PRIVACY-PRESERVING MACHINE LEARNING

Aurélien Bellet (Inria)

Peyresq Summer School 2022

ABOUT MYSELF

- Researcher at Inria (Lille Nord-Europe research center)
- Member of the Magnet team (Machine learning in information networks) on ML in/with graphs and applications to NLP
- My current research topics:
    - Privacy-preserving ML
    - Federated ML
    - Representation learning for speech and natural language processing
    - Fairness in ML
- More details on my homepage

OUTLINE OF THE COURSE

1. Context & Motivation
2. Differential Privacy
3. The Gaussian Mechanism
4. Differentially Private SGD
5. Introduction to Federated Learning
6. Differentially Private Federated Learning

Context & Motivation

Ability of an individual
to seclude themselves or to **withhold information about themselves**

(“right to be let alone”)

PRIVACY IN THE BIG DATA ERA

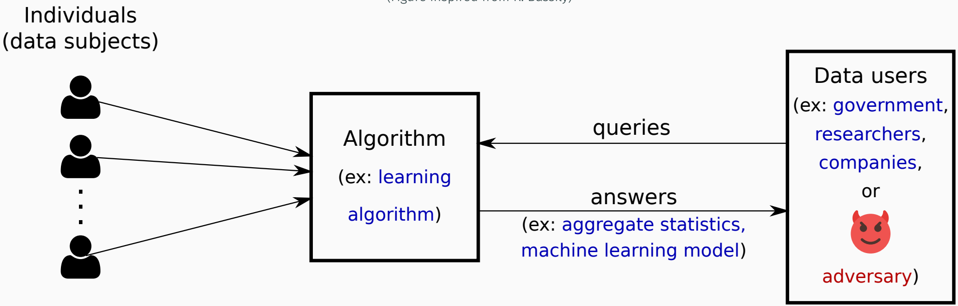
- Massive collection of personal data by companies and public organizations, driven by the progress of data science and AI

- Data is increasingly sensitive and detailed: browsing history, purchase history, social network posts, speech, geolocation, health...

- It is sometimes shared unknowingly and without a clear understanding of the risks
  - Risks include discrimination, blackmailing, unsolicited micro-targeting, public shaming...

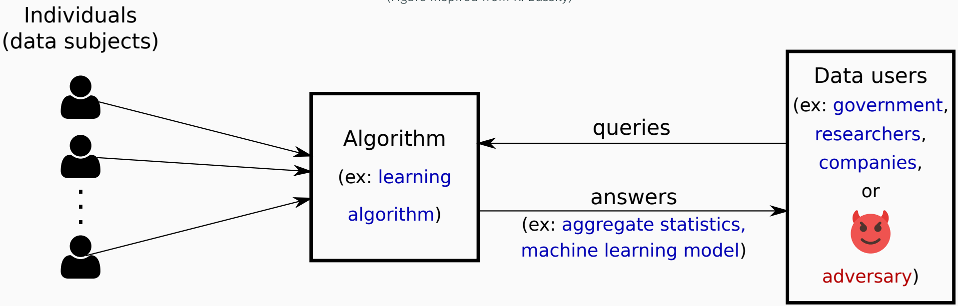
- There is increasing regulation to address privacy-related harms for the collection, use and release of personal data
    - General regulations (e.g., adoption of GDPR by the EU in 2018)
    - Sector- and context-specific regulations, e.g. in health, education, research, finance...
- Privacy has a cost on the utility of the analysis, but ideally it should not destroy it
- One of the main goals of privacy research is to find good trade-offs between utility and privacy so we can better protect individuals and also unlock new applications

(Figure inspired from R. Bassily)



- **Goal:** achieve utility while preserving privacy (conflicting objectives!)
- This is separate from security concerns (e.g., unauthorized access to the system)

(Figure inspired from R. Bassily)



- **Goal:** achieve utility while preserving privacy (conflicting objectives!)
- This is separate from security concerns (e.g., unauthorized access to the system)
- Any ideas on how to do this?

DATA "ANONYMIZATION" IS NOT SAFE

Name	Birth date	Zip code	Gender	Diagnosis	...
Ewen Jordan	1993-09-15	13741	M	Asthma	...
Lea Yang	1999-11-07	13440	F	Type-1 diabetes	...
William Weld	1945-07-31	02110	M	Cancer	...
Clarice Mueller	1950-03-13	02061	F	Cancer	...

- **Anonymization**: removing personally identifiable information before publishing data
- First solution: strip attributes that uniquely identify an individual (e.g., name, social security number...)

DATA "ANONYMIZATION" IS NOT SAFE

Name	Birth date	Zip code	Gender	Diagnosis	...
	1993-09-15	13741	M	Asthma	...
	1999-11-07	13440	F	Type-1 diabetes	...
	1945-07-31	02110	M	Cancer	...
	1950-03-13	02061	F	Cancer	...

- **Anonymization**: removing personally identifiable information before publishing data
- First solution: strip attributes that uniquely identify an individual (e.g., name, social security number...)
- Now we cannot know that William Weld has cancer!

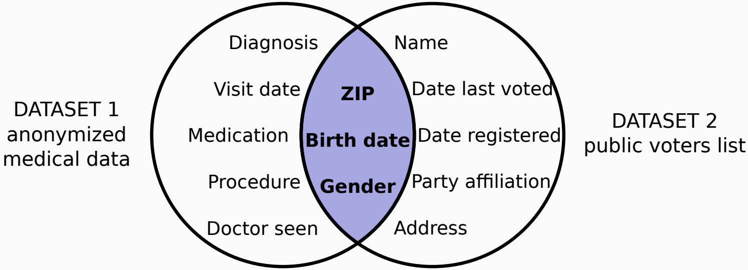
DATA "ANONYMIZATION" IS NOT SAFE

Name	Birth date	Zip code	Gender	Diagnosis	...
	1993-09-15	13741	M	Asthma	...
	1999-11-07	13440	F	Type-1 diabetes	...
	1945-07-31	02110	M	Cancer	...
	1950-03-13	02061	F	Cancer	...

- **Anonymization**: removing personally identifiable information before publishing data
- First solution: strip attributes that uniquely identify an individual (e.g., name, social security number...)
- Now we cannot know that William Weld has cancer!
- Or can we? 😈

DATA "ANONYMIZATION" IS NOT SAFE

(Figure inspired from C. Palamidessi)



- **Problem**: susceptible to linkage attacks, i.e. uniquely linking a record in the anonymized dataset to an identified record in a public dataset
- For instance, an estimated 87% of the US population is uniquely identified by the combination of their gender, birthdate and zip code
- In the late 90s, L. Sweeney managed to re-identify the medical record of the governor of Massachusetts using a public voters list

DATA "ANONYMIZATION" IS NOT SAFE

Name	Birth date	Zip code	Gender	Diagnosis	...
	1993-09-15	13741	M	Asthma	...
	1999-11-07	13440	F	Type-1 diabetes	...
	1945-07-31	02110	M	Cancer	...
	1950-03-13	02061	F	Cancer	...

- Second solution: k-anonymity [Sweeney, 2002]
  1. Define a set of attributes as quasi-identifiers (QIs)
  2. Suppress/generalize attributes and/or add dummy records to make every record in the dataset indistinguishable from at least $k-1$ other records with respect to QIs

DATA "ANONYMIZATION" IS NOT SAFE

	Quasi identifiers			Sensitive attribute	
Name	**Age**	**Zip code**	**Gender**	**Diagnosis**	**...**
	20-30	13***		Asthma	...
	20-30	13***		Type-1 diabetes	...
	70-80	02***		Cancer	...
	70-80	02***		Cancer	...

- Second solution: k-anonymity [Sweeney, 2002]
    1. Define a set of attributes as quasi-identifiers (QIs)
    2. Suppress/generalize attributes and/or add dummy records to make every record in the dataset indistinguishable from at least $k-1$ other records with respect to QIs
- Better now?

DATA “ANONYMIZATION” IS NOT SAFE

	Quasi identifiers			Sensitive attribute	
Name	**Age**	**Zip code**	**Gender**	**Diagnosis**	**...**
	20-30	13***		Asthma	...
	20-30	13***		Type-1 diabetes	...
	70-80	02***		Cancer	...
	70-80	02***		Cancer	...

- Second solution: k-anonymity [Sweeney, 2002]
    1. Define a set of attributes as quasi-identifiers (QIs)
    2. Suppress/generalize attributes and/or add dummy records to make every record in the dataset indistinguishable from at least $k-1$ other records with respect to QIs
- Better now?
- No! Can still infer that W. Weld has cancer (everyone in the group has cancer)

DATA "ANONYMIZATION" IS NOT SAFE

- Variants of k-anonymity (t-closeness, ℓ-diversity) try to address the previous issue but require to modify the original data even more, which often destroys utility
- In high-dimensional and sparse datasets, any combination of attributes is a potential PII that can be exploited using appropriate auxiliary knowledge
  - De-anonymization of Netflix dataset protected with k-anonymity using a few public ratings from IMDB [Narayanan and Shmatikov, 2008]
  - De-anonymization of Twitter graph using Flickr [Narayanan and Shmatikov, 2009]
  - 4 spatio-temporal points uniquely identify most people [de Montjoye et al., 2013]
- **Conclusion**: data cannot be fully anonymized AND remain useful

AGGREGATE STATISTICS ARE NOT SAFE

- How about releasing aggregate statistics about many individuals?

- How about releasing aggregate statistics about many individuals?
- **Problem 1**: differencing attacks, i.e. combining aggregate queries to obtain precise information about specific individuals (note: this can be hard to detect)
  - Average salary in a company before and after an employee joins

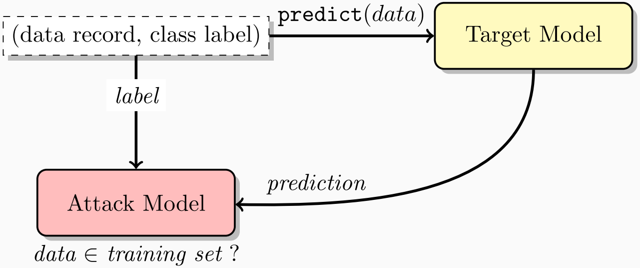
AGGREGATE STATISTICS ARE NOT SAFE

- How about releasing aggregate statistics about many individuals?
- **Problem 1**: differencing attacks, i.e. combining aggregate queries to obtain precise information about specific individuals (note: this can be hard to detect)
  - Average salary in a company before and after an employee joins
- **Problem 2**: membership inference attacks, i.e. inferring presence of known individual in a dataset from (high-dimensional) aggregate statistics
  - Statistics about genomic variants [Homer et al., 2008]

AGGREGATE STATISTICS ARE NOT SAFE

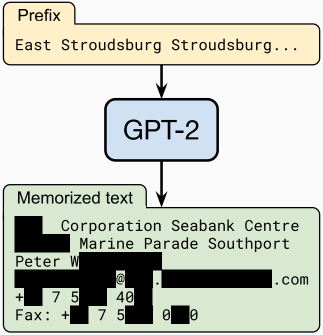
- How about releasing aggregate statistics about many individuals?
- **Problem 1**: differencing attacks, i.e. combining aggregate queries to obtain precise information about specific individuals (note: this can be hard to detect)
  - Average salary in a company before and after an employee joins
- **Problem 2**: membership inference attacks, i.e. inferring presence of known individual in a dataset from (high-dimensional) aggregate statistics
  - Statistics about genomic variants [Homer et al., 2008]
- **Problem 3**: reconstruction attacks, i.e. inferring (part of) the dataset from the output of many aggregate queries
  - See this short video for an overview of the classic attack of [Dinur and Nissim, 2003]

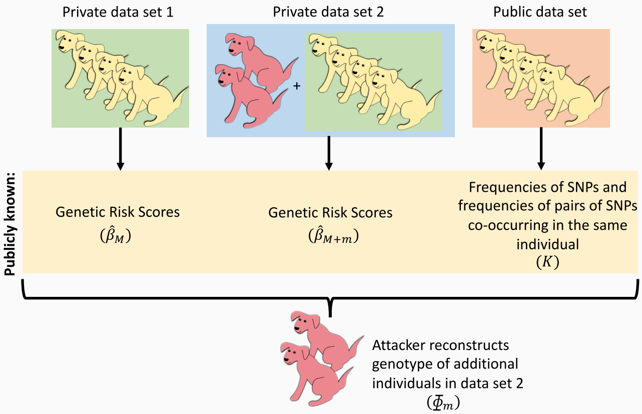
- ML models are elaborate kinds of aggregate statistics!
- As such, they are susceptible to membership inference attacks, i.e. inferring the presence of a known individual in the training set
- For instance, one can exploit the confidence in model predictions [Shokri et al., 2017] [Carlini et al., 2022]



ML MODELS ARE NOT SAFE

- ML models are also susceptible to reconstruction attacks
- For instance, one can extract sensitive text from large language models [Carlini et al., 2021] or run differencing attacks on ML models [Paige et al., 2020]





ORDINARY FACTS ARE NOT ALWAYS SAFE

- Revealing ordinary facts to inappropriate parties may also be problematic, especially if an individual is followed over time
- Example: Alice buys bread every day for 20 years and then stops

ORDINARY FACTS ARE NOT ALWAYS SAFE

- Revealing ordinary facts to inappropriate parties may also be problematic, especially if an individual is followed over time
- Example: Alice buys bread every day for 20 years and then stops
    - An insurance analyst might conclude that Alice has been diagnosed with type 2 diabetes
    - This may be wrong, but in any case Alice could be harmed (e.g., charged with higher insurance premiums)

SUMMARY OF THE KEY ISSUES / REQUIREMENTS

1. Auxiliary knowledge (also called background knowledge or side information): we need to be robust to whatever knowledge the adversary may have, since we cannot predict what an adversary knows or might know in the future

1. Auxiliary knowledge (also called background knowledge or side information): we need to be robust to whatever knowledge the adversary may have, since we cannot predict what an adversary knows or might know in the future
2. Multiple analyses: we need to be able to track how much information is leaked when asking several questions about the same data, and avoid catastrophic leaks

Differential Privacy

First attempt at privacy definition

"An analysis of a dataset is private if the result reveals no more about an individual than what was already known about him/her before the analysis."

- Bayesian version: posterior belief same as prior belief

First attempt at privacy definition

"An analysis of a dataset is private if the result reveals no more about an individual than what was already known about him/her before the analysis."

- Bayesian version: posterior belief same as prior belief
- **Problem 1**: Impossible to reveal exactly nothing if the result is to depend at all on the data (otherwise we get zero utility)

First attempt at privacy definition

"An analysis of a dataset is private if the result reveals no more about an individual than what was already known about him/her before the analysis."

- **Problem 2**: "Before/after" requirement unachievable under auxiliary knowledge

First attempt at privacy definition

"An analysis of a dataset is private if the result reveals no more about an individual than what was already known about him/her before the analysis."

- **Problem 2**: "Before/after" requirement unachievable under auxiliary knowledge
- Think of "stupid priors" (e.g., a person's height is between 10 and 20 meters)

First attempt at privacy definition

"An analysis of a dataset is private if the result reveals no more about an individual than what was already known about him/her before the analysis."

- **Problem 2**: "Before/after" requirement unachievable under auxiliary knowledge
- Think of "stupid priors" (e.g., a person's height is between 10 and 20 meters)
- Think about whether Bob's privacy was violated in the following example:
    - Suppose an insurance company knows that Bob is a smoker
    - A medical data analysis reveals that smoking and cancer are correlated
    - The insurance company decides to raise Bob's rates

First attempt at privacy definition

"An analysis of a dataset is private if the result reveals no more about an individual than what was already known about him/her before the analysis."

- **Problem 2**: "Before/after" requirement unachievable under auxiliary knowledge
- Think of "stupid priors" (e.g., a person's height is between 10 and 20 meters)
- Think about whether Bob's privacy was violated in the following example:
    - Suppose an insurance company knows that Bob is a smoker
    - A medical data analysis reveals that smoking and cancer are correlated
    - The insurance company decides to raise Bob's rates
- This happens even if Bob's data wasn't included in the analysis!

First attempt at privacy definition

"An analysis of a dataset is private if the result reveals no more about an individual than what was already known about him/her before the analysis."

- **Problem 2**: "Before/after" requirement unachievable under auxiliary knowledge
- Think of "stupid priors" (e.g., a person's height is between 10 and 20 meters)
- Think about whether Bob's privacy was violated in the following example:
  - Suppose an insurance company knows that Bob is a smoker
  - A medical data analysis reveals that smoking and cancer are correlated
  - The insurance company decides to raise Bob's rates
- This happens even if Bob's data wasn't included in the analysis!
- Such correlations are precisely the kind of things we want to be able to learn

Second attempt at privacy definition

"An analysis of a dataset is private if what can be learned about an individual in the dataset is not much more than what would be learned if the same analysis was conducted without him/her in the dataset."

Second attempt at privacy definition

"An analysis of a dataset is private if what can be learned about an individual in the dataset is not much more than what would be learned if the same analysis was conducted without him/her in the dataset."

