

Centralized setting (also called global setting or trusted curator setting): $\mathcal{A}$ is differentially private wrt dataset $\mathcal{D}$



Decentralized/federated setting (also called local setting or untrusted curator setting): each $\mathcal{R}_k$ is DP wrt record x_k (or local dataset $\mathcal{D}_k$)



- Most (server-orchestrated) FL algorithms follow the same high-level pattern:

  for $t = 1$ to T **do**
  At each party k: compute $\theta_k \leftarrow \textsc{LocalUpdate}(\theta, \theta_k)$, send θ_k to server
  At server: compute $\theta \leftarrow \frac{1}{K}\sum_k \theta_k$, send θ back to the parties

- Therefore:

$$\text{DP aggregation} \; + \; \text{Composition property of DP} \implies \text{DP-FL}$$

- **Differentially private aggregation:** given a private value $\theta_k \in \mathbb{R}$ (computed from $\mathcal{D}_k$) for each party k, we want to accurately estimate $\theta^{avg} = \frac{1}{K}\sum_k \theta_k$ under a DP constraint

EXISTING APPROACHES TO DP AGGREGATION

- Centralized setting: trusted curator adds (Gaussian) noise to the average θ^{avg}
- Decentralized setting: each party k adds noise to its own value θ_k before sharing it
- For a fixed DP guarantee, the error is $O(\sqrt{K})$ larger in the decentralized case!
- Cryptographic primitives such as secure aggregation [Bonawitz et al., 2017] and secure shuffling [Balle et al., 2019] can be used to close this gap
- However their practical implementation poses important challenges when K is large

Challenge 2: Preserving privacy

Zoom on an accurate and scalable protocol for private aggregation

Algorithm GOPA protocol

Parameters: graph G, variances $\sigma_\Delta^2, \sigma_\eta^2 \in \mathbb{R}^+$

for all neighboring parties $\{k, l\}$ in G **do**
  k and l draw $y \sim \mathcal{N}(0, \sigma_\Delta^2)$
  set $\Delta_{k,l} \leftarrow y$, $\Delta_{l,k} \leftarrow -y$
for each party k **do**
  k draws $\eta_k \sim \mathcal{N}(0, \sigma_\eta^2)$
  k reveals $\hat{\theta}_k \leftarrow \theta_k + \sum_{l \sim k} \Delta_{k,l} + \eta_k$

1. Neighbors $\{k, l\}$ in G securely exchange pairwise-canceling Gaussian noise
2. Each party k generate independent Gaussian noise
3. Party k reveals the sum of private value, pairwise and independent noise terms

- **Unbiased estimate of the average:** $\hat{\theta}^{avg} = \frac{1}{K} \sum_k \hat{\theta}_k$, with variance σ_η^2 / K

- **Adversary**: proportion $1 - \rho$ of colluding malicious parties who observe all communications they take part in
- Denote by U^H the set of honest-but-curious parties, and by G^H the honest subgraph
- GOPA can achieve (ϵ, δ)-DP for any $\epsilon, \delta > 0$ for connected G^H and large enough $\sigma_\eta^2, \sigma_\Delta^2$
- We show that σ_η^2 can be as small as in the centralized setting (matching its utility)
- We show that the required σ_Δ^2 depends on the topology of G^H

Theorem (Case of random k-out graph)

Let $\epsilon, \delta' \in (0, 1)$ and let:

- *G be obtained by letting all parties randomly choose $m = O(\log(\rho n)/\rho)$ neighbors*
- *σ_η^2 so as to satisfy (ϵ, δ)-DP in the centralized (trusted curator) setting*
- $\sigma_\Delta^2 = O(\sigma_\eta^2 |U^H|/m)$

Then GOPA is (ϵ, δ)-differentially private for $\delta = O(\delta')$.

