Differential privacy and generalization

Germán Bassi

School of Electrical Engineering and Computer Science
KTH Royal Institute of Technology

November 18, 2019

Outline

Outline

What is privacy?

- It is the 1940s in the US; a large medical study reveals that smoking regularly has a strong connection to higher risks of developing lung cancer.
- Bob, who regularly smokes, participates in the study.
- He later sees that, as a result of this medical study, his health-insurance premiums rise.
- Clearly, the study revealed some unknown information about Bob. But, was this a breach of privacy?

Privacy in the context of DP

> *Differential privacy addresses the paradox of learning nothing about an individual while learning useful information about a population.*[1]

In the context of differential privacy (DP), we consider that there is no leakage of private information if the "impact" on the smoker is the same independent of whether or not he was in the study.

↓

Bob's participation should not affect the result of the study

[1] C. Dwork and A. Roth. "The algorithmic foundations of differential privacy." Foundations and Trends in Theoretical Computer Science 9.3–4 (2014): 211–407.

Privacy in the context of DP

Differential privacy addresses the paradox of learning nothing about an individual while learning useful information about a population.[1]

In the context of differential privacy (DP), we consider that there is no leakage of private information if the "impact" on the smoker is the same independent of whether or not he was in the study.

↓

Bob's participation should not affect the result of the study

[1] C. Dwork and A. Roth. "The algorithmic foundations of differential privacy." Foundations and Trends in Theoretical Computer Science 9.3–4 (2014): 211–407.

The basic model in DP

Differential privacy is a definition of privacy tailored to the problem of privacy-preserving data analysis:

- We assume that the information is stored in a database D.
- The database has n rows, each from a different individual.
- The database is maintained by a trusted and trustworthy curator.
- A data analyst, who does not have direct access to D, asks queries in order to perform some statistical analysis of the database as a whole (aggregated data).

If the data analyst does not have access to D, how can there be a breach of Bob's privacy?

The basic model in DP

Differential privacy is a definition of privacy tailored to the problem of privacy-preserving data analysis:

- We assume that the information is stored in a database D.
- The database has n rows, each from a different individual.
- The database is maintained by a trusted and trustworthy curator.
- A data analyst, who does not have direct access to D, asks queries in order to perform some statistical analysis of the database as a whole (aggregated data).

If the data analyst does not have access to D, how can there be a breach of Bob's privacy?

A differencing attack

Person	# of cig.
Alice	3
Bob	5
Charles	1
⋮	⋮
Zoe	4

Database 1

Person	# of cig.
Alice	3
Charles	1
Diana	0
⋮	⋮
Zoe	4

Database 2

- Assume you have two identical databases except for Bob's data, which it only appears in the first one.
- The innocent-looking queries "total number of participants" and "average cigarette consumption", when performed on both databases, allow for perfectly characterizing Bob's daily cigarette consumption.

A differencing attack

Person	# of cig.
Alice	3
Bob	5
Charles	1
⋮	⋮
Zoe	4

Database 1

Person	# of cig.
Alice	3
Charles	1
Diana	0
⋮	⋮
Zoe	4

Database 2

- Adding zero-mean independent random noise to the queries' answers helps us protect the participants but degrades the quality of the statistical analysis. There is a trade-off.
- If the analyst performs multiple identical queries, the effect of the noise can be averaged out.

Lessons learned

- The most detrimental thing (from the point of view of DP) is to have two almost identical databases, i.e., two databases that differ in just one entry.
  - We need to focus on databases with "distance 1".
- Some noise needs to be added to the queries' answers in order to assure some level of privacy.
  - There is a natural and unavoidable trade-off between statistical accuracy and privacy.
- Allowing multiple queries to the same database inherently decreases the level of privacy any method can achieve.
  - We cannot release unlimited number of results. Clever strategies need to be devise to release the most amount of useful information.

Lessons learned

- The most detrimental thing (from the point of view of DP) is to have two almost identical databases, i.e., two databases that differ in just one entry.
  - We need to focus on databases with "distance 1".
- Some noise needs to be added to the queries' answers in order to assure some level of privacy.
  - There is a natural and unavoidable trade-off between statistical accuracy and privacy.
- Allowing multiple queries to the same database inherently decreases the level of privacy any method can achieve.
  - We cannot release unlimited number of results. Clever strategies need to be devise to release the most amount of useful information.

Lessons learned

- The most detrimental thing (from the point of view of DP) is to have two almost identical databases, i.e., two databases that differ in just one entry.
  - We need to focus on databases with "distance 1".
- Some noise needs to be added to the queries' answers in order to assure some level of privacy.
  - There is a natural and unavoidable trade-off between statistical accuracy and privacy.
- Allowing multiple queries to the same database inherently decreases the level of privacy any method can achieve.
  - We cannot release unlimited number of results. Clever strategies need to be devise to release the most amount of useful information.

Outline

First definitions

- A database x is a collection of records from a universe $\mathcal{X}$.
  - In our example about the daily cigarette consumption, $\mathcal{X} = \{0, 1, 2, \ldots\}$
- The database is represented by a vector of occurrences of each type in $\mathcal{X}$, i.e., $x \in \mathbb{N}^{|\mathcal{X}|}$.
  - In our example, $x = [10, 2, 4, 13, 44, 35]$ means that the database contains 10 participants that declared 0 cigarettes per day, 2 participants that smoke only 1 cigarette per day, ...
  - It is similar to the type in information theory but without normalization.
- A (privacy) mechanism is an algorithm that takes as inputs a database, a query, and other parameters, and returns the corresponding answer to the query in the database (hopefully with some guarantees on the level of privacy).

First definitions (cont.)

- With our previous definition of database, a useful measure is the ℓ_1 norm. Given two databases x and y:

$$\underbrace{\|x\|_1 = \sum_{i=1}^{|\mathcal{X}|} |x_i|}_{\text{size of } x} \qquad \underbrace{\|x - y\|_1 = \sum_{i=1}^{|\mathcal{X}|} |x_i - y_i|}_{\text{number of records which are different in } x \text{ and } y}$$

- Going back to our example of the differencing attack, x represents the database which has Bob's data (he smokes 5 cigarettes a day) and y is the other almost identical database:

$$\begin{aligned} \text{x} &= [10,\ 2,\ 4,\ 13,\ 44,\ 35] \\ \text{y} &= [10,\ 2,\ 4,\ 13,\ 44,\ 34] \end{aligned} \quad \longrightarrow \quad \|x - y\|_1 = 1.$$

First definitions (cont.)

Randomized Algorithm

A randomized algorithm $\mathcal{M}$ with domain A and discrete range B is associated with a mapping $M : A \to \Delta(B)$, where $\Delta(B)$ represents the probability simplex on B.