- Intuition: cannot infer the presence/absence of an individual in the dataset, or anything "specific" about an individual (here, 'specific" refers to information that cannot be inferred unless the individual's data is used in the analysis)

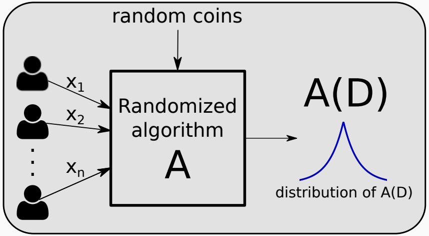
RANDOMIZED ALGORITHM

- Let $\mathcal{X}$ denote an abstract data domain
- A dataset $\mathcal{D} \in \mathcal{X}^n$ is a multiset of n elements (records, or rows) from $\mathcal{X}$

- Let $\mathcal{X}$ denote an abstract data domain
- A dataset $\mathcal{D} \in \mathcal{X}^n$ is a multiset of n elements (records, or rows) from $\mathcal{X}$

Definition (Randomized algorithm)

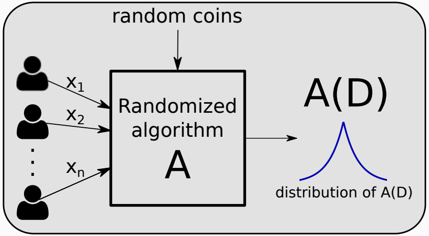
A randomized algorithm $\mathcal{A}$ is a mapping $\mathcal{A} : \mathcal{X}^n \to \mathcal{O}$ where $\mathcal{O}$ is a probability space. In other words, for any dataset $\mathcal{D} \in \mathcal{X}^n$, $\mathcal{A}(\mathcal{D})$ is a random variable taking values in $\mathcal{O}$.



- Let $\mathcal{X}$ denote an abstract data domain
- A dataset $\mathcal{D} \in \mathcal{X}^n$ is a multiset of n elements (records, or rows) from $\mathcal{X}$

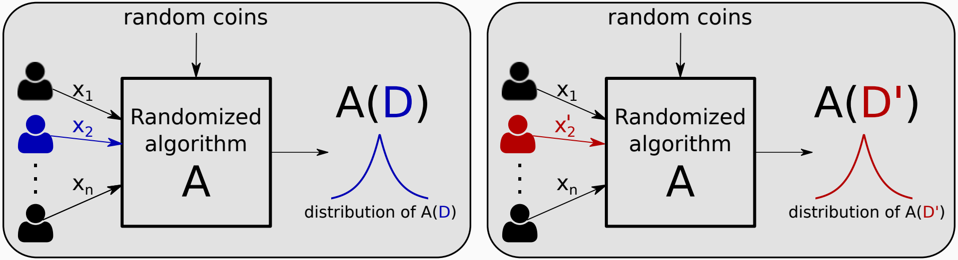
Definition (Randomized algorithm)

A randomized algorithm $\mathcal{A}$ is a mapping $\mathcal{A} : \mathcal{X}^n \rightarrow \mathcal{O}$ where $\mathcal{O}$ is a probability space. In other words, for any dataset $\mathcal{D} \in \mathcal{X}^n$, $\mathcal{A}(\mathcal{D})$ is a random variable taking values in $\mathcal{O}$.



- Example: for a counting algorithm returning (an estimate of) the number of records in $\mathcal{D}$ matching some condition, we have $\mathcal{O} = \mathbb{N}$
- The output space $\mathcal{O}$ may be the same as the input space $\mathcal{X}^n$

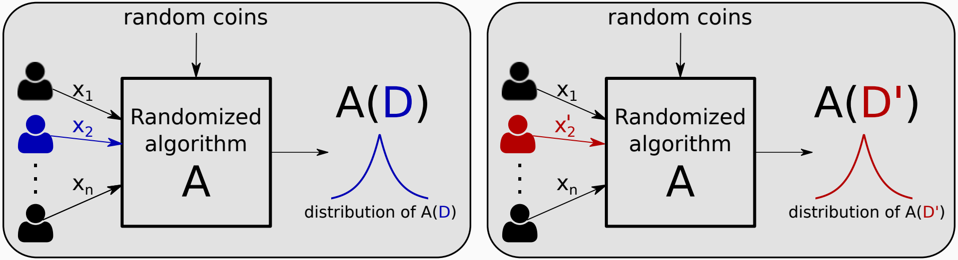
(Figure inspired from R. Bassily)



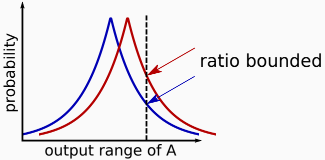
- Datasets $\mathcal{D}, \mathcal{D}' \in \mathcal{X}^n$ are neighboring ($\mathcal{D} \sim \mathcal{D}'$) if they differ on at most one record

DIFFERENTIAL PRIVACY

(Figure inspired from R. Bassily)



- Datasets $\mathcal{D}, \mathcal{D}' \in \mathcal{X}^n$ are neighboring ($\mathcal{D} \sim \mathcal{D}'$) if they differ on at most one record
- DP requires that $\mathcal{A}(\mathcal{D})$ and $\mathcal{A}(\mathcal{D}')$ have "close" distribution



Definition (Differential privacy [Dwork et al., 2006])

Let $\varepsilon > 0$ and $\delta \in [0, 1)$. A randomized algorithm $\mathcal{A} : \mathcal{X}^n \to \mathcal{O}$ is (ε, δ)-differentially private (DP) if for all pairs of neighboring datasets $\mathcal{D} \sim \mathcal{D}'$ and for all $\mathcal{S} \subseteq \mathcal{O}$:

$$\Pr[\mathcal{A}(\mathcal{D}) \in \mathcal{S}] \leq e^{\varepsilon} \Pr[\mathcal{A}(\mathcal{D}') \in \mathcal{S}] + \delta, \tag{1}$$

where the probability space is over the coin flips of $\mathcal{A}$.

- **Key principle**: privacy is a property of the analysis, not of a particular output (in contrast to e.g., k-anonymization)
- Eq. (1) must hold for *all* pairs of neighboring datasets and *all* possible outputs of $\mathcal{A}$
- A non-trivial differentially private algorithm *must* be randomized
- In 2017, Dwork, McSherry, Nissim & Smith won the Gödel prize for introducing DP

INTERPRETING DP: THE PRIVACY LOSS

- $(\varepsilon, 0)$-DP ensures that, for *every* run of the algorithm $\mathcal{A}(\mathcal{D})$, the output is almost equally likely to be observed on every neighboring dataset *simultaneously*
- $(\varepsilon, 0)$-DP is called pure ε-DP. How can we interpret approximate (ε, δ)-DP?

- $(\varepsilon, 0)$-DP ensures that, for *every* run of the algorithm $\mathcal{A}(\mathcal{D})$, the output is almost equally likely to be observed on every neighboring dataset *simultaneously*
- $(\varepsilon, 0)$-DP is called pure ε-DP. How can we interpret approximate (ε, δ)-DP?
- Consider the following quantity, which is often referred to as the privacy loss incurred by observing an output $o \in \mathcal{O}$:

$$L^{o}_{\mathcal{A}(\mathcal{D}),\mathcal{A}(\mathcal{D}')} = \ln\left(\frac{\Pr[\mathcal{A}(\mathcal{D}) = o]}{\Pr[\mathcal{A}(\mathcal{D}') = o]}\right)$$

- $(\varepsilon, 0)$-DP ensures that, for *every* run of the algorithm $\mathcal{A}(\mathcal{D})$, the output is almost equally likely to be observed on every neighboring dataset *simultaneously*

- $(\varepsilon, 0)$-DP is called pure ε-DP. How can we interpret approximate (ε, δ)-DP?

- Consider the following quantity, which is often referred to as the privacy loss incurred by observing an output $o \in \mathcal{O}$:

$$L^{o}_{\mathcal{A}(\mathcal{D}),\mathcal{A}(\mathcal{D}')} = \ln\left(\frac{\Pr[\mathcal{A}(\mathcal{D}) = o]}{\Pr[\mathcal{A}(\mathcal{D}') = o]}\right)$$

- (ε, δ)-DP ensures that the absolute value of the privacy loss will be bounded by ε with probability at least $1 - \delta$ over $o \sim \mathcal{A}(\mathcal{D})$

- Note: ϵ can be seen as a function of δ

- For meaningful privacy guarantees, δ should be $o(1/n)$

- For meaningful privacy guarantees, δ should be $o(1/n)$
- Indeed, setting δ of order $1/n$ allows to release the records of a small number of individuals in the dataset preserves privacy ("just a few" principle)

INTERPRETING DP: VALUES OF ε AND δ

- For meaningful privacy guarantees, δ should be $o(1/n)$
- Indeed, setting δ of order $1/n$ allows to release the records of a small number of individuals in the dataset preserves privacy (“just a few” principle)
- For ε, there are some rules of thumb:
  - $\varepsilon = 1$ (i.e., $e^\varepsilon \approx 2.7$) is considered to be a good guarantee
  - $\varepsilon = 0.1$ (i.e., $e^\varepsilon \approx 1.1$) is considered to be a very strong guarantee

- For meaningful privacy guarantees, δ should be $o(1/n)$
- Indeed, setting δ of order $1/n$ allows to release the records of a small number of individuals in the dataset preserves privacy ("just a few" principle)
- For ε, there are some rules of thumb:
    - $\varepsilon = 1$ (i.e., $e^\varepsilon \approx 2.7$) is considered to be a good guarantee
    - $\varepsilon = 0.1$ (i.e., $e^\varepsilon \approx 1.1$) is considered to be a very strong guarantee
- Guarantees against concrete attacks depend on the use-case and attack scenario, see [Abowd, 2018] [Jayaraman and Evans, 2019] [Nasr et al., 2021] for empirical studies

- DP guarantees are intrinsically robust to arbitrary auxiliary knowledge: it bounds the relative advantage that an adversary gets from observing the output of an algorithm
  - Adversary may know all the dataset except one record
  - Adversary may know all external sources of knowledge, present and future
- The algorithm $\mathcal{A}$ can be public: only the randomness needs to remain hidden
  - A key requirement of modern security ("security by obscurity" has long been rejected)
  - Allows to openly discuss the algorithms and their guarantees

Theorem (Postprocessing)

Let $\mathcal{A} : \mathcal{X}^n \to \mathcal{O}$ be (ε, δ)-DP and let $f : \mathcal{O} \to \mathcal{O}'$ be an arbitrary (randomized) function independent of $\mathcal{A}$. Then

$$f \circ \mathcal{A} : \mathcal{X}^n \to \mathcal{O}'$$

is (ε, δ)-DP.

- "Thinking about" the output of a differentially private algorithm cannot make it less differentially private $\to$ can let data users do whatever they want with it
- This holds regardless of attacker strategy and computational power

- Composition allows to control the *worst-case* cumulative privacy loss over multiple analyses run on the same dataset, including complex multi-step algorithms

Theorem (Simple composition)

Let $\mathcal{A}_1, \dots, \mathcal{A}_K$ be such that $\mathcal{A}_k$ satisfies $(\varepsilon_k, \delta_k)$-DP. For any dataset $\mathcal{D}$, let $\mathcal{A}$ be such that $\mathcal{A}(\mathcal{D}) = (\mathcal{A}_1(\mathcal{D}), \dots, \mathcal{A}_k(\mathcal{D}))$. Then $\mathcal{A}$ is (ε, δ)-DP with $\varepsilon = \sum_{k=1}^K \varepsilon_k$ and $\delta = \sum_{k=1}^K \delta_k$.

- Composition allows to control the *worst-case* cumulative privacy loss over multiple analyses run on the same dataset, including complex multi-step algorithms

Theorem (Simple composition)

Let $\mathcal{A}_1, \dots, \mathcal{A}_K$ be such that $\mathcal{A}_k$ satisfies $(\varepsilon_k, \delta_k)$-DP. For any dataset $\mathcal{D}$, let $\mathcal{A}$ be such that $\mathcal{A}(\mathcal{D}) = (\mathcal{A}_1(\mathcal{D}), \dots, \mathcal{A}_k(\mathcal{D}))$. Then $\mathcal{A}$ is (ε, δ)-DP with $\varepsilon = \sum_{k=1}^K \varepsilon_k$ and $\delta = \sum_{k=1}^K \delta_k$.

Theorem (Advanced composition)

Let $\epsilon, \delta, \delta' > 0$. If $\mathcal{A}_k$ satisfies (ε, δ)-DP, then $\mathcal{A}$ is $(\varepsilon', K\delta + \delta')$-DP with

$$\varepsilon' = \sqrt{2K \ln(1/\delta')}\varepsilon + K\varepsilon(e^{\varepsilon} - 1)$$

- Composition allows to control the *worst-case* cumulative privacy loss over multiple analyses run on the same dataset, including complex multi-step algorithms

Theorem (Simple composition)

Let $\mathcal{A}_1, \dots, \mathcal{A}_K$ be such that $\mathcal{A}_k$ satisfies $(\varepsilon_k, \delta_k)$-DP. For any dataset $\mathcal{D}$, let $\mathcal{A}$ be such that $\mathcal{A}(\mathcal{D}) = (\mathcal{A}_1(\mathcal{D}), \dots, \mathcal{A}_k(\mathcal{D}))$. Then $\mathcal{A}$ is (ε, δ)-DP with $\varepsilon = \sum_{k=1}^K \varepsilon_k$ and $\delta = \sum_{k=1}^K \delta_k$.

Theorem (Advanced composition)

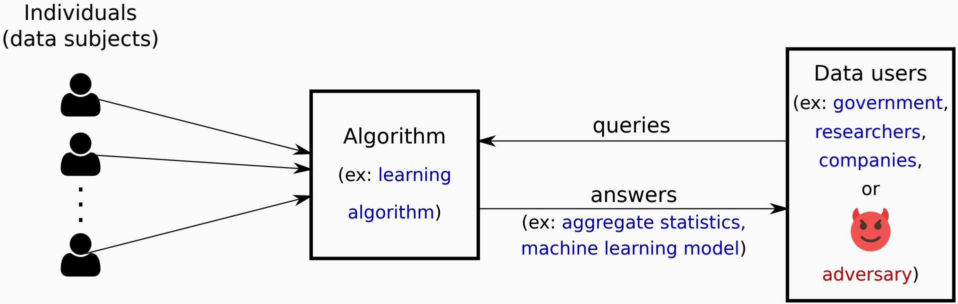
Let $\epsilon, \delta, \delta' > 0$. If $\mathcal{A}_k$ satisfies (ε, δ)-DP, then $\mathcal{A}$ is $(\varepsilon', K\delta + \delta')$-DP with

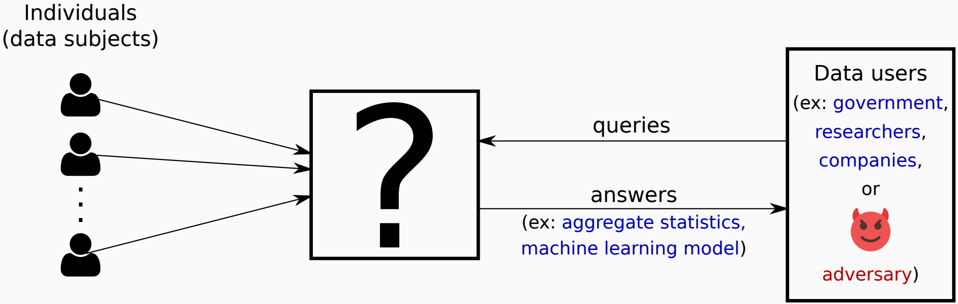
$$\varepsilon' = \sqrt{2K\ln(1/\delta')}\varepsilon + K\varepsilon(e^\varepsilon - 1)$$

- The sequence of algorithms can be chosen adaptively
- Numerically tighter composition can be obtained with through a variant of DP based on the Rényi divergence [Mironov, 2017]

- DP has become a gold standard metric of privacy in fundamental science but is also being increasingly used in real-world deployments
- Thousands of scientific papers in the fields of privacy, security, databases, data mining, machine learning...
- DP is deployed for computing/releasing statistics (including by tech giants...):
  - Adoption by the US Census Bureau starting in 2020 [Abowd, 2018]
  - Telemetry in Google Chrome [Erlingsson et al., 2014]
  - Keyboard statistics in iOS and macOS [Differential Privacy Team, Apple, 2017]
  - Application usage statistics by Microsoft [Ding et al., 2017]
- Open source software for DP in ML: TensorFlow Privacy, Opacus, PySyft...

HOW TO DESIGN DP ALGORITHMS?




Individuals
(data subjects)
?
queries
answers
(ex: aggregate statistics,
machine learning model)
Data users
(ex: government,
researchers,
companies,
or
adversary)

The Gaussian Mechanism

- Suppose we want to compute a numeric function $f : \mathcal{X}^n \to \mathbb{R}^K$ of a private dataset $\mathcal{D}$
- How to construct a DP algorithm (or mechanism) for computing $f(\mathcal{D})$?
  - How much randomness (error) do we add?
  - How to introduce this randomness in the output?

Definition (Global ℓ_2 sensitivity)

The global ℓ_2 sensitivity of a query (function) $f : \mathcal{X}^n \to \mathbb{R}^K$ is

$$\Delta_2(f) = \max_{\mathcal{D} \sim \mathcal{D}'} \|f(\mathcal{D}) - f(\mathcal{D}')\|_2$$

- How much one record can affect the value of the function
- Intuitively, it gives the amount of uncertainty needed to hide any single contribution
- Think about the sensitivity of the following queries:

Definition (Global ℓ_2 sensitivity)

The global ℓ_2 sensitivity of a query (function) $f : \mathcal{X}^n \to \mathbb{R}^K$ is

$$\Delta_2(f) = \max_{\mathcal{D} \sim \mathcal{D}'} \|f(\mathcal{D}) - f(\mathcal{D}')\|_2$$

- How much one record can affect the value of the function
- Intuitively, it gives the amount of uncertainty needed to hide any single contribution
- Think about the sensitivity of the following queries:
  - How many people have blond hair?