- Trusted curator utility with logarithmic number of messages per party
- Our theoretical results give practical values for m and σ_Δ^2

- Utility can be compromised by malicious parties tampering with the protocol (e.g., sending incorrect values to bias the outcome)
- It is impossible to force a party k to give the "right" input θ_k (this also holds in the trusted curator setting)
- We enable each party k to prove the following properties:

$$
\begin{aligned}
\theta_k &\in [0,1], && \forall k \in \{1, \dots, K\} \\
\Delta_{k,l} &= -\Delta_{l,k}, && \forall \{k,l\} \text{ neighbors in } G \\
\eta_k &\sim \mathcal{N}(0, \sigma_\eta^2), && \forall k \in \{1, \dots, K\} \\
\hat{\theta}_k &= \theta_k + \sum_{l \sim k} \Delta_{k,l} + \eta_k, && \forall k \in \{1, \dots, K\}
\end{aligned}
$$

ENSURING CORRECTNESS

- Parties publish an encrypted log of the computation using Pedersen commitments [Blum, 1983, Pedersen, 1991], which are additively homomorphic
- Based on these commitments, parties prove that the computation was done correctly using zero knowledge proofs

Theorem (Informal)

A party k that passes the verification proves that $\hat{\theta}_k$ was computed correctly. Additionally, by doing so, k does not reveal any additional information about θ_k.

- Costs per party remain linear in the number of neighbors
- Can prove consistency across multiple runs on same/similar data
- Can handle drop out

Wrapping up

SOME OTHER INTERESTING TOPICS IN FL

- Going beyond empirical risk minimization formulations: tree-based methods [Li et al., 2020a], online learning [Dubey and Pentland, 2020], Bayesian learning...
- Vertical data partitioning, where parties have access to different features about the same examples [Patrini et al., 2016]
- Compressing updates to reduce communication [Koloskova et al., 2020a]
- Fairness in FL [Mohri et al., 2020, Li et al., 2020c, Laguel et al., 2020]
- Security in FL: how to mitigate poisoning attacks [Bagdasaryan et al., 2020] [Blanchard et al., 2017]

Survey paper: Advances and Open Problems in FL [Kairouz et al., 2021]

- A large collaborative effort (50+ authors!)
- Updated in December 2020, to appear in FnTML 2021

Online seminar: Federated Learning One World (FLOW)

`https://sites.google.com/view/one-world-seminar-series-flow/`

- Co-organized with D. Alistarh, V. Smith and P. Richtárik, started in May 2020
- Weekly talks (usually on Wednesdays, 1pm UTC) covering all aspects of FL
- The videos and slides of all previous talks are available online

REFERENCES I

[Bagdasaryan et al., 2020] Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., and Shmatikov, V. (2020).
How To Backdoor Federated Learning.
In *AISTATS*.

[Balle et al., 2018] Balle, B., Barthe, G., and Gaboardi, M. (2018).
Privacy Amplification by Subsampling: Tight Analyses via Couplings and Divergences.
In *NeurIPS*.

[Balle et al., 2019] Balle, B., Bell, J., Gascón, A., and Nissim, K. (2019).
The Privacy Blanket of the Shuffle Model.
In *CRYPTO*.

[Bellet et al., 2018] Bellet, A., Guerraoui, R., Taziki, M., and Tommasi, M. (2018).
Personalized and Private Peer-to-Peer Machine Learning.
In *AISTATS*.

[Blanchard et al., 2017] Blanchard, P., Mhamdi, E. M. E., Guerraoui, R., and Stainer, J. (2017).
Machine learning with adversaries: Byzantine tolerant gradient descent.
In *NIPS*.

[Blum, 1983] Blum, M. (1983).
Coin flipping by telephone a protocol for solving impossible problems.
ACM SIGACT News, 15(1):23–27.

[Bonawitz et al., 2017] Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., Ramage, D., Segal, A., and Seth, K. (2017).
Practical Secure Aggregation for Privacy-Preserving Machine Learning.
In *CCS*.

[Cyffers and Bellet, 2020] Cyffers, E. and Bellet, A. (2020).
Privacy Amplification by Decentralization.
Technical report, arXiv:2012.05326.