On input $a \in A$, the algorithm $\mathcal{M}$ outputs $\mathcal{M}(a) = b$ with probability $(M(a))_b$ for each $b \in B$.

Example: In our example about the daily cigarette intake, let $\mathcal{M}(x)$ calculate the (noisy) empirical mean of the $\|x\|_1 = n$ integer values from the database.

First definitions (cont.)

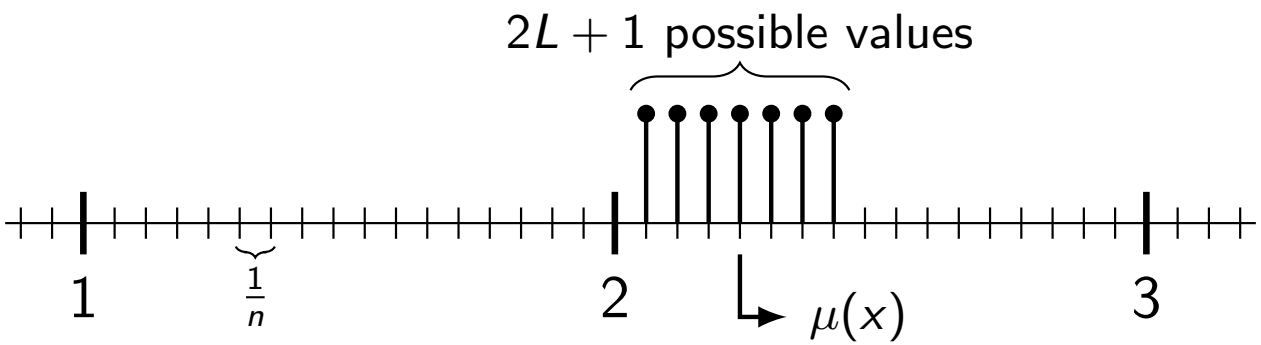
Remark: for simplicity ignore border effects. If $\mathcal{X} = \{0, 1, 2, 3\}$ the mean should not be larger than 3. This example ignores this fact.

We take $B = \left\{\frac{k}{n}\right\}_{k=0}^{\infty}$, i.e., a quantization of the space.

The true mean is $\mu(x) = \frac{1}{n}\sum_{i=1}^{|\mathcal{X}|}(i-1)\,x_i$.

However, $\mathcal{M}$ outputs a random value close to the true mean:

$$\Pr\{\mathcal{M}(x) = b\} = \begin{cases} \frac{1}{2L+1} & \text{if } b \in \left[\mu(x) - \frac{L}{n}, \mu(x) + \frac{L}{n}\right] \\ 0 & \text{else} \end{cases}$$



Definition and properties of DP

Differential Privacy

A randomized algorithm $\mathcal{M}$ with domain $\mathbb{N}^{|\mathcal{X}|}$ is (ε, δ)-differentially private if for all $\mathcal{S} \subseteq \text{Range}(\mathcal{M})$ and for all x, $y \in \mathbb{N}^{|\mathcal{X}|}$ such that $\|x - y\|_1 \leq 1$:

$$\Pr\{\mathcal{M}(x) \in \mathcal{S}\} \leq \exp(\varepsilon) \Pr\{\mathcal{M}(y) \in \mathcal{S}\} + \delta,$$

where the probability space is over the coin flips of the mechanism $\mathcal{M}$. If $\delta = 0$, we say that $\mathcal{M}$ is ε-differentially private (sometimes referred to as pure DP).

Remark: The smaller ε and δ are, the more private the algorithm is.

Definition and properties of DP (cont.)

DP: $\forall x, y \in \mathbb{N}^{|\mathcal{X}|}$ s.t. $\|x - y\|_1 \leq 1$,

$$\Pr\{\mathcal{M}(x) \in \mathcal{S}\} \leq \exp(\varepsilon) \Pr\{\mathcal{M}(y) \in \mathcal{S}\} + \delta$$

If $\delta = 0$ we see that

$$\varepsilon \geq \ln \frac{\Pr\{\mathcal{M}(x) \in \mathcal{S}\}}{\Pr\{\mathcal{M}(y) \in \mathcal{S}\}},$$

i.e., the log-likelihood ratio should be bounded.

For every run of the mechanism $\mathcal{M}(x)$, the output observed is (almost) equally likely to be observed on every neighboring database.

Moreover, $\Pr\{\mathcal{M}(x) \in \mathcal{S}\} = 0 \Leftrightarrow \Pr\{\mathcal{M}(y) \in \mathcal{S}\} = 0$.

Definition and properties of DP (cont.)

DP: $\forall x, y \in \mathbb{N}^{|\mathcal{X}|}$ s.t. $\|x - y\|_1 \leq 1$,

$$\Pr\{\mathcal{M}(x) \in \mathcal{S}\} \leq \exp(\varepsilon) \Pr\{\mathcal{M}(y) \in \mathcal{S}\} + \delta$$

If $\delta \neq 0$ we see that, $\forall \mathcal{S} \subseteq \text{Range}(\mathcal{M})$ s.t. $\Pr\{\mathcal{M}(x) \in \mathcal{S}\} \geq \delta$,

$$\varepsilon \geq \ln \frac{\Pr\{\mathcal{M}(x) \in \mathcal{S}\} - \delta}{\Pr\{\mathcal{M}(y) \in \mathcal{S}\}}.$$

For very small δ, it is extremely unlikely that the observed value $\mathcal{M}(x)$ will be much more or much less likely to be generated when the database is x than when the database is y.

However, it is possible that $0 < \Pr\{\mathcal{M}(x) \in \mathcal{S}\} < \delta$ and $\Pr\{\mathcal{M}(y) \in \mathcal{S}\} = 0$.

Definition and properties of DP (cont.)

Post-processing

Let $\mathcal{M} : \mathbb{N}^{|\mathcal{X}|} \to R$ be a randomized algorithm that is (ε, δ)-DP. Let $f : R \to R'$ be an arbitrary randomized mapping. Then $f \circ M : \mathbb{N}^{|\mathcal{X}|} \to R'$ is (ε, δ)-DP.

Proof: Let x and y s.t. $\|x - y\|_1 \leq 1$.

Assume f is deterministic, fix $S \subseteq R'$ and let $T = \{r \in R : f(r) \in S\}$. Then, we have that:

$$\begin{aligned}\Pr\{f(\mathcal{M}(x)) \in S\} &= \Pr\{\mathcal{M}(x) \in T\} \\ &\leq \exp(\varepsilon)\Pr\{\mathcal{M}(y) \in T\} + \delta \\ &= \exp(\varepsilon)\Pr\{f(\mathcal{M}(y)) \in S\} + \delta.\end{aligned}$$

Stochastic f follows as a convex combination of deterministic functions.