Definition (Global ℓ_2 sensitivity)

The global ℓ_2 sensitivity of a query (function) $f : \mathcal{X}^n \to \mathbb{R}^K$ is

$$\Delta_2(f) = \max_{\mathcal{D} \sim \mathcal{D}'} \|f(\mathcal{D}) - f(\mathcal{D}')\|_2$$

- How much one record can affect the value of the function
- Intuitively, it gives the amount of uncertainty needed to hide any single contribution
- Think about the sensitivity of the following queries:
  - How many people have blond hair?
  - How many males, how many people with blond hair?

Definition (Global ℓ_2 sensitivity)

The global ℓ_2 sensitivity of a query (function) $f : \mathcal{X}^n \to \mathbb{R}^K$ is

$$\Delta_2(f) = \max_{\mathcal{D} \sim \mathcal{D}'} \|f(\mathcal{D}) - f(\mathcal{D}')\|_2$$

- How much one record can affect the value of the function
- Intuitively, it gives the amount of uncertainty needed to hide any single contribution
- Think about the sensitivity of the following queries:
    - How many people have blond hair?
    - How many males, how many people with blond hair?
    - How many people have blond hair, how many people have dark hair, how many people have brown hair, how many people have red hair?

Definition (Global ℓ_2 sensitivity)

The global ℓ_2 sensitivity of a query (function) $f : \mathcal{X}^n \to \mathbb{R}^K$ is

$$\Delta_2(f) = \max_{\mathcal{D} \sim \mathcal{D}'} \|f(\mathcal{D}) - f(\mathcal{D}')\|_2$$

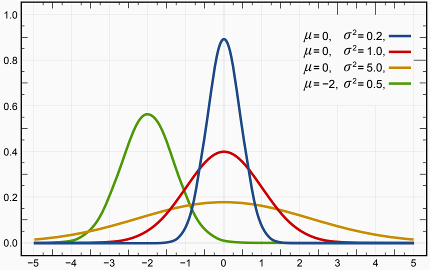
- How much one record can affect the value of the function
- Intuitively, it gives the amount of uncertainty needed to hide any single contribution
- Think about the sensitivity of the following queries:
  - How many people have blond hair?
  - How many males, how many people with blond hair?
  - How many people have blond hair, how many people have dark hair, how many people have brown hair, how many people have red hair?
  - What is the average salary?

Definition (Gaussian distribution)

For $\mu \in \mathbb{R}$, $\sigma^2 > 0$, The Gaussian distribution $\mathcal{N}(\mu, \sigma^2)$ with mean μ and variance σ^2 is the distribution with probability density function:

$$p(y; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right), \quad y \in \mathbb{R}.$$

- If $Y \sim \mathcal{N}(\mu, \sigma^2)$, then $\mathbb{E}[Y] = \mu$, $Var[Y] = \sigma^2$
- Tail bound: $\Pr[|Y - \mu| > t\sigma] \leq 2e^{-\frac{t^2}{2}}$



Algorithm: Gaussian mechanism $\mathcal{A}_{\text{Gauss}}(\mathcal{D}, f : \mathcal{X}^n \to \mathbb{R}^K, \varepsilon, \delta)$

1. Compute $\Delta = \Delta_2(f)$
2. For $k = 1, \ldots, K$: draw $Y_k \sim \mathcal{N}(0, \sigma^2)$ independently for each k, where $\sigma = \frac{\sqrt{2 \ln(1.25/\delta)}\Delta}{\varepsilon}$
3. Output $f(\mathcal{D}) + Y$, where $Y = (Y_1, \ldots, Y_K) \in \mathbb{R}^K$

- This is output perturbation: perturb each entry of $f(\mathcal{D})$ with independent Gaussian noise calibrated to the sensitivity Δ of f and the privacy parameters (ε, δ)

Algorithm: Gaussian mechanism $\mathcal{A}_{\text{Gauss}}(\mathcal{D}, f : \mathcal{X}^n \to \mathbb{R}^K, \varepsilon, \delta)$

1. Compute $\Delta = \Delta_2(f)$
2. For $k = 1, \ldots, K$: draw $Y_k \sim \mathcal{N}(0, \sigma^2)$ independently for each k, where $\sigma = \frac{\sqrt{2\ln(1.25/\delta)}\Delta}{\varepsilon}$
3. Output $f(\mathcal{D}) + Y$, where $Y = (Y_1, \ldots, Y_K) \in \mathbb{R}^K$

- This is output perturbation: perturb each entry of $f(\mathcal{D})$ with independent Gaussian noise calibrated to the sensitivity Δ of f and the privacy parameters (ε, δ)
- The dependence of σ^2 on $1/\delta$ is logarithmic, which is good since we want δ very small!

Algorithm: Gaussian mechanism $\mathcal{A}_{\text{Gauss}}(\mathcal{D}, f : \mathcal{X}^n \to \mathbb{R}^K, \varepsilon, \delta)$

1. Compute $\Delta = \Delta_2(f)$
2. For $k = 1, \dots, K$: draw $Y_k \sim \mathcal{N}(0, \sigma^2)$ independently for each k, where $\sigma = \frac{\sqrt{2\ln(1.25/\delta)}\Delta}{\varepsilon}$
3. Output $f(\mathcal{D}) + Y$, where $Y = (Y_1, \dots, Y_K) \in \mathbb{R}^K$

- This is output perturbation: perturb each entry of $f(\mathcal{D})$ with independent Gaussian noise calibrated to the sensitivity Δ of f and the privacy parameters (ε, δ)
- The dependence of σ^2 on $1/\delta$ is logarithmic, which is good since we want δ very small!
- It is not possible to achieve $\delta = 0$

Algorithm: Gaussian mechanism $\mathcal{A}_{\text{Gauss}}(\mathcal{D}, f : \mathcal{X}^n \to \mathbb{R}^K, \varepsilon, \delta)$

1. Compute $\Delta = \Delta_2(f)$
2. For $k = 1, \dots, K$: draw $Y_k \sim \mathcal{N}(0, \sigma^2)$ independently for each k, where $\sigma = \frac{\sqrt{2\ln(1.25/\delta)}\Delta}{\varepsilon}$
3. Output $f(\mathcal{D}) + Y$, where $Y = (Y_1, \dots, Y_K) \in \mathbb{R}^K$

- This is output perturbation: perturb each entry of $f(\mathcal{D})$ with independent Gaussian noise calibrated to the sensitivity Δ of f and the privacy parameters (ε, δ)
- The dependence of σ^2 on $1/\delta$ is logarithmic, which is good since we want δ very small!
- It is not possible to achieve $\delta = 0$

Theorem (DP guarantees for Gaussian mechanism)

Let $\varepsilon, \delta > 0$ and $f : \mathcal{X}^n \to \mathbb{R}^K$. The Gaussian mechanism $\mathcal{A}_{Gauss}(\cdot, f, \varepsilon, \delta)$ is (ε, δ)-DP.

Proof sketch (see [Dwork and Roth, 2014], Appendix A for details).

- Consider any pair of datasets $\mathcal{D}, \mathcal{D}'$ such that $\mathcal{D} \sim \mathcal{D}'$, and let $K = 1$ for simplicity

$\square$

Proof sketch (see [Dwork and Roth, 2014], Appendix A for details).

- Consider any pair of datasets $\mathcal{D}, \mathcal{D}'$ such that $\mathcal{D} \sim \mathcal{D}'$, and let $K = 1$ for simplicity
- We can bound the absolute privacy loss of observing output $f(\mathcal{D}) + y$ by:

$$\left| \ln \frac{\Pr[\mathcal{A}(\mathcal{D}) = f(\mathcal{D}) + y]}{\Pr[\mathcal{A}(\mathcal{D}') = f(\mathcal{D}) + y]} \right| \leq \left| \ln \frac{e^{-(1/2\sigma^2)y^2}}{e^{-(1/2\sigma^2)(y+\Delta_2(f))^2}} \right| = \left| \frac{1}{2\sigma^2}(2y\Delta_2(f) + \Delta_2(f)^2) \right|$$

$\square$

Proof sketch (see [Dwork and Roth, 2014], Appendix A for details).

- Consider any pair of datasets $\mathcal{D}, \mathcal{D}'$ such that $\mathcal{D} \sim \mathcal{D}'$, and let $K = 1$ for simplicity
- We can bound the absolute privacy loss of observing output $f(\mathcal{D}) + y$ by:

$$\left| \ln \frac{\Pr[\mathcal{A}(\mathcal{D}) = f(\mathcal{D}) + y]}{\Pr[\mathcal{A}(\mathcal{D}') = f(\mathcal{D}) + y]} \right| \leq \left| \ln \frac{e^{-(1/2\sigma^2)y^2}}{e^{-(1/2\sigma^2)(y+\Delta_2(f))^2}} \right| = \left| \frac{1}{2\sigma^2}(2y\Delta_2(f) + \Delta_2(f)^2) \right|$$

- This is bounded by ε whenever $y < \sigma^2\varepsilon/\Delta_2(f) - \Delta_2(f)/2$

$\square$

Proof sketch (see [Dwork and Roth, 2014], Appendix A for details).

- Consider any pair of datasets $\mathcal{D}, \mathcal{D}'$ such that $\mathcal{D} \sim \mathcal{D}'$, and let $K = 1$ for simplicity
- We can bound the absolute privacy loss of observing output $f(\mathcal{D}) + y$ by:

$$\left| \ln \frac{\Pr[\mathcal{A}(\mathcal{D}) = f(\mathcal{D}) + y]}{\Pr[\mathcal{A}(\mathcal{D}') = f(\mathcal{D}) + y]} \right| \leq \left| \ln \frac{e^{-(1/2\sigma^2)y^2}}{e^{-(1/2\sigma^2)(y+\Delta_2(f))^2}} \right| = \left| \frac{1}{2\sigma^2}(2y\Delta_2(f) + \Delta_2(f)^2) \right|$$

- This is bounded by ε whenever $y < \sigma^2\varepsilon/\Delta_2(f) - \Delta_2(f)/2$
- To guarantee (ε, δ)-DP, it is sufficient to prove that

$$\Pr[|y| \geq \sigma^2\varepsilon/\Delta_2(f) - \Delta_2(f)/2] \leq \delta$$

$\square$

Proof sketch (see [Dwork and Roth, 2014], Appendix A for details).

- Consider any pair of datasets $\mathcal{D}, \mathcal{D}'$ such that $\mathcal{D} \sim \mathcal{D}'$, and let $K = 1$ for simplicity
- We can bound the absolute privacy loss of observing output $f(\mathcal{D}) + y$ by:

$$\left| \ln \frac{\Pr[\mathcal{A}(\mathcal{D}) = f(\mathcal{D}) + y]}{\Pr[\mathcal{A}(\mathcal{D}') = f(\mathcal{D}) + y]} \right| \leq \left| \ln \frac{e^{-(1/2\sigma^2)y^2}}{e^{-(1/2\sigma^2)(y+\Delta_2(f))^2}} \right| = \left| \frac{1}{2\sigma^2}(2y\Delta_2(f) + \Delta_2(f)^2) \right|$$

- This is bounded by ε whenever $y < \sigma^2\varepsilon/\Delta_2(f) - \Delta_2(f)/2$
- To guarantee (ε, δ)-DP, it is sufficient to prove that

$$\Pr[|y| \geq \sigma^2\varepsilon/\Delta_2(f) - \Delta_2(f)/2] \leq \delta$$

- We bound the left hand side using the Gaussian tail bound and verify that the condition is satisfied for the choice of σ

$\square$

THE GAUSSIAN MECHANISM: UTILITY GUARANTEES

- DP induces a privacy-utility trade-off, here in terms of the variance of the estimate

- DP induces a privacy-utility trade-off, here in terms of the variance of the estimate
- In fact, the MSE achieved by the Gaussian mechanism is worst-case optimal

- DP induces a privacy-utility trade-off, here in terms of the variance of the estimate
- In fact, the MSE achieved by the Gaussian mechanism is worst-case optimal
- We can derive high-probability error bounds

Theorem (High probability bound on ℓ_∞ error of the Gaussian mechanism)

Let $\varepsilon > 0$. For a query $f : \mathcal{X}^n \to \mathbb{R}^K$ and any dataset $D \in \mathcal{X}^n$, the Gaussian mechanism $\mathcal{A}_{Gauss}(\mathcal{D}, f, \varepsilon)$ has the following utility guarantee:

$$\Pr\left[\|\mathcal{A}_{Gauss}(\mathcal{D}, f, \varepsilon) - f(\mathcal{D})\|_\infty < \sqrt{2\ln(1.25/\delta)\ln(K/\beta)}\frac{\Delta_2(f)}{\varepsilon}\right] \geq 1 - \beta.$$

- Proof: use the Gaussian tail bound and a union bound

- Suppose we wish to calculate which first names, from a list of 10,000 potential names, are most common among participants of the 2018 French census

- Suppose we wish to calculate which first names, from a list of 10,000 potential names, are most common among participants of the 2018 French census
- We can think of this as a query $f : \mathcal{X}^n \to \mathbb{R}^{10000}$

- Suppose we wish to calculate which first names, from a list of 10,000 potential names, are most common among participants of the 2018 French census
- We can think of this as a query $f : \mathcal{X}^n \to \mathbb{R}^{10000}$
- This is a histogram query with sensitivity $\Delta_2(f) = \sqrt{2}$

- Suppose we wish to calculate which first names, from a list of 10,000 potential names, are most common among participants of the 2018 French census
- We can think of this as a query $f : \mathcal{X}^n \to \mathbb{R}^{10000}$
- This is a histogram query with sensitivity $\Delta_2(f) = \sqrt{2}$
- We can answer this query with $(1, 10^{-9})$-DP and, using the previous theorem, with probability 0.99 no estimate will be off by more than an additive error of

$$\sqrt{4 \ln(1.25 \cdot 10^9) / \ln(10000/.05)} \approx 34$$

- Suppose we wish to calculate which first names, from a list of 10,000 potential names, are most common among participants of the 2018 French census
- We can think of this as a query $f : \mathcal{X}^n \to \mathbb{R}^{10000}$
- This is a histogram query with sensitivity $\Delta_2(f) = \sqrt{2}$
- We can answer this query with $(1, 10^{-9})$-DP and, using the previous theorem, with probability 0.99 no estimate will be off by more than an additive error of

$$\sqrt{4 \ln(1.25 \cdot 10^9) / \ln(10000/.05)} \approx 34$$

- This is pretty low for a country of more than $66,000,000$ people!

Differentially Private SGD

- A trusted curator wants to privately release a model trained on data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$
- We focus here on approximately solving an Empirical Risk Minimization (ERM) problem under an (ϵ, δ)-DP constraint:

$$\min_{\theta \in \Theta} \left\{ F(\theta; \mathcal{D}) := \frac{1}{n} \sum_{i=1}^{n} L(\theta; x_i, y_i) \right\}, \quad \text{with } L \text{ differentiable in } \theta$$

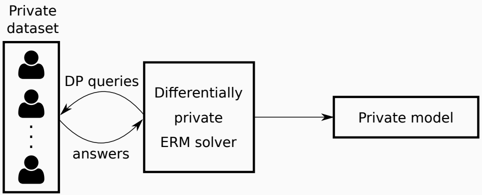
(Note: in some cases, DP can imply generalization [Bassily et al., 2016, Jung et al., 2021])

- A trusted curator wants to privately release a model trained on data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$
- We focus here on approximately solving an Empirical Risk Minimization (ERM) problem under an (ϵ, δ)-DP constraint:

$$\min_{\theta \in \Theta} \left\{ F(\theta; \mathcal{D}) := \frac{1}{n} \sum_{i=1}^{n} L(\theta; x_i, y_i) \right\}, \quad \text{with } L \text{ differentiable in } \theta$$

(Note: in some cases, DP can imply generalization [Bassily et al., 2016, Jung et al., 2021])

- We can achieve this by designing a differentially private ERM solver



- Denote by $\Pi_\Theta(\theta) = \arg\min_{\theta' \in \Theta} \|\theta - \theta'\|_2$ the projection operator onto Θ

- Denote by $\Pi_\Theta(\theta) = \arg\min_{\theta' \in \Theta} \|\theta - \theta'\|_2$ the projection operator onto Θ

Algorithm: Non-private (projected) SGD

- Initialize parameters to $\theta^{(0)} \in \Theta$
- For $t = 0, \dots, T-1$:
  - Pick $i_t \in \{1, \dots, n\}$ uniformly at random
  - $\theta^{(t+1)} \leftarrow \Pi_\Theta\big(\theta^{(t)} - \gamma_t \nabla L(\theta^{(t)}; x_{i_t}, y_{i_t})\big)$
- Return $\theta^{(T)}$

- Denote by $\Pi_\Theta(\theta) = \arg\min_{\theta' \in \Theta} \|\theta - \theta'\|_2$ the projection operator onto Θ

Algorithm: Non-private (projected) SGD

- Initialize parameters to $\theta^{(0)} \in \Theta$
- For $t = 0, \ldots, T-1$:
  - Pick $i_t \in \{1, \ldots, n\}$ uniformly at random
  - $\theta^{(t+1)} \leftarrow \Pi_\Theta\left(\theta^{(t)} - \gamma_t \nabla L(\theta^{(t)}; x_{i_t}, y_{i_t})\right)$
- Return $\theta^{(T)}$

- SGD is a natural candidate solver: simple, flexible, scalable, heavily used in ML

- Denote by $\Pi_\Theta(\theta) = \arg\min_{\theta' \in \Theta} \|\theta - \theta'\|_2$ the projection operator onto Θ

Algorithm: Non-private (projected) SGD

- Initialize parameters to $\theta^{(0)} \in \Theta$
- For $t = 0, \ldots, T-1$:
  - Pick $i_t \in \{1, \ldots, n\}$ uniformly at random
  - $\theta^{(t+1)} \leftarrow \Pi_\Theta\big(\theta^{(t)} - \gamma_t \nabla L(\theta^{(t)}; x_{i_t}, y_{i_t})\big)$
- Return $\theta^{(T)}$

- SGD is a natural candidate solver: simple, flexible, scalable, heavily used in ML
- Any idea on how to design a DP version of SGD?