[Dinh et al., 2020] Dinh, C. T., Tran, N. H., and Nguyen, T. D. (2020).
Personalized Federated Learning with Moreau Envelopes.
In *NeurIPS*.

[Dubey and Pentland, 2020] Dubey, A. and Pentland, A. S. (2020).
Differentially-Private Federated Linear Bandits.
In *NeurIPS*.

[Dwork et al., 2006] Dwork, C., McSherry, F., Nissim, K., and Smith, A. (2006).
Calibrating noise to sensitivity in private data analysis.
In *Theory of Cryptography (TCC)*.

[Erlingsson et al., 2019] Erlingsson, U., Feldman, V., Mironov, I., Raghunathan, A., and Talwar, K. (2019).
Amplification by Shuffling: From Local to Central Differential Privacy via Anonymity.
In *SODA*.

[Fallah et al., 2020] Fallah, A., Mokhtari, A., and Ozdaglar, A. (2020).
Personalized Federated Learning with Theoretical Guarantees: A Model-Agnostic Meta-Learning Approach.
In *NeurIPS*.

[Feldman et al., 2018] Feldman, V., Mironov, I., Talwar, K., and Thakurta, A. (2018).
Privacy Amplification by Iteration.
In *FOCS*.

[Hanzely et al., 2020] Hanzely, F., Hanzely, S., Horváth, S., and Richtarik, P. (2020).
Lower Bounds and Optimal Algorithms for Personalized Federated Learning.
In *NeurIPS*.

[Jelasity et al., 2007] Jelasity, M., Voulgaris, S., Guerraoui, R., Kermarrec, A.-M., and van Steen, M. (2007).
Gossip-based peer sampling.
ACM Trans. Comput. Syst., 25(3).

[Kairouz et al., 2021] Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Eichner, H., Rouayheb, S. E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konecný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Qi, H., Ramage, D., Raskar, R., Raykova, M., Song, D., Song, W., Stich, S. U., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F. X., Yu, H., and Zhao, S. (2021).
Advances and Open Problems in Federated Learning.
Foundations and Trends® in Machine Learning, 14(1–2):1–210.

[Kairouz et al., 2015] Kairouz, P., Oh, S., and Viswanath, P. (2015).
The Composition Theorem for Differential Privacy.
In *ICML*.

[Kalofolias, 2016] Kalofolias, V. (2016).
How to learn a graph from smooth signals.
In *AISTATS*.

[Karimireddy et al., 2020] Karimireddy, S. P., Kale, S., Mohri, M., Reddi, S. J., Stich, S. U., and Suresh, A. T. (2020).
SCAFFOLD: Stochastic Controlled Averaging for On-Device Federated Learning.
In *ICML*.

[Koloskova et al., 2020a] Koloskova, A., Lin, T., Stich, S. U., and Jaggi, M. (2020a).
Decentralized Deep Learning with Arbitrary Communication Compression.
In *ICLR*.

[Koloskova et al., 2020b] Koloskova, A., Loizou, N., Boreiri, S., Jaggi, M., and Stich, S. U. (2020b).
A Unified Theory of Decentralized SGD with Changing Topology and Local Updates.
In *ICML*.

[Laguel et al., 2020] Laguel, Y., Pillutla, K., Malick, J., and Harchaoui, Z. (2020).
Device Heterogeneity in Federated Learning:A Superquantile Approach.
Technical report, arXiv:2002.11223.

[Li et al., 2020a] Li, Q., Wen, Z., and He, B. (2020a).
Practical Federated Gradient Boosting Decision Trees.
In *AAAI*.

[Li et al., 2020b] Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. (2020b).
Federated Optimization in Heterogeneous Networks.
In *MLSys*.

[Li et al., 2020c] Li, T., Sanjabi, M., Beirami, A., and Smith, V. (2020c).
Fair Resource Allocation in Federated Learning.
In *ICLR*.

[Lian et al., 2017] Lian, X., Zhang, C., Zhang, H., Hsieh, C.-J., Zhang, W., and Liu, J. (2017).
Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent.
In *NIPS*.