Composition of DP mechanisms

What happens when we combine the outputs of two DP mechanisms (on the same database)?

As we saw in the first part, the strength of the privacy guarantee degrades by performing several queries.

We show first the following simple result:

Composition of two *pure* DP mechanisms

Let $\mathcal{M}_1 : \mathbb{N}^{|\mathcal{X}|} \to R_1$ be an ε_1-DP algorithm and let $\mathcal{M}_2 : \mathbb{N}^{|\mathcal{X}|} \to R_2$ be an ε_2-DP algorithm. Then their combination, defined to be $\mathcal{M}_{1,2} : \mathbb{N}^{|\mathcal{X}|} \to R_1 \times R_2$ by the mapping: $\mathcal{M}_{1,2}(x) = \big(\mathcal{M}_1(x), \mathcal{M}_2(x)\big)$ is $(\varepsilon_1 + \varepsilon_2)$-DP.

Composition of DP mechanisms (cont.)

Proof: Let x, $y \in \mathbb{N}^{|\mathcal{X}|}$ be such that $\|x - y\|_1 \leq 1$. Fix any $(r_1, r_2) \in R_1 \times R_2$. Then,

$$\begin{aligned}\frac{\Pr\{\mathcal{M}_{1,2}(x) = (r_1, r_2)\}}{\Pr\{\mathcal{M}_{1,2}(y) = (r_1, r_2)\}} &= \frac{\Pr\{\mathcal{M}_1(x) = r_1\}}{\Pr\{\mathcal{M}_1(y) = r_1\}} \frac{\Pr\{\mathcal{M}_2(x) = r_2\}}{\Pr\{\mathcal{M}_2(y) = r_2\}} \\ &\leq \exp(\varepsilon_1) \exp(\varepsilon_2) \\ &= \exp(\varepsilon_1 + \varepsilon_2).\end{aligned}$$

By symmetry,

$$\begin{aligned}\frac{\Pr\{\mathcal{M}_{1,2}(x) = (r_1, r_2)\}}{\Pr\{\mathcal{M}_{1,2}(y) = (r_1, r_2)\}} &\geq \exp(-\varepsilon_1) \exp(-\varepsilon_2) \\ &= \exp\big(-(\varepsilon_1 + \varepsilon_2)\big),\end{aligned}$$

which proves the statement. ■

Composition of DP mechanisms (cont.)

What if the algorithms are not *pure* DP?

Composition of several DP mechanisms

Let $\mathcal{M}_i : \mathbb{N}^{|\mathcal{X}|} \to R_i$ be an $(\varepsilon_i, \delta_i)$-DP algorithm for $i \in [k]$. If $\mathcal{M}_{[k]} : \mathbb{N}^{|\mathcal{X}|} \to \prod_{i=1}^{k} R_i$ is defined to be $\mathcal{M}_{[k]}(x) = (\mathcal{M}_1(x), \ldots, \mathcal{M}_k(x))$, then $\mathcal{M}_{[k]}$ is $\left(\sum_{i=1}^{k} \varepsilon_i, \sum_{i=1}^{k} \delta_i\right)$-DP.

What if the algorithms are chosen adaptively?

Adaptive composition of several DP mechanisms

Let $\mathcal{M}_1 : \mathbb{N}^{|\mathcal{X}|} \to R_1$ be an $(\varepsilon_1, \delta_1)$-DP algorithm and, for $2 \leq i \leq k$, let $\mathcal{M}_i : \mathbb{N}^{|\mathcal{X}|} \times \prod_{j=1}^{i-1} R_j \to R_i$ be an $(\varepsilon_i, \delta_i)$-DP algorithm for every input $(r_1, \ldots, r_{i-1}) \in \prod_{j=1}^{i-1} R_j$.

Then their adaptive composition, defined to be $\mathcal{M} : \mathbb{N}^{|\mathcal{X}|} \to R_k$, is $\left(\sum_{i=1}^{k} \varepsilon_i, \sum_{i=1}^{k} \delta_i\right)$-DP.

Composition of DP mechanisms (cont.)

What if the algorithms are not *pure* DP?

Composition of several DP mechanisms

Let $\mathcal{M}_i : \mathbb{N}^{|\mathcal{X}|} \to R_i$ be an $(\varepsilon_i, \delta_i)$-DP algorithm for $i \in [k]$. If $\mathcal{M}_{[k]} : \mathbb{N}^{|\mathcal{X}|} \to \prod_{i=1}^{k} R_i$ is defined to be $\mathcal{M}_{[k]}(x) = (\mathcal{M}_1(x), \ldots, \mathcal{M}_k(x))$, then $\mathcal{M}_{[k]}$ is $\left(\sum_{i=1}^{k} \varepsilon_i, \sum_{i=1}^{k} \delta_i\right)$-DP.

What if the algorithms are chosen adaptively?

Adaptive composition of several DP mechanisms

Let $\mathcal{M}_1 : \mathbb{N}^{|\mathcal{X}|} \to R_1$ be an $(\varepsilon_1, \delta_1)$-DP algorithm and, for $2 \leq i \leq k$, let $\mathcal{M}_i : \mathbb{N}^{|\mathcal{X}|} \times \prod_{j=1}^{i-1} R_j \to R_i$ be an $(\varepsilon_i, \delta_i)$-DP algorithm for every input $(r_1, \ldots, r_{i-1}) \in \prod_{j=1}^{i-1} R_j$.

Then their adaptive composition, defined to be $\mathcal{M} : \mathbb{N}^{|\mathcal{X}|} \to R_k$, is $\left(\sum_{i=1}^{k} \varepsilon_i, \sum_{i=1}^{k} \delta_i\right)$-DP.

Advanced composition

A more sophisticated argument yields significant improvement on the scaling of ε when we allow for a degradation in terms of δ.

Advanced adaptive composition

For all $\varepsilon, \delta, \delta' \geq 0$, the adaptive composition of k arbitrary (ε, δ)-DP algorithms is $(\varepsilon', k\delta + \delta')$-DP, where

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

Assuming $\varepsilon \ll 1$, $e^{\varepsilon} \approx 1 + \varepsilon$, thus

$$\varepsilon' \approx \sqrt{2k \ln(1/\delta')}\,\varepsilon + k\varepsilon^2 \approx \sqrt{2k \ln(1/\delta')}\,\varepsilon.$$

By allowing a small δ', this result states that $\varepsilon' = \mathcal{O}(\sqrt{k}\varepsilon)$ instead of $\varepsilon' = \mathcal{O}(k\varepsilon)$ as in the simple argument.