- Assume that $L(\cdot; x, y)$ is l-Lipschitz with respect to the ℓ_2 norm for any $(x, y) \in \mathcal{X} \times \mathcal{Y}$:

$$|L(\theta; x, y) - L(\theta'; x, y)| \leq l\|\theta - \theta'\|, \quad \text{for all } \theta, \theta' \in \Theta$$

- Assume that $L(\cdot; x, y)$ is l-Lipschitz with respect to the ℓ_2 norm for any $(x, y) \in \mathcal{X} \times \mathcal{Y}$:

$$|L(\theta; x, y) - L(\theta'; x, y)| \leq l\|\theta - \theta'\|, \quad \text{for all } \theta, \theta' \in \Theta$$

- This implies that for all x, y, θ we have $\|\nabla L(\theta; x, y)\| \leq l$

- Assume that $L(\cdot; x, y)$ is l-Lipschitz with respect to the ℓ_2 norm for any $(x, y) \in \mathcal{X} \times \mathcal{Y}$:

$$|L(\theta; x, y) - L(\theta'; x, y)| \leq l\|\theta - \theta'\|, \quad \text{for all } \theta, \theta' \in \Theta$$

- This implies that for all x, y, θ we have $\|\nabla L(\theta; x, y)\| \leq l$
- Therefore, at any step t of SGD, the ℓ_2 sensitivity of $\nabla L(\theta^{(t-1)}; x_{i_t}, y_{i_t})$ is bounded by $2l$ and we can use the Gaussian mechanism

- Assume that $L(\cdot; x, y)$ is l-Lipschitz with respect to the ℓ_2 norm for any $(x, y) \in \mathcal{X} \times \mathcal{Y}$:

$$|L(\theta; x, y) - L(\theta'; x, y)| \leq l\|\theta - \theta'\|, \quad \text{for all } \theta, \theta' \in \Theta$$

- This implies that for all x, y, θ we have $\|\nabla L(\theta; x, y)\| \leq l$
- Therefore, at any step t of SGD, the ℓ_2 sensitivity of $\nabla L(\theta^{(t-1)}; x_{i_t}, y_{i_t})$ is bounded by $2l$ and we can use the Gaussian mechanism
- It feels like we can do better...

Theorem (Amplification by subsampling [Balle et al., 2018])

Let $\mathcal{X}$ be a data domain and $\mathcal{S} : \mathcal{X}^n \to \mathcal{X}^m$ be a procedure such that $\mathcal{S}(D)$ returns a random subset of m records sampled uniformly without replacement from D. Let $\mathcal{A}$ be an (ε, δ)-DP algorithm. Then $\mathcal{A} \circ \mathcal{S}$ satisfies $(\varepsilon', \frac{m}{n}\delta)$-DP with $\varepsilon' = \ln\left(1 + \frac{m}{n}(e^\varepsilon - 1)\right)$.

Theorem (Amplification by subsampling [Balle et al., 2018])

Let $\mathcal{X}$ be a data domain and $\mathcal{S} : \mathcal{X}^n \to \mathcal{X}^m$ be a procedure such that $\mathcal{S}(D)$ returns a random subset of m records sampled uniformly without replacement from D. Let $\mathcal{A}$ be an (ε, δ)-DP algorithm. Then $\mathcal{A} \circ \mathcal{S}$ satisfies $(\varepsilon', \frac{m}{n}\delta)$-DP with $\varepsilon' = \ln\left(1 + \frac{m}{n}(e^\varepsilon - 1)\right)$.

- The amplification effect is due to the secrecy of the samples

Theorem (Amplification by subsampling [Balle et al., 2018])

Let $\mathcal{X}$ be a data domain and $\mathcal{S} : \mathcal{X}^n \rightarrow \mathcal{X}^m$ be a procedure such that $\mathcal{S}(D)$ returns a random subset of m records sampled uniformly without replacement from D. Let $\mathcal{A}$ be an (ε, δ)-DP algorithm. Then $\mathcal{A} \circ \mathcal{S}$ satisfies $(\varepsilon', \frac{m}{n}\delta)$-DP with $\varepsilon' = \ln\left(1 + \frac{m}{n}(e^{\varepsilon} - 1)\right)$.

- The amplification effect is due to the secrecy of the samples
- For simplicity of exposition, we will use the following approximation: when $\varepsilon \leq 1$, $\ln\left(1 + \frac{m}{n}(e^{\varepsilon} - 1)\right) \leq 2\frac{m}{n}\varepsilon$ (but in practice the tight version above should be used!)
- The proof and results with other sampling schemes can be found in [Balle et al., 2018]

Algorithm: Differentially Private SGD $\mathcal{A}_{\text{DP-SGD}}(\mathcal{D}, L, \varepsilon, \delta)$

- Initialize parameters to $\theta^{(0)} \in \Theta$ (must be independent of $\mathcal{D}$)
- For $t = 0, \dots, T-1$:
  - Pick $i_t \in \{1, \dots, n\}$ uniformly at random
  - $\eta^{(t)} \leftarrow (\eta_1^{(t)}, \dots, \eta_p^{(t)}) \in \mathbb{R}^p$ where each $\eta_j^{(t)} \sim \mathcal{N}(0, \sigma^2)$ with $\sigma = \frac{16l\sqrt{T \ln(2/\delta) \ln(2.5T/\delta n)}}{n\varepsilon}$
  - $\theta^{(t+1)} \leftarrow \Pi_\Theta\left(\theta^{(t)} - \gamma_t(\nabla L(\theta^{(t)}; x_{i_t}, y_{i_t}) + \eta^{(t)})\right)$
- Return $\theta^{(T)}$

Algorithm: Differentially Private SGD $\mathcal{A}_{\text{DP-SGD}}(\mathcal{D}, L, \varepsilon, \delta)$

- Initialize parameters to $\theta^{(0)} \in \Theta$ (must be independent of $\mathcal{D}$)
- For $t = 0, \dots, T-1$:
  - Pick $i_t \in \{1, \dots, n\}$ uniformly at random
  - $\eta^{(t)} \leftarrow (\eta_1^{(t)}, \dots, \eta_p^{(t)}) \in \mathbb{R}^p$ where each $\eta_j^{(t)} \sim \mathcal{N}(0, \sigma^2)$ with $\sigma = \frac{16l\sqrt{T\ln(2/\delta)\ln(2.5T/\delta n)}}{n\varepsilon}$
  - $\theta^{(t+1)} \leftarrow \Pi_\Theta\left(\theta^{(t)} - \gamma_t(\nabla L(\theta^{(t)}; x_{i_t}, y_{i_t}) + \eta^{(t)})\right)$
- Return $\theta^{(T)}$

- More data (larger n) $\rightarrow$ less noise added to each gradient

Algorithm: Differentially Private SGD $\mathcal{A}_{\text{DP-SGD}}(\mathcal{D}, L, \varepsilon, \delta)$

- Initialize parameters to $\theta^{(0)} \in \Theta$ (must be independent of $\mathcal{D}$)
- For $t = 0, \ldots, T-1$:
  - Pick $i_t \in \{1, \ldots, n\}$ uniformly at random
  - $\eta^{(t)} \leftarrow (\eta_1^{(t)}, \ldots, \eta_p^{(t)}) \in \mathbb{R}^p$ where each $\eta_j^{(t)} \sim \mathcal{N}(0, \sigma^2)$ with $\sigma = \frac{16l\sqrt{T\ln(2/\delta)\ln(2.5T/\delta n)}}{n\varepsilon}$
  - $\theta^{(t+1)} \leftarrow \Pi_\Theta\left(\theta^{(t)} - \gamma_t\left(\nabla L(\theta^{(t)}; x_{i_t}, y_{i_t}) + \eta^{(t)}\right)\right)$
- Return $\theta^{(T)}$

- More data (larger n) $\rightarrow$ less noise added to each gradient
- More iterations (larger T) $\rightarrow$ more noise added to each gradient

Algorithm: Differentially Private SGD $\mathcal{A}_{\text{DP-SGD}}(\mathcal{D}, L, \varepsilon, \delta)$

- Initialize parameters to $\theta^{(0)} \in \Theta$ (must be independent of $\mathcal{D}$)
- For $t = 0, \ldots, T-1$:
  - Pick $i_t \in \{1, \ldots, n\}$ uniformly at random
  - $\eta^{(t)} \leftarrow (\eta_1^{(t)}, \ldots, \eta_p^{(t)}) \in \mathbb{R}^p$ where each $\eta_j^{(t)} \sim \mathcal{N}(0, \sigma^2)$ with $\sigma = \frac{16l\sqrt{T \ln(2/\delta) \ln(2.5T/\delta n)}}{n\varepsilon}$
  - $\theta^{(t+1)} \leftarrow \Pi_\Theta\left(\theta^{(t)} - \gamma_t(\nabla L(\theta^{(t)}; x_{i_t}, y_{i_t}) + \eta^{(t)})\right)$
- Return $\theta^{(T)}$

- More data (larger n) $\rightarrow$ less noise added to each gradient
- More iterations (larger T) $\rightarrow$ more noise added to each gradient

Theorem (DP guarantees for DP-SGD)

Let $\varepsilon \leq 1, \delta > 0$. Let the loss function $L(\cdot; x, y)$ be l-Lipschitz w.r.t. the ℓ_2 norm for all $x, y \in \mathcal{X} \times \mathcal{Y}$. Then $\mathcal{A}_{\text{DP-SGD}}(\cdot, L, \varepsilon, \delta)$ is (ε, δ)-DP.

Proof.

- Recall that for a query with ℓ_2 sensitivity Δ, achieving (ε', δ') with the Gaussian mechanism requires to add noise with standard deviation $\sigma' = \frac{\sqrt{2\ln(1.25/\delta')}\Delta}{\varepsilon'}$

$\square$

Proof.

- Recall that for a query with ℓ_2 sensitivity Δ, achieving (ε', δ') with the Gaussian mechanism requires to add noise with standard deviation $\sigma' = \frac{\sqrt{2\ln(1.25/\delta')}\Delta}{\varepsilon'}$
- So with $\Delta = 2l$, $\sigma = \frac{16l\sqrt{T\ln(2/\delta)\ln(2.5T/\delta n)}}{n\varepsilon}$, each noisy gradient is $\left(\frac{n\varepsilon}{4\sqrt{2T\ln(2/\delta)}}, \frac{\delta n}{2T}\right)$-DP

$\square$

Proof.

- Recall that for a query with ℓ_2 sensitivity Δ, achieving (ε', δ') with the Gaussian mechanism requires to add noise with standard deviation $\sigma' = \frac{\sqrt{2\ln(1.25/\delta')}\Delta}{\varepsilon'}$
- So with $\Delta = 2l$, $\sigma = \frac{16l\sqrt{T\ln(2/\delta)\ln(2.5T/\delta n)}}{n\varepsilon}$, each noisy gradient is $\left(\frac{n\varepsilon}{4\sqrt{2T\ln(2/\delta)}}, \frac{\delta n}{2T}\right)$-DP
- Now, taking into account the randomness in the choice of i_t using privacy amplification by subsampling, each noisy gradient is in fact $\left(\frac{\varepsilon}{2\sqrt{2T\ln(2/\delta)}}, \frac{\delta}{2T}\right)$-DP

$\square$

Proof.

- Recall that for a query with ℓ_2 sensitivity Δ, achieving (ε', δ') with the Gaussian mechanism requires to add noise with standard deviation $\sigma' = \frac{\sqrt{2\ln(1.25/\delta')}\Delta}{\varepsilon'}$
- So with $\Delta = 2l$, $\sigma = \frac{16l\sqrt{T\ln(2/\delta)\ln(2.5T/\delta n)}}{n\varepsilon}$, each noisy gradient is $\left(\frac{n\varepsilon}{4\sqrt{2T\ln(2/\delta)}}, \frac{\delta n}{2T}\right)$-DP
- Now, taking into account the randomness in the choice of i_t using privacy amplification by subsampling, each noisy gradient is in fact $\left(\frac{\varepsilon}{2\sqrt{2T\ln(2/\delta)}}, \frac{\delta}{2T}\right)$-DP
- DP-SGD can be seen as an adaptive composition of T DP mechanisms

$\square$

Proof.

- Recall that for a query with ℓ_2 sensitivity Δ, achieving (ε', δ') with the Gaussian mechanism requires to add noise with standard deviation $\sigma' = \frac{\sqrt{2\ln(1.25/\delta')}\Delta}{\varepsilon'}$
- So with $\Delta = 2l$, $\sigma = \frac{16l\sqrt{T\ln(2/\delta)\ln(2.5T/\delta n)}}{n\varepsilon}$, each noisy gradient is $\left(\frac{n\varepsilon}{4\sqrt{2T\ln(2/\delta)}}, \frac{\delta n}{2T}\right)$-DP
- Now, taking into account the randomness in the choice of i_t using privacy amplification by subsampling, each noisy gradient is in fact $\left(\frac{\varepsilon}{2\sqrt{2T\ln(2/\delta)}}, \frac{\delta}{2T}\right)$-DP
- DP-SGD can be seen as an adaptive composition of T DP mechanisms
- By advanced composition, we know that T compositions of an $(\varepsilon_1, \delta_1)$-DP algorithm satisfies $(2\sqrt{2T\ln(1/\delta')}\varepsilon_1, T\delta_1 + \delta')$-DP for any $\delta' > 0$

$\square$

Proof.

- Recall that for a query with ℓ_2 sensitivity Δ, achieving (ε', δ') with the Gaussian mechanism requires to add noise with standard deviation $\sigma' = \frac{\sqrt{2\ln(1.25/\delta')}\Delta}{\varepsilon'}$
- So with $\Delta = 2l$, $\sigma = \frac{16l\sqrt{T\ln(2/\delta)\ln(2.5T/\delta n)}}{n\varepsilon}$, each noisy gradient is $\left(\frac{n\varepsilon}{4\sqrt{2T\ln(2/\delta)}}, \frac{\delta n}{2T}\right)$-DP
- Now, taking into account the randomness in the choice of i_t using privacy amplification by subsampling, each noisy gradient is in fact $\left(\frac{\varepsilon}{2\sqrt{2T\ln(2/\delta)}}, \frac{\delta}{2T}\right)$-DP
- DP-SGD can be seen as an adaptive composition of T DP mechanisms
- By advanced composition, we know that T compositions of an $(\varepsilon_1, \delta_1)$-DP algorithm satisfies $(2\sqrt{2T\ln(1/\delta')}\varepsilon_1, T\delta_1 + \delta')$-DP for any $\delta' > 0$
- Applying this formula with $\delta' = \delta/2$ gives that DP-SGD satisfies (ε, δ)-DP

$\square$

Theorem (Utility guarantees for DP-SGD [Bassily et al., 2014])

Let Θ be a convex domain of diameter bounded by R, and let the loss function L be convex and l-Lipschitz over Θ. For $T = n^2$ and $\gamma_t = O(R/\sqrt{t})$, DP-SGD guarantees:

$$\mathbb{E}[F(\theta^{(T)})] - \min_{\theta \in \Theta} F(\theta) \leq O\left(\frac{lR\sqrt{p \ln(1/\delta)} \ln^{3/2}(n/\delta)}{n\varepsilon}\right).$$

- Proof: plug variance of gradients in standard SGD analysis [Shamir and Zhang, 2013], and set T to balance optimization and privacy errors

Theorem (Utility guarantees for DP-SGD [Bassily et al., 2014])

Let Θ be a convex domain of diameter bounded by R, and let the loss function L be convex and l-Lipschitz over Θ. For $T = n^2$ and $\gamma_t = O(R/\sqrt{t})$, DP-SGD guarantees:

$$\mathbb{E}[F(\theta^{(T)})] - \min_{\theta \in \Theta} F(\theta) \leq O\left(\frac{lR\sqrt{p \ln(1/\delta)} \ln^{3/2}(n/\delta)}{n\varepsilon}\right).$$

- Proof: plug variance of gradients in standard SGD analysis [Shamir and Zhang, 2013], and set T to balance optimization and privacy errors
- Utility gap w.r.t. the non-private model is $\widetilde{O}(\sqrt{p}/\epsilon n)$: it reduces with the number of training points but increases with the dimension

Theorem (Utility guarantees for DP-SGD [Bassily et al., 2014])

Let Θ be a convex domain of diameter bounded by R, and let the loss function L be convex and l-Lipschitz over Θ. For $T = n^2$ and $\gamma_t = O(R/\sqrt{t})$, DP-SGD guarantees:

$$\mathbb{E}[F(\theta^{(T)})] - \min_{\theta \in \Theta} F(\theta) \leq O\left(\frac{lR\sqrt{p\ln(1/\delta)}\ln^{3/2}(n/\delta)}{n\varepsilon}\right).$$