[Mao et al., 2020] Mao, X., Yuan, K., Hu, Y., Gu, Y., Sayed, A. H., and Yin, W. (2020).
Walkman: A Communication-Efficient Random-Walk Algorithm for Decentralized Optimization.
IEEE Transactions on Signal Processing, 68:2513–2528.

[McMahan et al., 2017] McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and Agüera y Arcas, B. (2017).
Communication-efficient learning of deep networks from decentralized data.
In *AISTATS*.

[Mohri et al., 2020] Mohri, M., Sivek, G., and Suresh, A. T. (2020).
Agnostic Federated Learning.
In *ICML*.

[Nasr et al., 2019] Nasr, M., Shokri, R., and Houmansadr, A. (2019).
Comprehensive Privacy Analysis of Deep Learning: Passive and Active White-box Inference Attacks against Centralized and Federated Learning.
In *IEEE Symposium on Security and Privacy*.

[Paige et al., 2020] Paige, B., Bell, J., Bellet, A., Gascón, A., and Ezer, D. (2020).
Reconstructing Genotypes in Private Genomic Databases from Genetic Risk Scores.
In *International Conference on Research in Computational Molecular Biology RECOMB*.

[Patrini et al., 2016] Patrini, G., Nock, R., Hardy, S., and Caetano, T. S. (2016).
Fast Learning from Distributed Datasets without Entity Matching.
In *IJCAI*.

[Pedersen, 1991] Pedersen, T. P. (1991).
Non-interactive and information-theoretic secure verifiable secret sharing.
In *CRYPTO*.

[Ram et al., 2009] Ram, S., Nedić, A., and Veeravalli, V. (2009).
Incremental stochastic subgradient algorithms for convex optimization.
SIAM Journal on Optimization, 20(2):691–717.

[Sabater et al., 2020] Sabater, C., Bellet, A., and Ramon, J. (2020).
Distributed Differentially Private Averaging with Improved Utility and Robustness to Malicious Parties.
Technical report, arXiv:2006.07218.

[Shokri et al., 2017] Shokri, R., Stronati, M., Song, C., and Shmatikov, V. (2017).
Membership Inference Attacks Against Machine Learning Models.
In *IEEE Symposium on Security and Privacy (S&P)*.

[Smith et al., 2017] Smith, V., Chiang, C.-K., Sanjabi, M., and Talwalkar, A. S. (2017).
Federated Multi-Task Learning.
In *NIPS*.

[Stich, 2019] Stich, S. U. (2019).
Local SGD Converges Fast and Communicates Little.
In *ICLR*.

[Vanhaesebrouck et al., 2017] Vanhaesebrouck, P., Bellet, A., and Tommasi, M. (2017).
Decentralized collaborative learning of personalized models over networks.
In *AISTATS*.

[Woodworth et al., 2020] Woodworth, B., Patel, K. K., Stich, S. U., Dai, Z., Bullins, B., McMahan, H. B., Shamir, O., and Srebro, N. (2020).
Is Local SGD Better than Minibatch SGD?
In *ICML*.

[Zantedeschi et al., 2020] Zantedeschi, V., Bellet, A., and Tommasi, M. (2020).
Fully Decentralized Joint Learning of Personalized Models and Collaboration Graphs.
In *AISTATS*.

Bonus 1: Private learning of personalized models

- In our algorithms, parties never communicate their local data but they exchange sequences of models computed from data
- We consider an adversary observing all the information sent over the network (but not the internal memory of parties)
- **Goal:** modify algorithm to satisfy differential privacy

1. Replace the update of the algorithm for learning personalized models by

$$\widetilde{\theta}_k(t+1) = (1-\alpha)\widetilde{\theta}_k(t) + \alpha\Big(\sum_{l \in \mathbb{N}_k} \frac{W_{kl}}{d_k}\widetilde{\theta}_l(t) - \frac{c_k}{\mu}\big(\nabla F_k(\widetilde{\theta}_k(t); \mathcal{D}_k) + \eta_k(t)\big)\Big),$$

   where $\eta_k \sim \text{Laplace}(0, s_i)^p \in \mathbb{R}^p$
2. Party k then broadcasts noisy iterate $\widetilde{\theta}_k(t+1)$ to its neighbors