Advanced composition (cont.)

Parameter optimization for composition of DP mechanisms[2]

Given target privacy parameters $\varepsilon', \delta' > 0$, to ensure $(\varepsilon', k\delta + \delta')$ cumulative privacy loss over k mechanisms, it suffices that each mechanism is (ε, δ)-DP, where

$$\varepsilon \leq \frac{\varepsilon'}{\sqrt{2k \ln(1/\delta')}}.$$

Example: Bob is a member of $k = 10{,}000$ ε-DP (uncoordinated) databases. What should be the value of ε so that Bob's cumulative privacy loss is bounded by $\varepsilon' = 1$ with probability at least $1 - e^{-32}$?

Taking $\delta' = e^{-32} \approx 10^{-14}$ it suffices to have $\varepsilon \leq 1/800$.

Compare this to $\varepsilon \leq 1/10{,}000$ from the simple composition bound, which assumes $\delta' = 0$.

[2] As we saw, this is an approximation and should be handled with care. For simplicity, we will take the inequality as true for the rest of the presentation.

Advanced composition (cont.)

The results on the composition of k (ε, δ)-DP mechanisms has two interpretations:

- One user that participates in k (ε, δ)-DP databases.
- One database that answers k (ε, δ)-DP queries.

In the following part, we will see that ε is related to the amount of noise that we need to add before answering the queries, and thus it is related to the expected distortion of the answers.

It is therefore convenient (if possible) to use the advanced composition result such that the final privacy level is $\varepsilon' = \mathcal{O}(\sqrt{k}\varepsilon)$ instead of $\varepsilon' = \mathcal{O}(k\varepsilon)$.

Take-away messages

- A randomized algorithm $\mathcal{M}$ is (ε, δ)-DP if for all databases x and $y \in \mathbb{N}^{|\mathcal{X}|}$ such that $\|x - y\|_1 \leq 1$:

$$\Pr\{\mathcal{M}(x) \in \mathcal{S}\} \leq \exp(\varepsilon) \Pr\{\mathcal{M}(y) \in \mathcal{S}\} + \delta.$$

- Post-processing: differential privacy is preserved under any arbitrary randomized mapping.
- The composition of k (ε, δ)-DP mechanisms is:
  - a $(k\varepsilon, k\delta)$-DP mechanism; or,
  - a $(\varepsilon', k\delta + \delta')$-DP mechanism, where $\varepsilon' = \mathcal{O}(\sqrt{k}\varepsilon)$ and $\delta' > 0$.

Outline

Preliminaries

The most fundamental type of queries are numeric queries, i.e., functions $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$.

ℓ_1-sensitivity

The ℓ_1-sensitivity of a function $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$ is:

$$\Delta f = \max_{x,y \in \mathbb{N}^{|\mathcal{X}|} \text{ s.t. } \|x-y\|_1 = 1} \|f(x) - f(y)\|_1$$

The ℓ_1-sensitivity of a function f captures the magnitude by which a single individual's data can change the function f the most.

This is related to the uncertainty in the response that we must introduce in order to hide the participation of a single individual.

Preliminaries (cont.)

Examples:

- Counting query: "How many users smoke 5 cigarettes a day?"

  Removing one user can only affect the count by one, i.e., $\Delta f = 1$.

- m (general) counting queries: 1) "How many users smoke 5 cigarettes a day? and 2) "How many users are older than 30?"

  In the worst case, a user affects all counts, i.e., $\Delta f = m$.

- m disjoint counting queries: "How many users smoke i cigarettes a day?" (repeat for $i = 0$ to 10^3)

  A user can only affect the count of only one question by one, i.e., $\Delta f = 1$.

Laplace mechanism

Laplace distribution

The Laplace distribution (centered at 0) with scale b is the distribution with probability density function:

$$\text{Lap}(x|b) = \frac{1}{2b} \exp\left(-\frac{|x|}{b}\right).$$

The variance of this distribution is $2b^2$.

For simplicity, we abuse notation and write $\text{Lap}(b)$ when clear.

Remark: The Laplace distribution is a symmetric version of the exponential distribution.

Laplace mechanism (cont.)

Laplace mechanism

Given any function $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$, the Laplace mechanism is defined as:

$$\mathcal{M}_L(x, f, \varepsilon) = f(x) + (Y_1, \ldots, Y_k),$$

where Y_i are i.i.d. random variables drawn from $\mathrm{Lap}(\sigma)$, and $\sigma = \Delta f / \varepsilon$ is sometimes called the noise rate.

Remark: The larger the ℓ_1-sensitivity of f ($\Delta f \to \infty$) or the more stringent the privacy requirement ($\varepsilon \to 0$), the larger the variance of the added noise.

Laplace mechanism (cont.)

Theorem

The Laplace mechanism preserves $(\varepsilon, 0)$-differential privacy.

Proof: Let $x, y \in \mathbb{N}^{|\mathcal{X}|}$ be such that $\|x - y\|_1 \leq 1$, and let $f(\cdot)$ be some function $f : \mathbb{N}^{|\mathcal{X}|} \to \mathbb{R}^k$.

Let p_x and p_y denote the probability density functions of $\mathcal{M}_L(x, f, \varepsilon)$ and $\mathcal{M}_L(y, f, \varepsilon)$, respectively. We compare the two at some arbitrary point $z \in \mathbb{R}^k$

$$\frac{p_x(z)}{p_y(z)} = \prod_{i=1}^{k} \frac{\exp\left(-\frac{\varepsilon|f(x)_i - z_i|}{\Delta f}\right)}{\exp\left(-\frac{\varepsilon|f(y)_i - z_i|}{\Delta f}\right)}$$

$$= \prod_{i=1}^{k} \exp\left(\frac{\varepsilon(|f(y)_i - z_i| - |f(x)_i - z_i|)}{\Delta f}\right)$$

Laplace mechanism (cont.)

$$\begin{aligned}\frac{p_x(z)}{p_y(z)} = \cdots &\leq \prod_{i=1}^{k} \exp\left(\frac{\varepsilon(|f(x)_i - f(y)_i|)}{\Delta f}\right) \\ &= \exp\left(\frac{\varepsilon \|f(x) - f(y)\|_1}{\Delta f}\right) \\ &\leq \exp(\varepsilon).\end{aligned}$$

Following similar steps, we can prove that $\frac{p_x(z)}{p_y(z)} \geq \exp(-\varepsilon)$. ■