- Proof: plug variance of gradients in standard SGD analysis [Shamir and Zhang, 2013], and set T to balance optimization and privacy errors
- Utility gap w.r.t. the non-private model is $\widetilde{O}(\sqrt{p}/\epsilon n)$: it reduces with the number of training points but increases with the dimension
- This gap is worst-case optimal (i.e., there exists an instance of the problem for which no DP algorithm can do better)

- For the mini-batch version, the same analysis applies with minor modifications

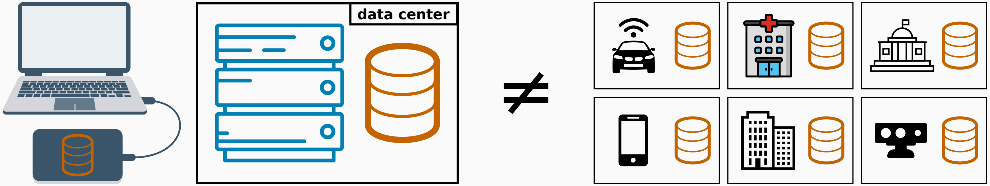
- For the mini-batch version, the same analysis applies with minor modifications
- One can readily use any data-independent regularization

- For the mini-batch version, the same analysis applies with minor modifications
- One can readily use any data-independent regularization
- If the loss is only sub-differentiable (e.g., hinge loss, ReLU), one can use a subgradient instead of the gradient

- For the mini-batch version, the same analysis applies with minor modifications
- One can readily use any data-independent regularization
- If the loss is only sub-differentiable (e.g., hinge loss, ReLU), one can use a subgradient instead of the gradient
- If the loss is non-Lipschitz (or the constant is hard to bound as in deep neural nets), one can use gradient clipping *before* adding the noise [Abadi et al., 2016]

Introduction to Federated Learning

- In the real world data is often decentralized across different parties



- Data may be considered too sensitive to be shared (e.g., due to legal restrictions, intellectual property rights, or because it provides a competitive advantage)
- Inferior performance and/or biased results if each party learns independently

Federated Learning (FL) aims to
collaboratively train ML models
while keeping the data decentralized

Federated Learning (FL) aims to
collaboratively train ML models
while keeping the data decentralized

- FL is a booming topic
  - Term first coined in 2016; more than 1,000 papers in first half of 2020 alone[1]
  - Several open-source libraries under development: PySyft, Flower, Fed-BioMed...
  - First real-world deployments by companies and researchers

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

Federated Learning (FL) aims to
collaboratively train ML models
while keeping the data decentralized

- FL is a booming topic
  - Term first coined in 2016; more than 1,000 papers in first half of 2020 alone[1]
  - Several open-source libraries under development: PySyft, Flower, Fed-BioMed...
  - First real-world deployments by companies and researchers
- FL is multidisciplinary: involves ML, optimization, statistics, privacy & security, networks, systems...

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

Distributed learning

- Data is centrally stored (e.g., in a data center)
- The goal is to train faster $\rightarrow$ distribute data uniformly at random across workers

Distributed learning

- Data is centrally stored (e.g., in a data center)
- The goal is to train faster $\rightarrow$ distribute data uniformly at random across workers

Federated Learning

- Data is naturally distributed $\rightarrow$ local datasets are heterogeneous (not iid, imbalance)

Distributed learning

- Data is centrally stored (e.g., in a data center)
- The goal is to train faster $\rightarrow$ distribute data uniformly at random across workers

Federated Learning

- Data is naturally distributed $\rightarrow$ local datasets are heterogeneous (not iid, imbalance)
- Data may be sensitive $\rightarrow$ need to enforce robust privacy constraints

KEY DIFFERENCES WITH DISTRIBUTED LEARNING

Distributed learning

- Data is centrally stored (e.g., in a data center)
- The goal is to train faster $\rightarrow$ distribute data uniformly at random across workers

Federated Learning

- Data is naturally distributed $\rightarrow$ local datasets are heterogeneous (not iid, imbalance)
- Data may be sensitive $\rightarrow$ need to enforce robust privacy constraints
- Participants may be unreliable, unavailable (with possible time/space correlations)

KEY DIFFERENCES WITH DISTRIBUTED LEARNING

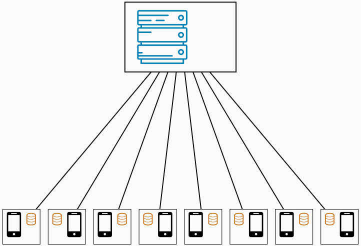
Distributed learning

- Data is centrally stored (e.g., in a data center)
- The goal is to train faster → distribute data uniformly at random across workers

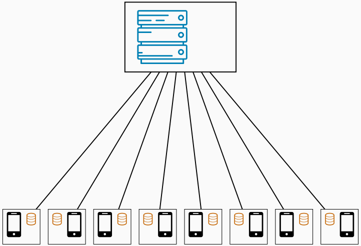
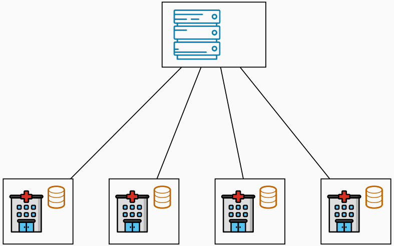
Federated Learning

- Data is naturally distributed → local datasets are heterogeneous (not iid, imbalance)
- Data may be sensitive → need to enforce robust privacy constraints
- Participants may be unreliable, unavailable (with possible time/space correlations)
- Participants may be malicious
- ...

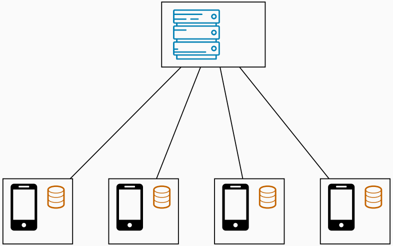
Cross-device FL



- 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

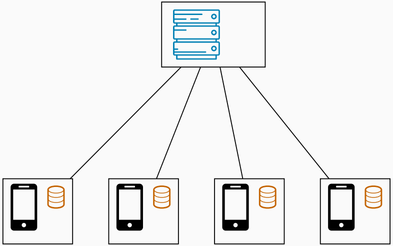
Cross-device FL	Cross-silo FL
	
• Massive number of parties (up to 10^{10})	• 2-100 parties
• Small dataset per party (could be size 1)	• Medium to large dataset per party
• Limited availability and reliability	• Reliable parties, almost always available
• Some parties may be malicious	• Parties are typically honest

Server-orchestrated FL



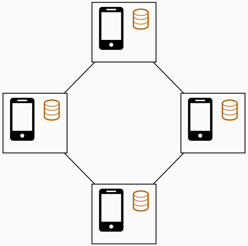
- Server-client communication
- Global coordination, global aggregation
- Server is a single point of failure and may become a bottleneck

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

- We consider a set of K parties (also called users or clients)

CLASSIC FL PROBLEM FORMULATION

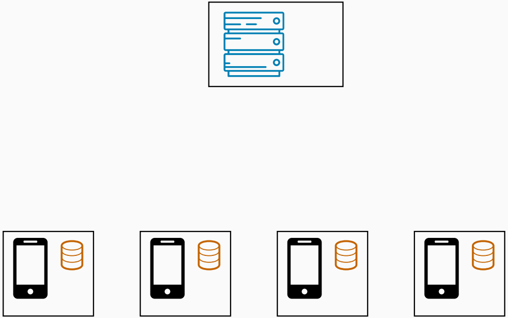
- We consider a set of K parties (also called users or clients)
- Each party k holds a dataset $\mathcal{D}_k$

- We consider a set of K parties (also called users or clients)
- Each party k holds a dataset $\mathcal{D}_k$
- We want to solve problems of the form $\min_{\theta \in \mathbb{R}^p} F(\theta; \mathcal{D})$ where:

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

- This is the empirical risk minimization problem considered before, but we now want to solve it in a federated manner

A BASELINE FL ALGORITHM: FEDAVG [McMahan et al., 2017]



Algorithm FedAvg (server-side)

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)

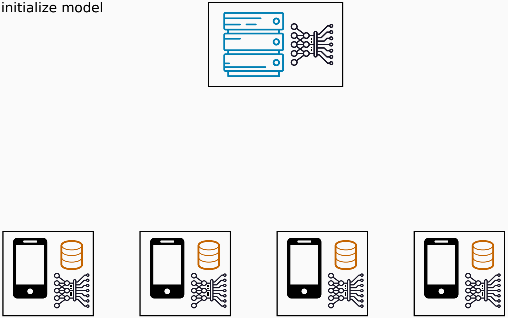
for each local step $1, \dots, L$ **do**

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

  $\theta \leftarrow \theta - \frac{\eta}{B} \sum_{d \in \mathcal{B}} \nabla f(\theta; d)$

send θ to server

A BASELINE FL ALGORITHM: FEDAVG [McMahan et al., 2017]



Algorithm FedAvg (server-side)

initialize θ

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**

    $\theta_k \leftarrow \text{ClientUpdate}(k, \theta)$

  $\theta \leftarrow \sum_{k \in \mathcal{S}_t} \frac{n_k}{n} \theta_k$

Algorithm ClientUpdate(k, θ)

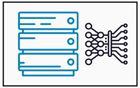
for each local step $1, \ldots, L$ **do**

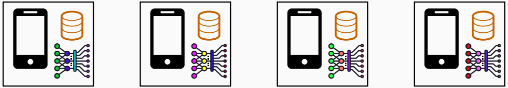
  $\mathcal{B} \leftarrow$ mini-batch of B examples from $\mathcal{D}_k$

  $\theta \leftarrow \theta - \frac{\eta}{B} \sum_{d \in \mathcal{B}} \nabla f(\theta; d)$

send θ to server

each party makes an update
using its local dataset





Algorithm FedAvg (server-side)

initialize θ

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**

    $\theta_k \leftarrow \text{ClientUpdate}(k, \theta)$

  $\theta \leftarrow \sum_{k \in \mathcal{S}_t} \frac{n_k}{n} \theta_k$

Algorithm ClientUpdate(k, θ)

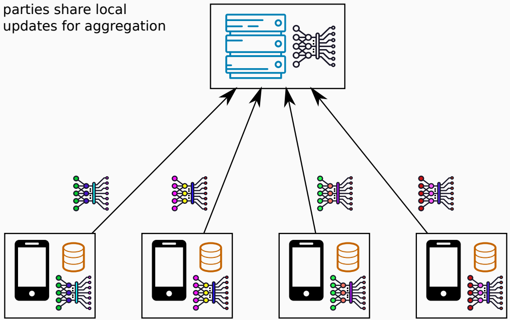
for each local step $1, \ldots, L$ **do**

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

  $\theta \leftarrow \theta - \frac{\eta}{B} \sum_{d \in \mathcal{B}} \nabla f(\theta; d)$

send θ to server

A BASELINE FL ALGORITHM: FEDAVG [McMahan et al., 2017]

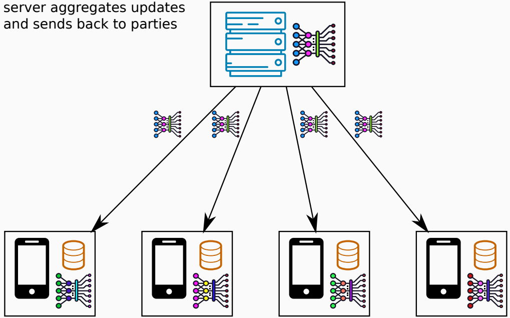


Algorithm FedAvg (server-side)

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**
    $\theta_k \leftarrow$ ClientUpdate(k, θ)
  $\theta \leftarrow \sum_{k \in \mathcal{S}_t} \frac{n_k}{n} \theta_k$

Algorithm ClientUpdate(k, θ)

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



Algorithm FedAvg (server-side)

initialize θ

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**

$\theta_k \leftarrow \text{ClientUpdate}(k, \theta)$

$\theta \leftarrow \sum_{k \in \mathcal{S}_t} \frac{n_k}{n} \theta_k$

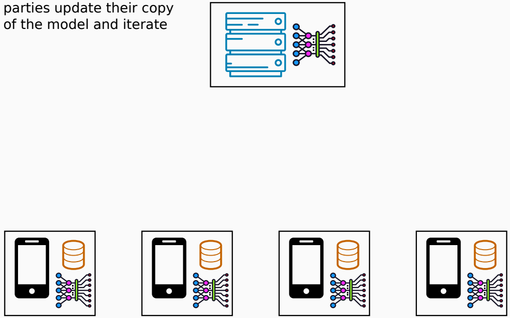
Algorithm ClientUpdate(k, θ)

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

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

$\theta \leftarrow \theta - \frac{\eta}{B} \sum_{d \in \mathcal{B}} \nabla f(\theta; d)$

send θ to server



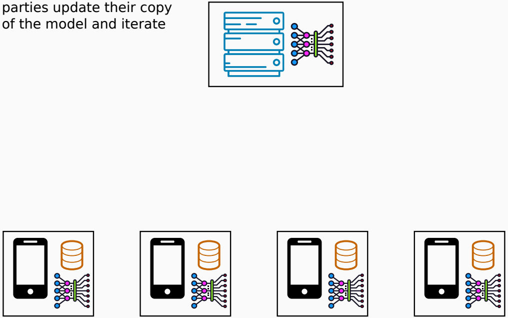
Algorithm FedAvg (server-side)

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)

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

A BASELINE FL ALGORITHM: FEDAVG [McMahan et al., 2017]



Algorithm FedAvg (server-side)

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)

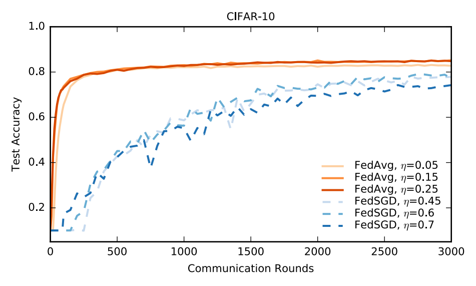
for each local step $1, \dots, L$ **do**

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

  $\theta \leftarrow \theta - \frac{\eta}{B} \sum_{d \in \mathcal{B}} \nabla f(\theta; d)$

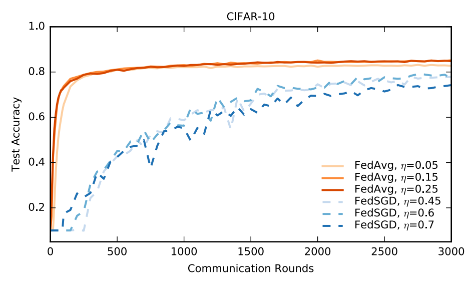
send θ to server

- $L > 1$ allows to reduce the number of communication rounds



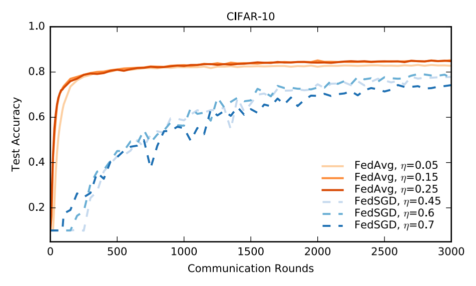
- For $L = 1$ and $\rho = 1$, FedAvg is equivalent to classic parallel SGD: updates are aggregated and the model synchronized at each step

A BASELINE FL ALGORITHM: FEDAVG [McMahan et al., 2017]



- For $L = 1$ and $\rho = 1$, FedAvg is equivalent to classic parallel SGD: updates are aggregated and the model synchronized at each step
- 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)

A BASELINE FL ALGORITHM: FEDAVG [McMahan et al., 2017]



- For $L = 1$ and $\rho = 1$, FedAvg is equivalent to classic parallel SGD: updates are aggregated and the model synchronized at each step
- 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)
- Convergence to the optimal model can be guaranteed for i.i.d. data [Stich, 2019] [Woodworth et al., 2020] but issues arise with heterogeneous data (more on this later)

FULLY DECENTRALIZED VARIANTS

- We can derive algorithms similar to FedAvg for the fully decentralized setting, where parties do not rely on a server for aggregating updates

FULLY DECENTRALIZED VARIANTS

- We can derive algorithms similar to FedAvg for the fully decentralized setting, where parties do not rely on a server for aggregating updates
- Let $G = (\{1, \ldots, K\}, E)$ be a connected undirected graph where nodes are parties and an edge $\{k, l\} \in E$ indicates that k and l can exchange messages

- We can derive algorithms similar to FedAvg for the fully decentralized setting, where parties do not rely on a server for aggregating updates
- Let $G = (\{1, \ldots, K\}, E)$ 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 gossip matrix: symmetric, doubly stochastic matrix such that $W_{k,l} = 0$ if and only if $\{k, l\} \notin E$

- 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 gossip matrix: 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:

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

GOSSIP-BASED DECENTRALIZED SGD [Lian et al., 2017, Koloskova et al., 2020b]

Algorithm Gossip-based decentralized SGD (run by party k)

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

initialize $\theta_k^{(0)}$

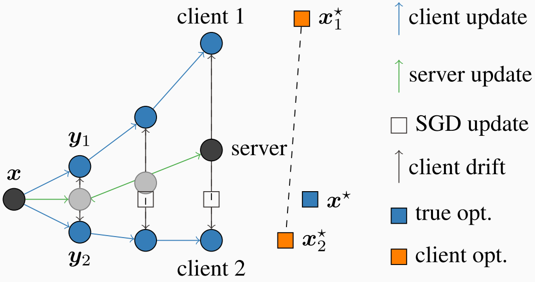
for each round $t = 0, 1, \dots$ **do**

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

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

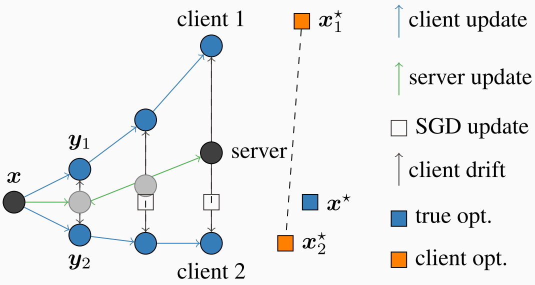
$\theta_k^{(t+1)} \leftarrow \sum_{l\in\mathcal{N}_k^{(t)}} W_{k,l}^{(t)} \theta_l^{(t+\frac{1}{2})}$