PRIVACY GUARANTEE

Theorem ([Bellet et al., 2018])

Let $k \in \{1, \dots, K\}$ and assume

- *The loss function is L_0-Lipschitz w.r.t. the L_1-norm for all data points*
- *Party k wakes up T_k times and use noise scale $s_k = \frac{L_0}{\epsilon_k n_k}$*
- *Algorithm $\mathcal{A}_k(\mathcal{D}_k)$: releases the sequence of party k's models*

For any $\widetilde{\Theta}(0)$ independent of $\mathcal{D}_k$, $\mathcal{A}_k(\mathcal{D}_k)$ is $\bar{\epsilon}_k$-DP with $\bar{\epsilon}_k = T_k \epsilon_k$.

- Follows from (L1) sensitivity analysis of the update and Laplace mechanism
- Can be improved by strong composition [Kairouz et al., 2015]

Theorem ([Bellet et al., 2018])

For any $T > 0$, let $(\Theta(t))_{t=1}^{T}$ be the sequence of iterates generated by T iterations. For σ-strongly convex f, we have:

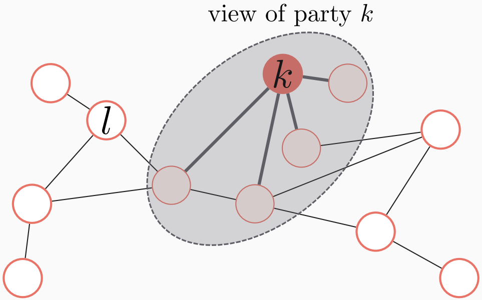
$$\mathbb{E}\left[f(\widetilde{\Theta}(T)) - f^\star\right] \leq \left(1 - \frac{\sigma}{nL_{max}}\right)^T \left(f(\widetilde{\Theta}(0)) - f^\star\right) + \frac{1}{KL_{min}} \sum_{t=0}^{T-1} \sum_{k=1}^{K} \left(1 - \frac{\sigma}{KL_{max}}\right)^t \left[d_k c_k s_k(t)\right]^2,$$

where L_{min} and L_{max} are smoothness parameters.

- Parties with less data add more noise but their contribution to the error is smaller
- T rules a trade-off between optimization error and noise error
- A good (differentially private) warm start can help a lot

Bonus 2: Privacy benefits of full decentralization

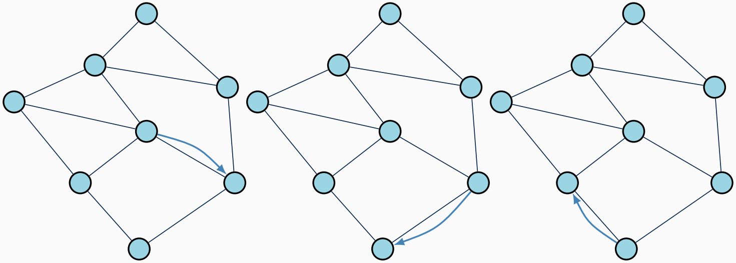
PRIVACY BENEFITS OF FULL DECENTRALIZATION [Cyffers and Bellet, 2020]



- In the fully decentralized case, each party has a limited view of the system
- Can this be used to prove stronger differential privacy guarantees?

PRIVACY BENEFITS OF FULL DECENTRALIZATION [Cyffers and Bellet, 2020]

- Consider algorithms that sequentially update the estimate (e.g., ML model) by following a walk over the network graph [Ram et al., 2009, Mao et al., 2020]



- We have shown that for some topologies (directed ring, complete graph), such algorithms can match the privacy-utility trade-off of the centralized setting
- Analysis relies on recent privacy amplification results [Balle et al., 2018] [Erlingsson et al., 2019, Feldman et al., 2018]