Accuracy of queries under the Laplace mechanism

Simple counting query: As we saw, for counting queries we have that $\Delta f = 1$, and thus ε-DP can be achieved for counting queries by the addition of noise drawn from $\text{Lap}(1/\varepsilon)$.
The expected distortion, or error, is

$$\mathbb{E}[|f(x) - \mathcal{M}_L(x, f, \varepsilon)|] = \mathbb{E}[|Y|] = \frac{1}{\varepsilon},$$

which is independent of the size of the database.

m (general) counting queries: In this case, $\Delta f = m$, and thus we need to add noise drawn from $\text{Lap}(m/\varepsilon)$ to the answer of each query in order to achieve ε-DP.
The expected distortion for each query is now

$$\mathbb{E}[|f(x)_i - \mathcal{M}_L(x, f, \tfrac{\varepsilon}{m})_i|] = \mathbb{E}[|Y_i|] = \frac{m}{\varepsilon},$$

Accuracy of queries under the Laplace mechanism

Simple counting query: As we saw, for counting queries we have that $\Delta f = 1$, and thus ε-DP can be achieved for counting queries by the addition of noise drawn from $\text{Lap}(1/\varepsilon)$.
The expected distortion, or error, is

$$\mathbb{E}[|f(x) - \mathcal{M}_L(x, f, \varepsilon)|] = \mathbb{E}[|Y|] = \frac{1}{\varepsilon},$$

which is independent of the size of the database.

m (general) counting queries: In this case, $\Delta f = m$, and thus we need to add noise drawn from $\text{Lap}(m/\varepsilon)$ to the answer of each query in order to achieve ε-DP.
The expected distortion for each query is now

$$\mathbb{E}[|f(x)_i - \mathcal{M}_L(x, f, \tfrac{\varepsilon}{m})_i|] = \mathbb{E}[|Y_i|] = \frac{m}{\varepsilon},$$

Accuracy of queries under the Laplace mechanism (cont.)

m (general) counting queries (advanced): Remember that by accepting a small $\delta > 0$, we can ensure (ε, δ) cumulative privacy loss over *m* mechanisms if each mechanism is ε_0-DP, where

$$\varepsilon_0 \approx \frac{\varepsilon}{\sqrt{2m \ln(1/\delta)}}.$$

Therefore, the expected distortion for each query is now

$$\mathbb{E}[|f(x)_i - \mathcal{M}_L(x, f, \varepsilon_0)_i|] = \mathbb{E}[|Y_i|] \approx \frac{\sqrt{2m \ln(1/\delta)}}{\varepsilon},$$

which has a lower scaling factor with respect to *m*.

Accuracy of queries under the Laplace mechanism (cont.)

If our database has n entries, how many counting queries m can we ask without losing too much accuracy?

Usually, counting queries are used to estimate the percentage of certain statement/property on a population:

$$\text{Percentage} = \frac{\text{Count}}{n}.$$

Therefore, given that the expected distortion is $\mathcal{O}(\sqrt{m})$, we may ask $o(n^2)$ queries with non-trivial accuracy.

Asking $o(n)$ queries reduces the error introduced by the DP mechanism below the level of sampling error, i.e., $o(1/\sqrt{n})$.

If the Laplacian noises are not i.i.d. this can be further improved. To achieve this, we need to coordinate the answers.

Accuracy of queries under the Laplace mechanism (cont.)

If our database has n entries, how many counting queries m can we ask without losing too much accuracy?

Usually, counting queries are used to estimate the percentage of certain statement/property on a population:

$$\text{Percentage} = \frac{\text{Count}}{n}.$$

Therefore, given that the expected distortion is $\mathcal{O}(\sqrt{m})$, we may ask $o(n^2)$ queries with non-trivial accuracy.

Asking $o(n)$ queries reduces the error introduced by the DP mechanism below the level of sampling error, i.e., $o(1/\sqrt{n})$.

If the Laplacian noises are not i.i.d. this can be further improved. To achieve this, we need to coordinate the answers.

Accuracy of queries under the Laplace mechanism (cont.)

If our database has n entries, how many counting queries m can we ask without losing too much accuracy?

Usually, counting queries are used to estimate the percentage of certain statement/property on a population:

$$\text{Percentage} = \frac{\text{Count}}{n}.$$

Therefore, given that the expected distortion is $\mathcal{O}(\sqrt{m})$, we may ask $o(n^2)$ queries with non-trivial accuracy.

Asking $o(n)$ queries reduces the error introduced by the DP mechanism below the level of sampling error, i.e., $o(1/\sqrt{n})$.

If the Laplacian noises are not i.i.d. this can be further improved. To achieve this, we need to coordinate the answers.

Accuracy of queries under the Laplace mechanism (cont.)

If our database has n entries, how many counting queries m can we ask without losing too much accuracy?

Usually, counting queries are used to estimate the percentage of certain statement/property on a population:

$$\text{Percentage} = \frac{\text{Count}}{n}.$$

Therefore, given that the expected distortion is $\mathcal{O}(\sqrt{m})$, we may ask $o(n^2)$ queries with non-trivial accuracy.

Asking $o(n)$ queries reduces the error introduced by the DP mechanism below the level of sampling error, i.e., $o(1/\sqrt{n})$.

If the Laplacian noises are not i.i.d. this can be further improved. To achieve this, we need to coordinate the answers.

Accuracy of queries under the Laplace mechanism (cont.)

If our database has n entries, how many counting queries m can we ask without losing too much accuracy?

Usually, counting queries are used to estimate the percentage of certain statement/property on a population:

$$\text{Percentage} = \frac{\text{Count}}{n}.$$

Therefore, given that the expected distortion is $\mathcal{O}(\sqrt{m})$, we may ask $o(n^2)$ queries with non-trivial accuracy.

Asking $o(n)$ queries reduces the error introduced by the DP mechanism below the level of sampling error, i.e., $o(1/\sqrt{n})$.

If the Laplacian noises are not i.i.d. this can be further improved. To achieve this, we need to coordinate the answers.

Report Noisy Max

Imagine we want to determine which of m counting queries has the highest value when the queries are not disjoint.

Releasing all the counts and letting the data analyst find the maximum is suboptimal. Remember that the vector of counts has high ℓ_1-sensitivity, specifically, $\Delta f = m$.

A better approach is to (internally) add independently generated Laplace noise $\mathrm{Lap}(1/\varepsilon)$ to each count and return the index of the largest noisy count.

The Report Noisy Max algorithm is $(\varepsilon, 0)$-differentially private.

Report Noisy Max

Imagine we want to determine which of m counting queries has the highest value when the queries are not disjoint.

Releasing all the counts and letting the data analyst find the maximum is suboptimal. Remember that the vector of counts has high ℓ_1-sensitivity, specifically, $\Delta f = m$.

A better approach is to (internally) add independently generated Laplace noise $\mathrm{Lap}(1/\varepsilon)$ to each count and return the index of the largest noisy count.