- The algorithm 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)



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

- When local datasets are heterogeneous, FedAvg suffers from client drift



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

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

THEORETICAL CONVERGENCE RATES FOR FEDAVG

- Analyzing the convergence rate of FL algorithms on heterogeneous data requires some assumption about how the local cost functions $F_1, \ldots, F_k$ 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, $c = c_1, \dots, c_K = 0$
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**
    $(\Delta\theta_k, \Delta c_k) \leftarrow$ 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 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
$c_k \leftarrow c_k^+$

SCAFFOLD: CORRECTING LOCAL UPDATES [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 $c_1, \ldots, c_K$ approximate an ideal unbiased update

Algorithm Scaffold (server-side)

Parameters: client sampling rate ρ, global learning rate η_g

initialize θ, $c = c_1, \dots, c_K = 0$
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**
    $(\Delta\theta_k, \Delta c_k) \leftarrow$ ClientUpdate (k, θ, c)
  $\theta \leftarrow \theta + \frac{\eta_g}{m} \sum_{k \in \mathcal{S}_t} \Delta\theta_k$
  $c \leftarrow c + \frac{1}{K} \sum_{k \in \mathcal{S}_t} \Delta c_k$

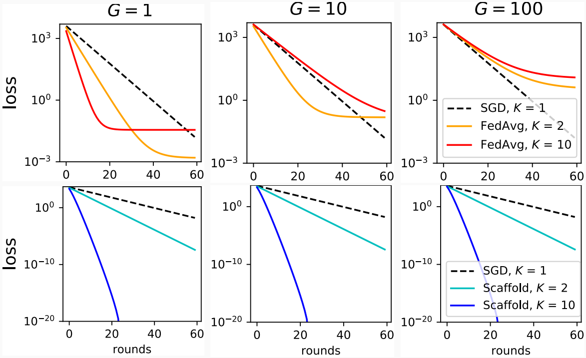
Algorithm ClientUpdate(k, θ, c)

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

Initialize $\theta_k \leftarrow \theta$
for each local step $1, \dots, L$ **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 $c_1, \dots, c_K$ approximate an ideal unbiased update
- Can show convergence rates which beat parallel SGD

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



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

- Learning from heterogeneous data is difficult/slow because each party wants the model to go in a particular direction

- Learning from heterogeneous 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

- Learning from heterogeneous 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 heterogeneous data is thus to lift the requirement that the learned model should be the same for all parties ("one size fits all")

- Learning from heterogeneous 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 heterogeneous 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:

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

PERSONALIZED MODELS FROM A "META" MODEL

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

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

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

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

- [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)

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

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

- 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]

- Inspired by multi-task learning, [Smith et al., 2017, Vanhaesebrouck et al., 2017] propose to regularize personalized models using (learned) relationships between tasks:

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

- Inspired by multi-task learning, [Smith et al., 2017, Vanhaesebrouck et al., 2017] propose to regularize personalized models using (learned) relationships between tasks:

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

- It is also well suited to the fully decentralized setting, since W can be seen as a graph of relationships over parties [Vanhaesebrouck et al., 2017]

- Inspired by multi-task learning, [Smith et al., 2017, Vanhaesebrouck et al., 2017] propose to regularize personalized models using (learned) relationships between tasks:

$$F(\theta_1, \dots, \theta_K, W; \mathcal{D}) = \frac{1}{K} \sum_{k=1}^{K} F_k(\theta_k; \mathcal{D}_k) + \sum_{k<l} W_{k,l} \|\theta_k - \theta_l\|^2$$

- It is also well suited to the fully decentralized setting, since W can be seen as a graph of relationships over parties [Vanhaesebrouck et al., 2017]
- [Marfoq et al., 2021]: assume local distributions are drawn from a mixture, learn several component models and personalized weights with a Federated EM-like algorithm

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...

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]

- 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]

- 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]

- 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]
- Robustness in FL: how to mitigate poisoning attacks [Bagdasaryan et al., 2020] [Blanchard et al., 2017], how to make local computation verifiable [Sabater et al., 2020]

Differentially Private Federated Learning

- We have seen that ML models are susceptible to various attacks on data privacy, such as membership inference and reconstruction attacks

PRIVACY ISSUES IN FEDERATED LEARNING

- We have seen that ML models are susceptible to various attacks on data privacy, such as membership inference and reconstruction attacks
- Federated Learning offers an additional attack surface as the server and other parties observe model updates (not only the final model)

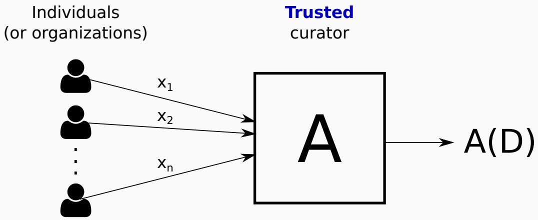
PRIVACY ISSUES IN FEDERATED LEARNING

- We have seen that ML models are susceptible to various attacks on data privacy, such as membership inference and reconstruction attacks
- Federated Learning offers an additional attack surface as the server and other parties observe model updates (not only the final model)
- This can be exploited by a participant (server or party) [Nasr et al., 2019] [Geiping et al., 2020], e.g. to reconstruct data from gradients or model updates

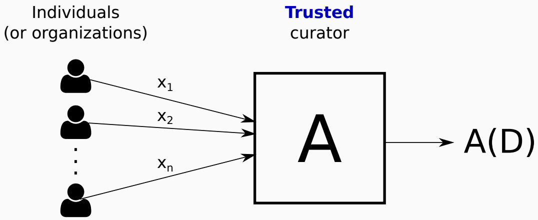

Original image


Reconstructed image

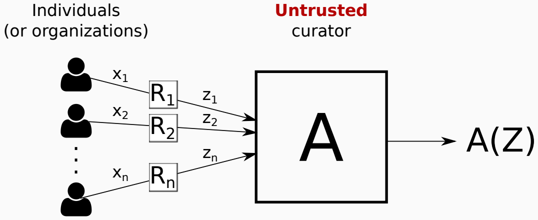
Trusted curator model (also called global model or central model):
$\mathcal{A}$ is differentially private wrt dataset $\mathcal{D}$



Trusted curator model (also called global model or central model):
$\mathcal{A}$ is differentially private wrt dataset $\mathcal{D}$



Untrusted curator model (also called local model or distributed model):
Each $\mathcal{R}_i$ is differentially private wrt record (or local dataset) x_i



- Consider the following setup:
  - A researcher wants to conduct a survey of n individuals, which consists of a single yes/no question that the researcher asks each individual

- Consider the following setup:
  - A researcher wants to conduct a survey of n individuals, which consists of a single yes/no question that the researcher asks each individual
  - The researcher is interested in the proportion of "yes" answers

PRIVATELY ANSWERING TO A SURVEY

- Consider the following setup:
  - A researcher wants to conduct a survey of n individuals, which consists of a single yes/no question that the researcher asks each individual
  - The researcher is interested in the proportion of “yes” answers
  - However the subject matter is very sensitive or embarrassing, such as “did you have sex with a prostitute this month?” or “have you ever assaulted someone?”

- Consider the following setup:
    - A researcher wants to conduct a survey of n individuals, which consists of a single yes/no question that the researcher asks each individual
    - The researcher is interested in the proportion of "yes" answers
    - However the subject matter is very sensitive or embarrassing, such as "did you have sex with a prostitute this month?" or "have you ever assaulted someone?"
- If the researcher was fully trusted to collect the true individual answers, we could use the Gaussian mechanism to make the final result differentially private

PRIVATELY ANSWERING TO A SURVEY

- Consider the following setup:
  - A researcher wants to conduct a survey of n individuals, which consists of a single yes/no question that the researcher asks each individual
  - The researcher is interested in the proportion of “yes” answers
  - However the subject matter is very sensitive or embarrassing, such as “did you have sex with a prostitute this month?” or “have you ever assaulted someone?”
- If the researcher was fully trusted to collect the true individual answers, we could use the Gaussian mechanism to make the final result differentially private
- However, this is not the case here: we can expect that just asking the individuals to reply truthfully will induce important bias in the result of the survey

PRIVATELY ANSWERING TO A SURVEY

- Consider the following setup:
  - A researcher wants to conduct a survey of n individuals, which consists of a single yes/no question that the researcher asks each individual
  - The researcher is interested in the proportion of "yes" answers
  - However the subject matter is very sensitive or embarrassing, such as "did you have sex with a prostitute this month?" or "have you ever assaulted someone?"
- If the researcher was fully trusted to collect the true individual answers, we could use the Gaussian mechanism to make the final result differentially private
- However, this is not the case here: we can expect that just asking the individuals to reply truthfully will induce important bias in the result of the survey
- **How can we provide privacy to the participants while getting an unbiased result?**

- We denote the truthful answer of individual i by $x_i \in \{0, 1\}$ and the true proportion of "yes" by $Y = \frac{1}{n} \sum_{i=1}^{n} x_i$

- We denote the truthful answer of individual i by $x_i \in \{0,1\}$ and the true proportion of "yes" by $Y = \frac{1}{n}\sum_{i=1}^{n} x_i$
- Consider the following simple randomized approach: each participant answers truthfully ($z_i = x_i$) with probability p and falsely ($z_i = \neg x_i$) with probability $1-p$

- We denote the truthful answer of individual i by $x_i \in \{0,1\}$ and the true proportion of "yes" by $Y = \frac{1}{n}\sum_{i=1}^{n} x_i$
- Consider the following simple randomized approach: each participant answers truthfully ($z_i = x_i$) with probability p and falsely ($z_i = \neg x_i$) with probability $1-p$
- The expected proportion of "yes" is given by $pY + (1-p)(1-Y)$, so we can recover an unbiased estimate $\hat{Y}$ of Y by computing:

$$\hat{Y} = \frac{\frac{1}{n}\sum_{i=1}^{n} z_i + p - 1}{2p - 1}$$

- We denote the truthful answer of individual i by $x_i \in \{0, 1\}$ and the true proportion of "yes" by $Y = \frac{1}{n}\sum_{i=1}^{n} x_i$
- Consider the following simple randomized approach: each participant answers truthfully ($z_i = x_i$) with probability p and falsely ($z_i = \neg x_i$) with probability $1 - p$
- The expected proportion of "yes" is given by $pY + (1-p)(1-Y)$, so we can recover an unbiased estimate $\hat{Y}$ of Y by computing:

$$\hat{Y} = \frac{\frac{1}{n}\sum_{i=1}^{n} z_i + p - 1}{2p - 1}$$

- This approach, which dates back to [Warner, 1965], satisfies local differential privacy!

- As always, let $\mathcal{X}$ denote an abstract data domain

- As always, let $\mathcal{X}$ denote an abstract data domain
- A local randomizer $\mathcal{R} : \mathcal{X} \to \mathcal{Z}$ is a randomized function which maps an input $x \in \mathcal{X}$ to an output $z \in \mathcal{Z}$

- As always, let $\mathcal{X}$ denote an abstract data domain
- A local randomizer $\mathcal{R} : \mathcal{X} \to \mathcal{Z}$ is a randomized function which maps an input $x \in \mathcal{X}$ to an output $z \in \mathcal{Z}$

Definition (Local Differential Privacy [Kasiviswanathan et al., 2008, Duchi et al., 2013])

Let $\varepsilon > 0$ and $\delta \in (0,1)$. A local randomizer algorithm $\mathcal{R}$ is (ε, δ)-locally differentially private (LDP) if for all $x, x' \in \mathcal{X}$ and any possible $z \in \mathcal{Z}$:

$$\Pr[\mathcal{R}(x) = z] \leq e^{\varepsilon} \Pr[\mathcal{R}(x') = z] + \delta.$$

- As always, let $\mathcal{X}$ denote an abstract data domain
- A local randomizer $\mathcal{R} : \mathcal{X} \to \mathcal{Z}$ is a randomized function which maps an input $x \in \mathcal{X}$ to an output $z \in \mathcal{Z}$

Definition (Local Differential Privacy [Kasiviswanathan et al., 2008, Duchi et al., 2013])

Let $\varepsilon > 0$ and $\delta \in (0, 1)$. A local randomizer algorithm $\mathcal{R}$ is (ε, δ)-locally differentially private (LDP) if for all $x, x' \in \mathcal{X}$ and any possible $z \in \mathcal{Z}$:

$$\Pr[\mathcal{R}(x) = z] \leq e^{\varepsilon} \Pr[\mathcal{R}(x') = z] + \delta.$$

- This is equivalent to (ε, δ)-DP for datasets of size 1!

- As always, let $\mathcal{X}$ denote an abstract data domain
- A local randomizer $\mathcal{R} : \mathcal{X} \to \mathcal{Z}$ is a randomized function which maps an input $x \in \mathcal{X}$ to an output $z \in \mathcal{Z}$

Definition (Local Differential Privacy [Kasiviswanathan et al., 2008, Duchi et al., 2013])

Let $\varepsilon > 0$ and $\delta \in (0, 1)$. A local randomizer algorithm $\mathcal{R}$ is (ε, δ)-locally differentially private (LDP) if for all $x, x' \in \mathcal{X}$ and any possible $z \in \mathcal{Z}$:

$$\Pr[\mathcal{R}(x) = z] \leq e^{\varepsilon} \Pr[\mathcal{R}(x') = z] + \delta.$$

- This is equivalent to (ε, δ)-DP for datasets of size 1!
- LDP is a much stronger model than central DP (no trusted curator)
- Indeed, LDP allows participants to have plausible deniability even if the curator is compromised: they can deny having value x on the basis of lack of evidence

- Assume a K-ary data domain $\mathcal{X} = \{v_1, \ldots, v_K\}$

Algorithm: K**-ary Randomized Response** $\mathcal{R}_{\mathrm{RR},K}(x, \varepsilon)$ [Kairouz et al., 2014]

1. Sample $b \sim \mathrm{Ber}(K/(e^{\varepsilon} + K - 1))$
2. If $b = 0$ output x, else output $y \sim \mathrm{Unif}(\mathcal{X})$

- Assume a K-ary data domain $\mathcal{X} = \{v_1, \ldots, v_K\}$

Algorithm: K**-ary Randomized Response** $\mathcal{R}_{\mathrm{RR},K}(x, \varepsilon)$ [Kairouz et al., 2014]

1. Sample $b \sim \mathrm{Ber}(K/(e^\varepsilon + K - 1))$
2. If $b = 0$ output x, else output $y \sim \mathrm{Unif}(\mathcal{X})$

- $K-$RR will output the true value w.p. $\frac{e^\varepsilon - 1}{e^\varepsilon + K - 1}$, or a random value w.p. $\frac{K}{e^\varepsilon + K - 1}$

- Assume a K-ary data domain $\mathcal{X} = \{v_1, \ldots, v_K\}$

Algorithm: K**-ary Randomized Response** $\mathcal{R}_{\mathrm{RR},K}(x, \varepsilon)$ [Kairouz et al., 2014]

1. Sample $b \sim \mathrm{Ber}(K/(e^\varepsilon + K - 1))$
2. If $b = 0$ output x, else output $y \sim \mathrm{Unif}(\mathcal{X})$

- K−RR will output the true value w.p. $\frac{e^\varepsilon - 1}{e^\varepsilon + K - 1}$, or a random value w.p. $\frac{K}{e^\varepsilon + K - 1}$
- This can be seen as a generalization of the simple binary version that we used earlier

- Assume a K-ary data domain $\mathcal{X} = \{v_1, \ldots, v_K\}$

Algorithm: K-ary Randomized Response $\mathcal{R}_{\mathrm{RR},K}(x, \varepsilon)$ [Kairouz et al., 2014]

1. Sample $b \sim \mathrm{Ber}(K/(e^\varepsilon + K - 1))$
2. If $b = 0$ output x, else output $y \sim \mathrm{Unif}(\mathcal{X})$

- K—RR will output the true value w.p. $\frac{e^\varepsilon - 1}{e^\varepsilon + K - 1}$, or a random value w.p. $\frac{K}{e^\varepsilon + K - 1}$
- This can be seen as a generalization of the simple binary version that we used earlier

Theorem (DP guarantees for K-RR mechanism)

Let $\varepsilon > 0$. The K-ary randomized response mechanism $\mathcal{R}_{RR,K}(\cdot, \varepsilon)$ satisfies ε-LDP.

Proof.

- For any $x, x' \in \mathcal{X}$ and $z \in \mathcal{Z}$, we want to show that $\frac{\Pr[\mathcal{R}_{\mathrm{RR},K}(x)=z]}{\Pr[\mathcal{R}_{\mathrm{RR},K}(x')=z]} \leq e^{\varepsilon}$

$\square$

Proof.

- For any $x, x' \in \mathcal{X}$ and $z \in \mathcal{Z}$, we want to show that $\frac{\Pr[\mathcal{R}_{\mathrm{RR},K}(x)=z]}{\Pr[\mathcal{R}_{\mathrm{RR},K}(x')=z]} \leq e^{\varepsilon}$
- If $x \neq z \wedge x' \neq z$ or $x = x' = z$, then clearly $\Pr[\mathcal{R}_{\mathrm{RR},K}(x) = z] = \Pr[\mathcal{R}_{\mathrm{RR},K}(x') = z]$

$\square$

Proof.

- For any $x, x' \in \mathcal{X}$ and $z \in \mathcal{Z}$, we want to show that $\frac{\Pr[\mathcal{R}_{\mathrm{RR},K}(x)=z]}{\Pr[\mathcal{R}_{\mathrm{RR},K}(x')=z]} \leq e^{\varepsilon}$
- If $x \neq z \wedge x' \neq z$ or $x = x' = z$, then clearly $\Pr[\mathcal{R}_{\mathrm{RR},K}(x) = z] = \Pr[\mathcal{R}_{\mathrm{RR},K}(x') = z]$
- We thus focus on the case $x = z$ and $x' \neq z$. We have:

$$\Pr[\mathcal{R}_{\mathrm{RR},K}(x) = z] = \frac{e^{\varepsilon} - 1}{e^{\varepsilon} + K - 1} + \frac{K}{K(e^{\varepsilon} + K - 1)} = \frac{e^{\varepsilon}}{e^{\varepsilon} + K - 1}$$

$\square$

Proof.

- For any $x, x' \in \mathcal{X}$ and $z \in \mathcal{Z}$, we want to show that $\frac{\Pr[\mathcal{R}_{\mathrm{RR},K}(x)=z]}{\Pr[\mathcal{R}_{\mathrm{RR},K}(x')=z]} \leq e^{\varepsilon}$
- If $x \neq z \wedge x' \neq z$ or $x = x' = z$, then clearly $\Pr[\mathcal{R}_{\mathrm{RR},K}(x) = z] = \Pr[\mathcal{R}_{\mathrm{RR},K}(x') = z]$
- We thus focus on the case $x = z$ and $x' \neq z$. We have:

$$\Pr[\mathcal{R}_{\mathrm{RR},K}(x) = z] = \frac{e^{\varepsilon} - 1}{e^{\varepsilon} + K - 1} + \frac{K}{K(e^{\varepsilon} + K - 1)} = \frac{e^{\varepsilon}}{e^{\varepsilon} + K - 1}$$

$$\Pr[\mathcal{R}_{\mathrm{RR},K}(x') = z] = \frac{1}{e^{\varepsilon} + K - 1}$$

$\square$

Proof.

- For any $x, x' \in \mathcal{X}$ and $z \in \mathcal{Z}$, we want to show that $\frac{\Pr[\mathcal{R}_{\mathrm{RR},K}(x)=z]}{\Pr[\mathcal{R}_{\mathrm{RR},K}(x')=z]} \leq e^{\varepsilon}$
- If $x \neq z \wedge x' \neq z$ or $x = x' = z$, then clearly $\Pr[\mathcal{R}_{\mathrm{RR},K}(x) = z] = \Pr[\mathcal{R}_{\mathrm{RR},K}(x') = z]$
- We thus focus on the case $x = z$ and $x' \neq z$. We have:

$$\Pr[\mathcal{R}_{\mathrm{RR},K}(x) = z] = \frac{e^{\varepsilon} - 1}{e^{\varepsilon} + K - 1} + \frac{K}{K(e^{\varepsilon} + K - 1)} = \frac{e^{\varepsilon}}{e^{\varepsilon} + K - 1}$$

$$\Pr[\mathcal{R}_{\mathrm{RR},K}(x') = z] = \frac{1}{e^{\varepsilon} + K - 1}$$

- Taking the ratio gives us the desired result

$\square$

- Let $h = (h_1, \ldots, h_K)$ denote the histogram of the private data: $h_k = \frac{1}{n} \sum_{i=1}^{n} \mathbb{I}[x_i = v_k]$

K-ARY RANDOMIZED RESPONSE: UTILITY GUARANTEES

- Let $h = (h_1, \ldots, h_K)$ denote the histogram of the private data: $h_k = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}[x_i = v_k]$
- Letting $p = \frac{e^\varepsilon - 1}{e^\varepsilon + K - 1}$, K-RR allows us to obtain an unbiased estimate $\hat{h}$ of h by setting

$$\hat{h}_k = \frac{\left(\frac{1}{n}\sum_{i=1}^{n} \mathbb{I}[z_i = v_k]\right) - \frac{1-p}{K}}{p} = \frac{\left(\frac{1}{n}\sum_{i=1}^{n} \mathbb{I}[z_i = v_k]\right)(e^\varepsilon + K - 1) - 1}{e^\varepsilon - 1}$$

- Let $h = (h_1, \dots, h_K)$ denote the histogram of the private data: $h_k = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}[x_i = v_k]$
- Letting $p = \frac{e^\varepsilon - 1}{e^\varepsilon + K - 1}$, K-RR allows us to obtain an unbiased estimate $\hat{h}$ of h by setting

$$\hat{h}_k = \frac{\left(\frac{1}{n}\sum_{i=1}^{n} \mathbb{I}[z_i = v_k]\right) - \frac{1-p}{K}}{p} = \frac{\left(\frac{1}{n}\sum_{i=1}^{n} \mathbb{I}[z_i = v_k]\right)(e^\varepsilon + K - 1) - 1}{e^\varepsilon - 1}$$

Theorem (ℓ_2 error of K-ary randomized response)

Let $\varepsilon > 0$. The histogram $\hat{h}$ obtained using the K-ary randomized response mechanism satisfies for any $k \in \{1, \dots, K\}$:

$$\mathbb{E}[(\hat{h}_k - h_k)^2] = \frac{K - 2 + e^\varepsilon}{n(e^\varepsilon - 1)^2}.$$

- Let f be a public function from $\mathcal{X}$ to a bounded numeric range (say $f : \mathcal{X} \to [0, 1]$)
- We want to compute an averaging query $\bar{f} = \frac{1}{n} \sum_{i=1}^{n} f(x_i)$

- Let f be a public function from $\mathcal{X}$ to a bounded numeric range (say $f : \mathcal{X} \to [0, 1]$)
- We want to compute an averaging query $\bar{f} = \frac{1}{n} \sum_{i=1}^{n} f(x_i)$
- How to do this in the LDP setting?

- Let f be a public function from $\mathcal{X}$ to a bounded numeric range (say $f : \mathcal{X} \to [0, 1]$)
- We want to compute an averaging query $\bar{f} = \frac{1}{n} \sum_{i=1}^{n} f(x_i)$
- How to do this in the LDP setting?
- We can readily use the Gaussian mechanism!

- Let f be a public function from $\mathcal{X}$ to a bounded numeric range (say $f : \mathcal{X} \to [0, 1]$)
- We want to compute an averaging query $\bar{f} = \frac{1}{n} \sum_{i=1}^{n} f(x_i)$
- How to do this in the LDP setting?
- We can readily use the Gaussian mechanism!
- Indeed, seeing each input as a dataset of size 1, the sensitivity of $f(x)$ is $\Delta_2(f) = 1$

- Let f be a public function from $\mathcal{X}$ to a bounded numeric range (say $f : \mathcal{X} \to [0, 1]$)
- We want to compute an averaging query $\bar{f} = \frac{1}{n} \sum_{i=1}^{n} f(x_i)$
- How to do this in the LDP setting?
- We can readily use the Gaussian mechanism!
- Indeed, seeing each input as a dataset of size 1, the sensitivity of $f(x)$ is $\Delta_2(f) = 1$
- With the Gaussian mechanism, we thus get an estimate of $\bar{f}$ with variance $\frac{2 \log(1.25/\delta)}{n\varepsilon^2}$

THE COST OF THE LOCAL MODEL

- As one can expect, there is a large utility gap between the central and the local model of DP: it is typically a factor of $O(1/\sqrt{n})$ in ℓ_1 error (or $O(1/n)$ in ℓ_2 error)

- As one can expect, there is a large utility gap between the central and the local model of DP: it is typically a factor of $O(1/\sqrt{n})$ in ℓ_1 error (or $O(1/n)$ in ℓ_2 error)
- Example 1: histograms
  - In the local model, we have seen that $\mathbb{E}[(\hat{h}_k - h_k)^2] = O(1/n)$
  - In the central model, we can compute the exact $h_k = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}[x_i = v_k]$ and add Gaussian noise calibrated to its ℓ_2 sensitivity $1/n$, hence we get $\mathbb{E}[(\hat{h}_k - h_k)^2] = O(1/n^2)$

- As one can expect, there is a large utility gap between the central and the local model of DP: it is typically a factor of $O(1/\sqrt{n})$ in ℓ_1 error (or $O(1/n)$ in ℓ_2 error)
- Example 1: histograms
    - In the local model, we have seen that $\mathbb{E}[(\hat{h}_k - h_k)^2] = O(1/n)$
    - In the central model, we can compute the exact $h_k = \frac{1}{n}\sum_{i=1}^n \mathbb{I}[x_i = v_k]$ and add Gaussian noise calibrated to its ℓ_2 sensitivity $1/n$, hence we get $\mathbb{E}[(\hat{h}_k - h_k)^2] = O(1/n^2)$
- Example 2: averaging queries
    - In the local model, we have seen that we get a variance of $O(1/n)$
    - In the central model, we can compute the exact $\bar{f}$ and add Gaussian noise calibrated to its ℓ_2 sensitivity $\Delta_1(\bar{f}) = 1/n$, hence we get a variance of $O(1/n^2)$

THE COST OF THE LOCAL MODEL

- As one can expect, there is a large utility gap between the central and the local model of DP: it is typically a factor of $O(1/\sqrt{n})$ in ℓ_1 error (or $O(1/n)$ in ℓ_2 error)
- Example 1: histograms
  - In the local model, we have seen that $\mathbb{E}[(\hat{h}_k - h_k)^2] = O(1/n)$
  - In the central model, we can compute the exact $h_k = \frac{1}{n}\sum_{i=1}^{n} \mathbb{I}[x_i = v_k]$ and add Gaussian noise calibrated to its ℓ_2 sensitivity $1/n$, hence we get $\mathbb{E}[(\hat{h}_k - h_k)^2] = O(1/n^2)$
- Example 2: averaging queries
  - In the local model, we have seen that we get a variance of $O(1/n)$
  - In the central model, we can compute the exact $\bar{f}$ and add Gaussian noise calibrated to its ℓ_2 sensitivity $\Delta_1(\bar{f}) = 1/n$, hence we get a variance of $O(1/n^2)$
- This gap is known to be unavoidable for some queries like averaging [Chan et al., 2012]

THE COST OF THE LOCAL MODEL

- As one can expect, there is a large utility gap between the central and the local model of DP: it is typically a factor of $O(1/\sqrt{n})$ in ℓ_1 error (or $O(1/n)$ in ℓ_2 error)
- Example 1: histograms
  - In the local model, we have seen that $\mathbb{E}[(\hat{h}_k - h_k)^2] = O(1/n)$
  - In the central model, we can compute the exact $h_k = \frac{1}{n}\sum_{i=1}^n \mathbb{I}[x_i = v_k]$ and add Gaussian noise calibrated to its ℓ_2 sensitivity $1/n$, hence we get $\mathbb{E}[(\hat{h}_k - h_k)^2] = O(1/n^2)$
- Example 2: averaging queries
  - In the local model, we have seen that we get a variance of $O(1/n)$
  - In the central model, we can compute the exact $\bar{f}$ and add Gaussian noise calibrated to its ℓ_2 sensitivity $\Delta_1(\bar{f}) = 1/n$, hence we get a variance of $O(1/n^2)$
- This gap is known to be unavoidable for some queries like averaging [Chan et al., 2012]
- This restricts the usefulness of LDP to applications where n is very large and motivates the exploration of intermediate trust models

A KEY FUNCTIONALITY: DP AGGREGATION

- Most FL algorithms with a server follow the same high-level structure:

  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

A KEY FUNCTIONALITY: DP AGGREGATION

- Most FL algorithms with a server follow the same high-level structure:

  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

- Observe that:

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

- **Differentially private aggregation:** given a private value $\theta_k \in [0, 1]$ 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

- Central model: trusted server adds Gaussian noise to the average θ^{avg}

- Central model: trusted server adds Gaussian noise to the average θ^{avg}
- Local model: each party k adds Gaussian noise to its own value θ_k before sharing it

- Central model: trusted server adds Gaussian noise to the average θ^{avg}
- Local model: each party k adds Gaussian noise to its own value θ_k before sharing it
- For a fixed DP guarantee, the error is $O(\sqrt{K})$ larger in the local case

EXISTING APPROACHES TO DP AGGREGATION

- Central model: trusted server adds Gaussian noise to the average θ^{avg}
- Local model: each party k adds Gaussian noise to its own value θ_k before sharing it
- For a fixed DP guarantee, the error is $O(\sqrt{K})$ larger in the local 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 without introducing a trusted server, but their practical implementation poses important challenges when K is large

- Assume that pairs of parties are able to exchange encrypted messages (the server may act as relay): this can be achieved e.g. through a public key infrastructure

- Assume that pairs of parties are able to exchange encrypted messages (the server may act as relay): this can be achieved e.g. through a public key infrastructure

Algorithm GOPA protocol [Sabater et al., 2020]

Each party k generates independent Gaussian noise η_k

- Assume that pairs of parties are able to exchange encrypted messages (the server may act as relay): this can be achieved e.g. through a public key infrastructure

Algorithm GOPA protocol [Sabater et al., 2020]

Each party k generates independent Gaussian noise η_k
Each party k selects a random set of m other parties

- Assume that pairs of parties are able to exchange encrypted messages (the server may act as relay): this can be achieved e.g. through a public key infrastructure

Algorithm GOPA protocol [Sabater et al., 2020]

Each party k generates independent Gaussian noise η_k
Each party k selects a random set of m other parties
for all selected pairs of parties $k \sim l$ **do**
    Parties k and l securely exchange pairwise-canceling Gaussian noise $\Delta_{k,l} = -\Delta_{l,k}$

- Assume that pairs of parties are able to exchange encrypted messages (the server may act as relay): this can be achieved e.g. through a public key infrastructure

---

Algorithm GOPA protocol [Sabater et al., 2020]

Each party k generates independent Gaussian noise η_k
Each party k selects a random set of m other parties
for all selected pairs of parties $k \sim l$ **do**
  Parties k and l securely exchange pairwise-canceling Gaussian noise $\Delta_{k,l} = -\Delta_{l,k}$
Each party k sends $\hat{\theta}_k = \theta_k + \sum_{k \sim l} \Delta_{k,l} + \eta_k$ to the server

---

- Assume that pairs of parties are able to exchange encrypted messages (the server may act as relay): this can be achieved e.g. through a public key infrastructure

Algorithm GOPA protocol [Sabater et al., 2020]

Each party k generates independent Gaussian noise η_k
Each party k selects a random set of m other parties
for all selected pairs of parties $k \sim l$ **do**
  Parties k and l securely exchange pairwise-canceling Gaussian noise $\Delta_{k,l} = -\Delta_{l,k}$
Each party k sends $\hat{\theta}_k = \theta_k + \sum_{k \sim l} \Delta_{k,l} + \eta_k$ to the server

- **Estimate of the average:** $\hat{\theta}^{avg} = \frac{1}{K} \sum_k \hat{\theta}_k = \theta^{avg} + \frac{1}{K} \sum_k \eta_k$

- **Adversary**: coalition of the server with a proportion $1 - \tau$ of the parties

- **Adversary**: coalition of the server with a proportion $1 - \tau$ of the parties

Theorem (Privacy of GOPA [Sabater et al., 2020], informal)

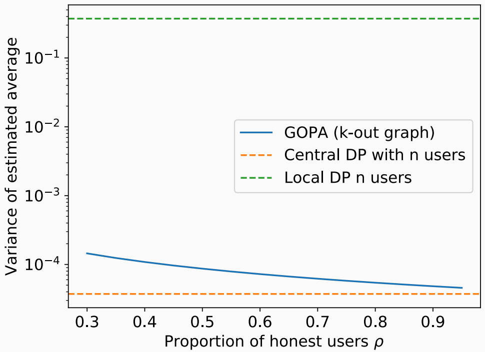
- *Let each party select $m = O(\log(\tau K)/\tau)$ other parties*
- *Set the independent noise variance so as to satisfy (ϵ, δ')-DP in the central model*
- *For large enough pairwise noise variance, GOPA is (ϵ, δ)-DP with $\delta = O(\delta')$.*

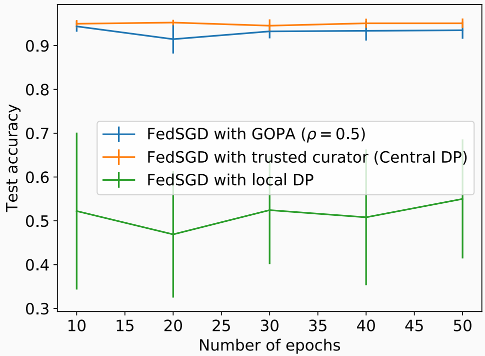
- **Adversary**: coalition of the server with a proportion $1-\tau$ of the parties

Theorem (Privacy of GOPA [Sabater et al., 2020], informal)

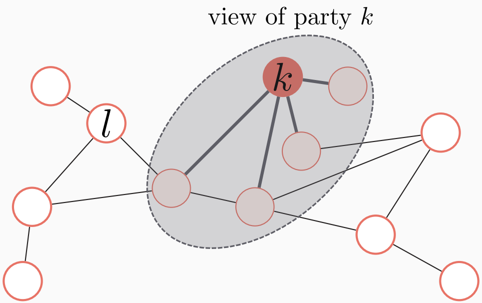
- *Let each party select $m = O(\log(\tau K)/\tau)$ other parties*
- *Set the independent noise variance so as to satisfy (ϵ, δ')-DP in the central model*
- *For large enough pairwise noise variance, GOPA is (ϵ, δ)-DP with $\delta = O(\delta')$.*

- Same utility as central DP with only logarithmic number of messages per party
- Our general result quantifies the effect of an arbitrary topology G on DP guarantees and gives practical values for the quantities above





- For reasonable proportions ρ of honest users, the variance of the estimated average produced by GOPA is similar to the trusted curator setting
- As expected, the resulting FL model also has similar accuracy



- In a fully decentralized setting, there is no server that observes all messages: each party/user k has a limited view of the system
- Folklore knowledge: “full decentralization improves privacy”. But can we formally prove stronger differential privacy guarantees?