The Report Noisy Max algorithm is $(\varepsilon, 0)$-differentially private.

Report Noisy Max

Imagine we want to determine which of m counting queries has the highest value when the queries are not disjoint.

Releasing all the counts and letting the data analyst find the maximum is suboptimal. Remember that the vector of counts has high ℓ_1-sensitivity, specifically, $\Delta f = m$.

A better approach is to (internally) add independently generated Laplace noise $\mathrm{Lap}(1/\varepsilon)$ to each count and return the index of the largest noisy count.

The Report Noisy Max algorithm is $(\varepsilon, 0)$-differentially private.

Report Noisy Max

Imagine we want to determine which of m counting queries has the highest value when the queries are not disjoint.

Releasing all the counts and letting the data analyst find the maximum is suboptimal. Remember that the vector of counts has high ℓ_1-sensitivity, specifically, $\Delta f = m$.

A better approach is to (internally) add independently generated Laplace noise $\mathrm{Lap}(1/\varepsilon)$ to each count and return the index of the largest noisy count.

The Report Noisy Max algorithm is $(\varepsilon, 0)$-differentially private.

Sparse vector mechanism

- The Laplace mechanism can be used to answer adaptively chosen low-sensitivity queries.
  - The privacy loss increases proportionally with the number of queries answered (or its square root)
  - What do we do if we want to answer a very large number of queries with reasonable accuracy?

- In some situations, we may only care on identifying the queries that lie above a certain threshold.
  - If we only report when the answer to a query exceeds the threshold, we can show that privacy degrades only with the number of answered queries, rather than with the total.
  - This can be a huge savings if we know that the set of queries that lie above the threshold is much smaller than the total number of queries – that is, if the answer vector is sparse.

Sparse vector mechanism

- The Laplace mechanism can be used to answer adaptively chosen low-sensitivity queries.
  - The privacy loss increases proportionally with the number of queries answered (or its square root)
  - What do we do if we want to answer a very large number of queries with reasonable accuracy?
- In some situations, we may only care on identifying the queries that lie above a certain threshold.
  - If we only report when the answer to a query exceeds the threshold, we can show that privacy degrades only with the number of answered queries, rather than with the total.
  - This can be a huge savings if we know that the set of queries that lie above the threshold is much smaller than the total number of queries – that is, if the answer vector is sparse.

Sparse vector mechanism (cont.)

Basic setup:

- There are m total number of sensitivity-1 queries, which may be chosen adaptively.
- There is a single threshold T fixed in advance but it is possible for each query to have its own threshold.
- We add noise to query values and compare the results to T.
- We expect a small number c of noisy values to exceed the threshold, and we only release the noisy values above the threshold.
- The algorithm stops after releasing c values.

Sparse vector mechanism (cont.)

Basic setup:

- There are m total number of sensitivity-1 queries, which may be chosen adaptively.
- There is a single threshold T fixed in advance but it is possible for each query to have its own threshold.
- We add noise to query values and compare the results to T.
- We expect a small number c of noisy values to exceed the threshold, and we only release the noisy values above the threshold.
- The algorithm stops after releasing c values.

Sparse vector mechanism (cont.)

Basic setup:

- There are m total number of sensitivity-1 queries, which may be chosen adaptively.
- There is a single threshold T fixed in advance but it is possible for each query to have its own threshold.
- We add noise to query values and compare the results to T.
- We expect a small number c of noisy values to exceed the threshold, and we only release the noisy values above the threshold.
- The algorithm stops after releasing c values.

Sparse vector mechanism (cont.)

Basic setup:

- There are m total number of sensitivity-1 queries, which may be chosen adaptively.
- There is a single threshold T fixed in advance but it is possible for each query to have its own threshold.
- We add noise to query values and compare the results to T.
- We expect a small number c of noisy values to exceed the threshold, and we only release the noisy values above the threshold.
- The algorithm stops after releasing c values.

Sparse vector mechanism (cont.)

Basic setup:

- There are m total number of sensitivity-1 queries, which may be chosen adaptively.
- There is a single threshold T fixed in advance but it is possible for each query to have its own threshold.
- We add noise to query values and compare the results to T.
- We expect a small number c of noisy values to exceed the threshold, and we only release the noisy values above the threshold.
- The algorithm stops after releasing c values.

Detour: AboveThreshold algorithm

The following is a special case with $c = 1$ released values.

procedure AboveThreshold($D, \{f_i\}, T, \varepsilon$)
  $\hat{T} \leftarrow T + \mathrm{Lap}\left(\frac{2}{\varepsilon}\right)$ ▷ Add noise to the threshold
  for all f_i **do**
    $\nu_i \leftarrow \mathrm{Lap}\left(\frac{4}{\varepsilon}\right)$ ▷ Noise for the query value
    if $f_i(D) + \nu_i \geq \hat{T}$ **then**
      output $a_i = \top$
      halt ▷ We stop after a positive match
    else
      output $a_i = \bot$
    end if
  end for
end procedure

Detour: AboveThreshold algorithm (cont.)

The AboveThreshold algorithm is $(\varepsilon, 0)$-differentially private.

Sketch of Proof: It is a sequential proof, i.e., we assume that we are on round i and the previous noises $\nu_1, \ldots, \nu_{i-1}$ are fixed.

We then calculate the probability of halting at round i (by integrating over the p.d.f. of $\hat{T}$ and ν_i) for two almost-equal databases.

The proof looks like the composition of two DP mechanisms: one hiding the effect of previous rounds (a 1-sensitivity query) and one hiding the effect of the present round (a 2-sensitivity query).

The parameters of the Laplacian noises of $\hat{T}$ and ν_i are selected such that the composition produces a ε-DP mechanism. ■

Sparse vector mechanism (again)

The Sparse algorithm is constructed as follows:

- Each received query is forwarded to the AboveThreshold algorithm.
- Each time a query produces a result exceeding the noisy threshold (and AboveThreshold halts), Sparse restarts AboveThreshold and continues feeding the remaining queries.
- The algorithm halts after AboveThreshold has been restarted c times.
- Since each instance of AboveThreshold is ε-DP, the composition results apply:
  - we may obtain a $(c\varepsilon, 0)$-DP mechanism; or,
  - we may obtain a (ε', δ')-DP mechanism with the advanced composition result.

Sparse vector mechanism (again)

The Sparse algorithm is constructed as follows:

- Each received query is forwarded to the AboveThreshold algorithm.
- Each time a query produces a result exceeding the noisy threshold (and AboveThreshold halts), Sparse restarts AboveThreshold and continues feeding the remaining queries.
- The algorithm halts after AboveThreshold has been restarted c times.
- Since each instance of AboveThreshold is ε-DP, the composition results apply:
  - we may obtain a $(c\varepsilon, 0)$-DP mechanism; or,
  - we may obtain a (ε', δ')-DP mechanism with the advanced composition result.