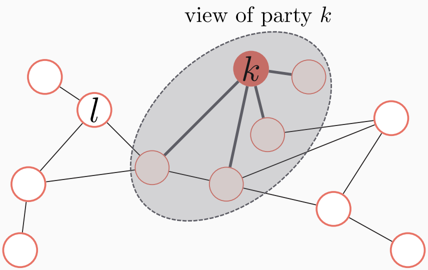
- Let $\mathcal{O}_k$ be the set of messages sent and received by party k

- Let $\mathcal{O}_k$ be the set of messages sent and received by party k

Definition (Network DP [Cyffers and Bellet, 2022])

An algorithm $\mathcal{A}$ satisfies (ϵ, δ)-network DP if for all pairs of distinct parties $k, l \in \{1, \dots, n\}$ and all pairs of datasets $\mathcal{D}, \mathcal{D}'$ that differ only in the local dataset of party l, we have:

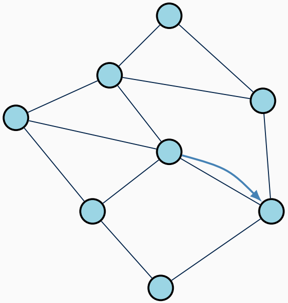
$$\Pr[\mathcal{O}_k(\mathcal{A}(\mathcal{D}))] \leq e^\epsilon \Pr[\mathcal{O}_k(\mathcal{A}(\mathcal{D}'))] + \delta.$$



- This is a relaxation of local DP: if $\mathcal{O}_k$ contains the full transcript of messages, then network DP boils down to local DP

- Consider the standard objective $F(\theta; \mathcal{D}) = \frac{1}{K}\sum_{k=1}^{K} F_k(\theta; \mathcal{D}_k)$ and a complete graph

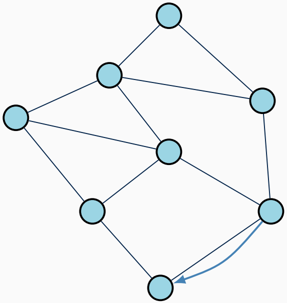
- Consider the standard objective $F(\theta; \mathcal{D}) = \frac{1}{K} \sum_{k=1}^{K} F_k(\theta; \mathcal{D}_k)$ and a complete graph
- Let us consider a fully decentralized algorithm where the model is updated sequentially by following a random walk



Algorithm Private decentralized SGD on a complete graph

Initialize model θ
for $t = 1$ to T **do**
  Current party updates θ by a gradient update with Gaussian noise
  Current party sends θ to a random party
return θ

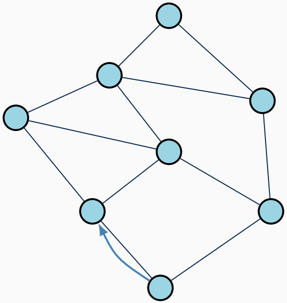
- Consider the standard objective $F(\theta; \mathcal{D}) = \frac{1}{K} \sum_{k=1}^{K} F_k(\theta; \mathcal{D}_k)$ and a complete graph
- Let us consider a fully decentralized algorithm where the model is updated sequentially by following a random walk



Algorithm Private decentralized SGD on a complete graph

Initialize model θ
for $t = 1$ to T **do**
  Current party updates θ by a gradient update with Gaussian noise
  Current party sends θ to a random party
return θ

- Consider the standard objective $F(\theta; \mathcal{D}) = \frac{1}{K} \sum_{k=1}^{K} F_k(\theta; \mathcal{D}_k)$ and a complete graph
- Let us consider a fully decentralized algorithm where the model is updated sequentially by following a random walk



Algorithm Private decentralized SGD on a complete graph

Initialize model θ
for $t = 1$ to T **do**
  Current party updates θ by a gradient update with Gaussian noise
  Current party sends θ to a random party
return θ

Theorem ([Cyffers and Bellet, 2022], informal)

To achieve a fixed (ϵ, δ)-DP guarantee with the previous algorithm, the standard deviation of the noise is $O(\sqrt{K}/\ln K)$ smaller under network DP than under local DP.

- Accounting for the limited view in fully decentralized algorithms amplifies privacy guarantees by a factor of $O(\ln K/\sqrt{K})$, nearly recovering the utility of central DP

Theorem ([Cyffers and Bellet, 2022], informal)

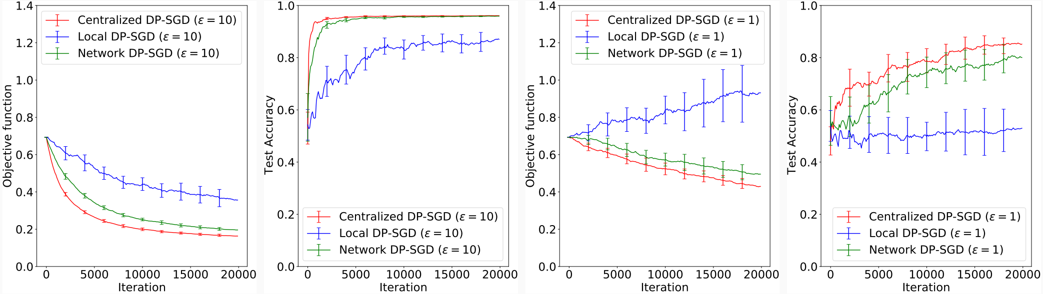
To achieve a fixed (ϵ, δ)-DP guarantee with the previous algorithm, the standard deviation of the noise is $O(\sqrt{K}/\ln K)$ smaller under network DP than under local DP.

- Accounting for the limited view in fully decentralized algorithms amplifies privacy guarantees by a factor of $O(\ln K/\sqrt{K})$, nearly recovering the utility of central DP
- The proof leverages recent results on privacy amplification by iteration [Feldman et al., 2018] and exploits the randomness of the path taken by the model

Theorem ([Cyffers and Bellet, 2022], informal)

To achieve a fixed (ϵ, δ)-DP guarantee with the previous algorithm, the standard deviation of the noise is $O(\sqrt{K}/\ln K)$ smaller under network DP than under local DP.

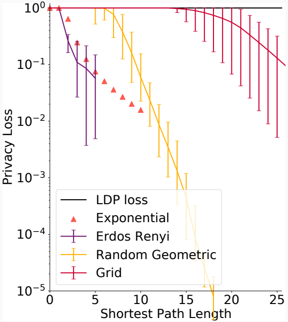
- Accounting for the limited view in fully decentralized algorithms amplifies privacy guarantees by a factor of $O(\ln K/\sqrt{K})$, nearly recovering the utility of central DP
- The proof leverages recent results on privacy amplification by iteration [Feldman et al., 2018] and exploits the randomness of the path taken by the model
- In a recent preprint [Cyffers et al., 2022] we refine network DP to capture the privacy loss across each pair of nodes and prove amplification guarantees for gossip-based algorithms on arbitrary graphs

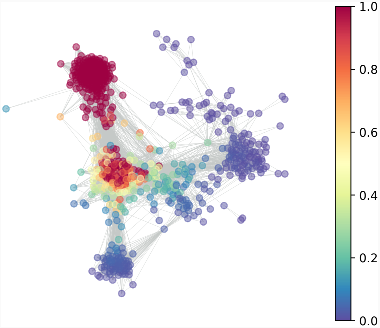


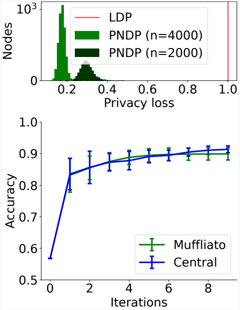
- Results are consistent with our theory: network DP-SGD significantly amplifies privacy guarantees compared to local DP-SGD

PRIVACY AMPLIFICATION FOR GOSSIP DECENTRALIZED SGD

- In a recent preprint [Cyffers et al., 2022] we refine network DP to capture the privacy loss across each pair of nodes and prove amplification guarantees for gossip-based algorithms on arbitrary graphs







Wrapping up

1. Any personal information can be sensitive, and anonymization is hard

TAKE-AWAYS OF THE COURSE

1. Any personal information can be sensitive, and anonymization is hard
2. Differential privacy provides a robust mathematical definition of privacy

TAKE-AWAYS OF THE COURSE

1. Any personal information can be sensitive, and anonymization is hard
2. Differential privacy provides a robust mathematical definition of privacy
3. Simple DP primitives can be used as basis to design complex algorithms like DP-SGD

TAKE-AWAYS OF THE COURSE

1. Any personal information can be sensitive, and anonymization is hard
2. Differential privacy provides a robust mathematical definition of privacy
3. Simple DP primitives can be used as basis to design complex algorithms like DP-SGD
4. Federated learning (FL) enables several entities to collaboratively train machine learning models without sharing their data

TAKE-AWAYS OF THE COURSE

1. Any personal information can be sensitive, and anonymization is hard
2. Differential privacy provides a robust mathematical definition of privacy
3. Simple DP primitives can be used as basis to design complex algorithms like DP-SGD
4. Federated learning (FL) enables several entities to collaboratively train machine learning models without sharing their data
5. To ensure privacy in FL with an untrusted server, DP can be deployed locally at the participants' level (LDP)

1. Any personal information can be sensitive, and anonymization is hard
2. Differential privacy provides a robust mathematical definition of privacy
3. Simple DP primitives can be used as basis to design complex algorithms like DP-SGD
4. Federated learning (FL) enables several entities to collaboratively train machine learning models without sharing their data
5. To ensure privacy in FL with an untrusted server, DP can be deployed locally at the participants' level (LDP)
6. The privacy-utility trade-off can be improved through appropriate relaxations of LDP so as to leverage crypto primitives or reap the benefits of full decentralization

SOME OPEN PROBLEMS IN PRIVACY & ML

- **Going beyond worst-case privacy-utility trade-offs:** leverage the structure of some machine learning problems to design better DP algorithms
- **Better privacy accounting:** tight, automatic and personalized
- **Correctness guarantees under malicious parties:** make computation verifiable while preserving privacy guarantees
- **Combining DP with secure multi-party computation:** identify tractable secure primitives under which one can achieve trusted curator utility for many problems
- **Concrete DP/FL deployments:** match DP bounds to protection against specific attacks, articulate with the law (GDPR), make FL transparent to end-users

IF YOU WANT MORE...

- Other DP mechanisms (Laplace, exponential), DP-ERM via output perturbation, lab sessions in Python... check my longer course:
  http://researchers.lille.inria.fr/abellet/teaching/private_machine_learning_course.html
- Advances in Federated Learning:
  - Survey paper [Kairouz et al., 2021]
  - Online seminar: https://sites.google.com/view/one-world-seminar-series-flow/

[Abadi et al., 2016] Abadi, M., Chu, A., Goodfellow, I. J., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016).
Deep learning with differential privacy.
In *CCS*.

[Abowd, 2018] Abowd, J. M. (2018).
The U.S. Census Bureau Adopts Differential Privacy.
In *KDD*.

[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*.

[Bassily et al., 2016] Bassily, R., Nissim, K., Smith, A., Steinke, T., Stemmer, U., and Ullman, J. (2016).
Algorithmic stability for adaptive data analysis.
In *STOC*.

[Bassily et al., 2014] Bassily, R., Smith, A. D., and Thakurta, A. (2014).
Private Empirical Risk Minimization: Efficient Algorithms and Tight Error Bounds.
In *FOCS*.

[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*.

[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*.

[Carlini et al., 2022] Carlini, N., Chien, S., Nasr, M., Song, S., Terzis, A., and Tramer, F. (2022).
In *S&P*.

[Carlini et al., 2021] Carlini, N., Tramèr, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., Roberts, A., Brown, T., Song, D., Erlingsson, Ú., Oprea, A., and Raffel, C. (2021).
Extracting training data from large language models.
In *USENIX Security*.

[Chan et al., 2012] Chan, T.-H. H., Shi, E., and Song, D. (2012).
Optimal Lower Bound for Differentially Private Multi-party Aggregation.
In *ESA*.

[Cyffers and Bellet, 2022] Cyffers, E. and Bellet, A. (2022).
Privacy Amplification by Decentralization.
In *AISTATS*.

[Cyffers et al., 2022] Cyffers, E., Even, M., Bellet, A., and Massoulié, L. (2022).
Muffliato: Peer-to-Peer Privacy Amplification for Decentralized Optimization and Averaging.
Technical report, arXiv:2206.05091.

[de Montjoye et al., 2013] de Montjoye, Y.-A., Hidalgo, C. A., Verleysen, M., and Blondel, V. D. (2013).
Unique in the crowd: The privacy bounds of human mobility.
Scientific Reports, 3.

[Differential Privacy Team, Apple, 2017] Differential Privacy Team, Apple (2017).
Learning with privacy at scale.

[Ding et al., 2017] Ding, B., Kulkarni, J., and Yekhanin, S. (2017).
Collecting telemetry data privately.
In *NIPS*.

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

[Dinur and Nissim, 2003] Dinur, I. and Nissim, K. (2003).
Revealing information while preserving privacy.
In *PODS*.

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

[Duchi et al., 2013] Duchi, J. C., Jordan, M. I., and Wainwright, M. J. (2013).
Local Privacy and Statistical Minimax Rates.
In *FOCS*.

[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)*.

[Dwork and Roth, 2014] Dwork, C. and Roth, A. (2014).
The Algorithmic Foundations of Differential Privacy.
Foundations and Trends in Theoretical Computer Science, 9(3–4):211–407.

[Erlingsson et al., 2014] Erlingsson, U., Pihur, V., and Korolova, A. (2014).
Rappor: Randomized aggregatable privacy-preserving ordinal response.
In *CCS*.

[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*.

[Geiping et al., 2020] Geiping, J., Bauermeister, H., Dröge, H., and Moeller, M. (2020).
Inverting gradients - how easy is it to break privacy in federated learning?
In *NeurIPS*.

[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*.

[Homer et al., 2008] Homer, N., Szelinger, S., Redman, M., Duggan, D., Tembe, W., Muehling, J., Pearson, J. V., Stephan, D. A., Nelson, S. F., and Craig, D. W. (2008).
Resolving individuals contributing trace amounts of dna to highly complex mixtures using high-density snp genotyping microarrays.
PLOS Genetics, 4(8):1–9.

[Jayaraman and Evans, 2019] Jayaraman, B. and Evans, D. (2019).
Evaluating Differentially Private Machine Learning in Practice.
In *USENIX Security*.

[Jung et al., 2021] Jung, C., Ligett, K., Neel, S., Roth, A., Sharifi-Malvajerdi, S., and Shenfeld, M. (2021).
A New Analysis of Differential Privacy's Generalization Guarantees (Invited Paper).

[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., 2014] Kairouz, P., Oh, S., and Viswanath, P. (2014).
Extremal mechanisms for local differential privacy.
In *NIPS*.

[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*.

[Kasiviswanathan et al., 2008] Kasiviswanathan, S. P., Lee, H. K., Nissim, K., Raskhodnikova, S., and Smith, A. D. (2008).
What Can We Learn Privately?
In *FOCS*.

[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*.

[Marfoq et al., 2021] Marfoq, O., Neglia, G., Bellet, A., Kameni, L., and Vidal, R. (2021).
Federated Multi-Task Learning under a Mixture of Distributions.
In *NeurIPS*.

[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*.

[Mironov, 2017] Mironov, I. (2017).
Renyi differential privacy.
In *CSF*.

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

[Narayanan and Shmatikov, 2008] Narayanan, A. and Shmatikov, V. (2008).
Robust de-anonymization of large sparse datasets.
In *IEEE Symposium on Security and Privacy (S&P)*.

[Narayanan and Shmatikov, 2009] Narayanan, A. and Shmatikov, V. (2009).
De-anonymizing social networks.
In *IEEE Symposium on Security and Privacy (S&P)*.

[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*.

[Nasr et al., 2021] Nasr, M., Song, S., Thakurta, A. G., Papernot, N., and Carlini, N. (2021).
Adversary Instantiation: Lower bounds for differentially private machine learning.
In *IEEE Symposium on Security and Privacy (S&P)*.

[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*.

[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.

[Shamir and Zhang, 2013] Shamir, O. and Zhang, T. (2013).
Stochastic Gradient Descent for Non-smooth Optimization: Convergence Results and Optimal Averaging Schemes.
In *ICML*.

[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*.

[Sweeney, 2002] Sweeney, L. (2002).
k-anonymity: A model for protecting privacy.
International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 10(5):557–570.

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

[Warner, 1965] Warner, S. L. (1965).
Randomised response: a survey technique for eliminating evasive answer bias.
Journal of the American Statistical Association, 60(309):63–69.

[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*.