Sparse vector mechanism (again)

The Sparse algorithm is constructed as follows:

- Each received query is forwarded to the AboveThreshold algorithm.
- Each time a query produces a result exceeding the noisy threshold (and AboveThreshold halts), Sparse restarts AboveThreshold and continues feeding the remaining queries.
- The algorithm halts after AboveThreshold has been restarted c times.
- Since each instance of AboveThreshold is ε-DP, the composition results apply:
  - we may obtain a $(c\varepsilon, 0)$-DP mechanism; or,
  - we may obtain a (ε', δ')-DP mechanism with the advanced composition result.

Sparse vector mechanism (again)

The Sparse algorithm is constructed as follows:

- Each received query is forwarded to the AboveThreshold algorithm.
- Each time a query produces a result exceeding the noisy threshold (and AboveThreshold halts), Sparse restarts AboveThreshold and continues feeding the remaining queries.
- The algorithm halts after AboveThreshold has been restarted c times.
- Since each instance of AboveThreshold is ε-DP, the composition results apply:
  - we may obtain a $(c\varepsilon, 0)$-DP mechanism; or,
  - we may obtain a (ε', δ')-DP mechanism with the advanced composition result.

Sparse vector mechanism (cont.)

procedure Sparse($D, \{f_i\}, T, c, \varepsilon, \delta$)
  if $\delta = 0$ **then** $\sigma \leftarrow 2\frac{c}{\varepsilon}$ **else** $\sigma \leftarrow 2\frac{\sqrt{2c \ln(1/\delta)}}{\varepsilon}$ ▷ Select noise level
  $\hat{T}_0 \leftarrow T + \mathrm{Lap}(\sigma)$
  $count \leftarrow 0$ ▷ Initialize the counter
  for all f_i **do**
    $\nu_i \leftarrow \mathrm{Lap}(2\sigma)$
    if $f_i(D) + \nu_i \geq \hat{T}_{count}$ **then**
      output $a_i = \top$
      $count \leftarrow count + 1$ ▷ Increase the counter
      $\hat{T}_{count} \leftarrow T + \mathrm{Lap}(\sigma)$ ▷ Restart AboveThreshold
    else
      output $a_i = \bot$
    end if
    if $count \geq c$ **then**
      halt ▷ Halt if we have produced c positive results
    end if
  end for
end procedure

Take-away messages

- Given a function f with ℓ_1-sensitivity equal to Δf, we can construct a $(\varepsilon, 0)$-DP mechanism to answer it by adding $\mathrm{Lap}(\Delta f/\varepsilon)$ noise to the result.
- The composition of k queries, each one being $(\varepsilon, 0)$-DP, is:
  - $(k\varepsilon, 0)$-DP; or,
  - (ε', δ')-DP with $\varepsilon' \approx \sqrt{2k \ln(1/\delta')}\,\varepsilon$ for $\delta' > 0$.
- We can answer more queries if we reveal less information about them, e.g., by only saying that the result is above a certain threshold.

Outline

The problem of generalization in data analysis

- It is commonly assumed that each analysis procedure operates on a freshly sampled dataset, or at least, it is validated on a freshly sampled holdout (or testing) set.
- However, in practice the holdout dataset is rarely used only once.
- If the set of all tested hypotheses is known and independent of the holdout set, then it is easy to account for such multiple testing.
- However, such approach does not apply if the estimates or hypotheses tested on the holdout are chosen adaptively: that is, if the choice of hypotheses depends on previous analyses performed on the dataset.

Model description

- Adaptive data analysis is seen as a sequence of procedures: $\mathcal{A}_1 \to \mathcal{A}_2 \to \ldots \to \mathcal{A}_m$.
- These steps analyze a fixed dataset $S = (x_1, \ldots, x_n)$ drawn from a distribution $\mathcal{D}$ over $\mathcal{X}^n$.
- Each step is an algorithm $\mathcal{A}_i : \mathcal{X}^n \times \prod_{j=1}^{i-1} \mathcal{Y}_j \to \mathcal{Y}_i$ that takes the fixed dataset S and the previous results as inputs.
- It is assumed that each individual algorithm generalizes well when executed on a fresh dataset sampled from $\mathcal{D}$ for every input vector $(y_1, \ldots, y_{i-1})$.

Model description (cont.)

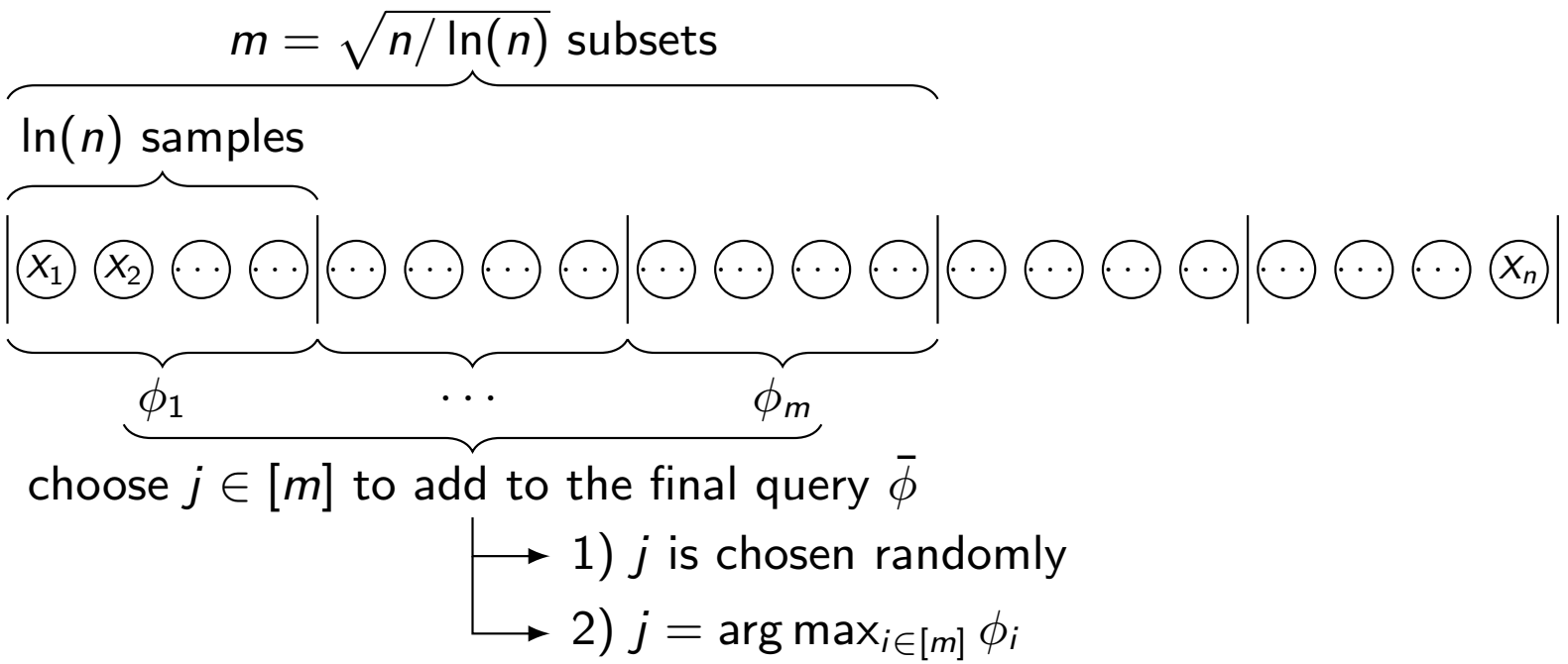
- For a function $\phi : \mathcal{X} \to \mathbb{R}$:
  - If we fix a dataset S, we denote the average value of ϕ on S as: $\mathcal{E}_S[\phi] = \frac{1}{n}\sum_{i=1}^{n} \phi(x_i)$.
  - If we take a distribution $\mathcal{P}$ over $\mathcal{X}$, we denote the expected value of ϕ as: $\mathcal{P}[\phi] = \mathbb{E}_{x\sim\mathcal{P}}[\phi(x)]$.
- In the (common) situation that the samples are taken i.i.d. according to a distribution $\mathcal{P}$ over $\mathcal{X}$, we say that the random dataset $\boldsymbol{S}$ is drawn from a distribution $\mathcal{P}^n$ over $\mathcal{X}^n$.
- When we say that an "algorithm generalizes well when executed on a fresh dataset," we mean that $\mathcal{E}_S[\phi]$ and $\mathcal{P}[\phi]$ should be close, for ϕ and S chosen independently.
- The random dataset $\boldsymbol{S}$ and the randomly selected function ϕ might not be independent in an adaptive setting, and thus $\mathcal{E}_{\boldsymbol{S}}[\phi]$ and $\mathcal{P}[\phi]$ might not be close.

Unrealistic example

- Assume that the dataset S contains n standard Gaussian RVs, i.e, $X \sim \mathcal{N}(0, 1)$.
- Further assume that the dataset is partitioned in $n/\ln(n)$ subsets, each one containing $\ln(n)$ samples. For simplicity, all these values are assumed to be integers.
- The query ϕ_j is the sample mean of the jth subset, whose answer is a zero-mean Gaussian RV with variance $1/\ln(n)$.
- If we were to randomly select $\sqrt{n/\ln(n)}$ subsets and average the queries' responses (final query $\bar{\phi}$), we would have an unbiased estimator of the true mean with a smaller variance ($1/\sqrt{n\ln(n)}$).
- But what if we do not choose these subsets randomly?

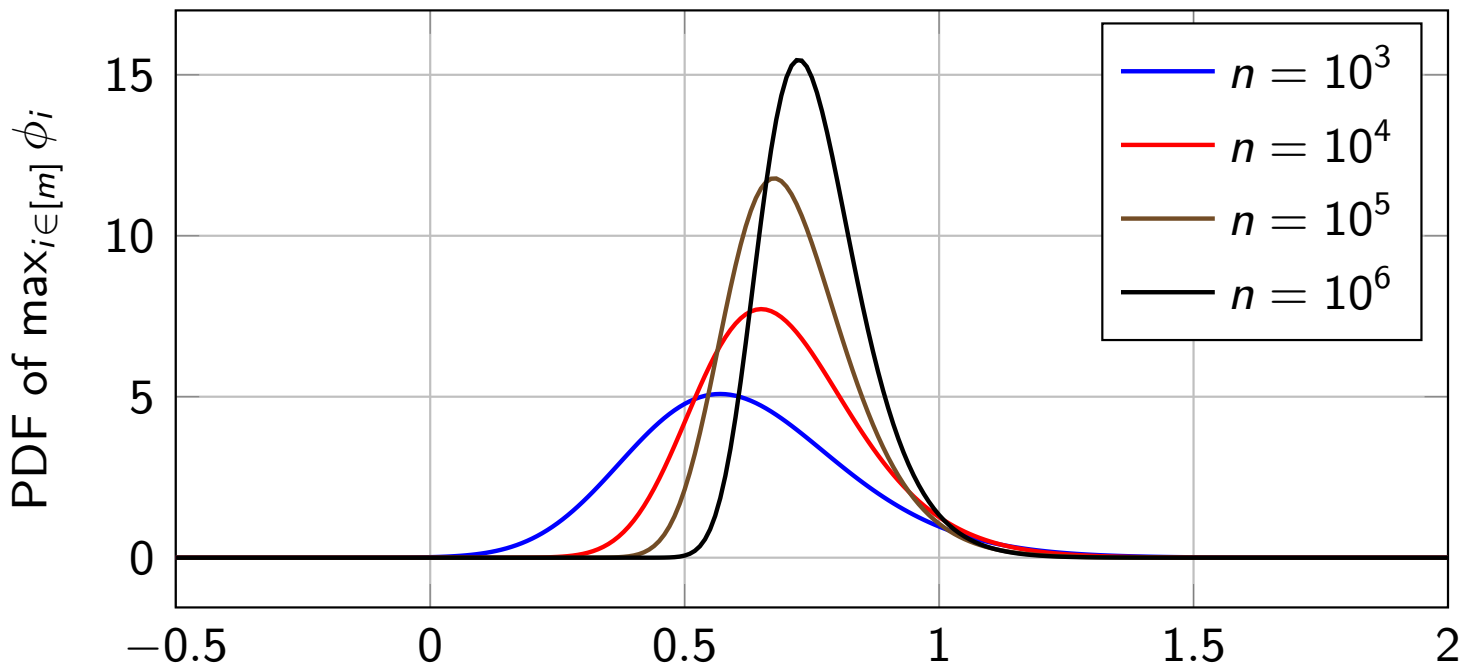
Unrealistic example (cont.)

- Assume that we randomly group the $n/\ln(n)$ subsets in $\sqrt{n/\ln(n)}$ groups, each containing $\sqrt{n/\ln(n)}$ subsets.
- After receiving the queries' responses, we pick the maximum from each group and we create the final query as the sample mean on those particular groups.



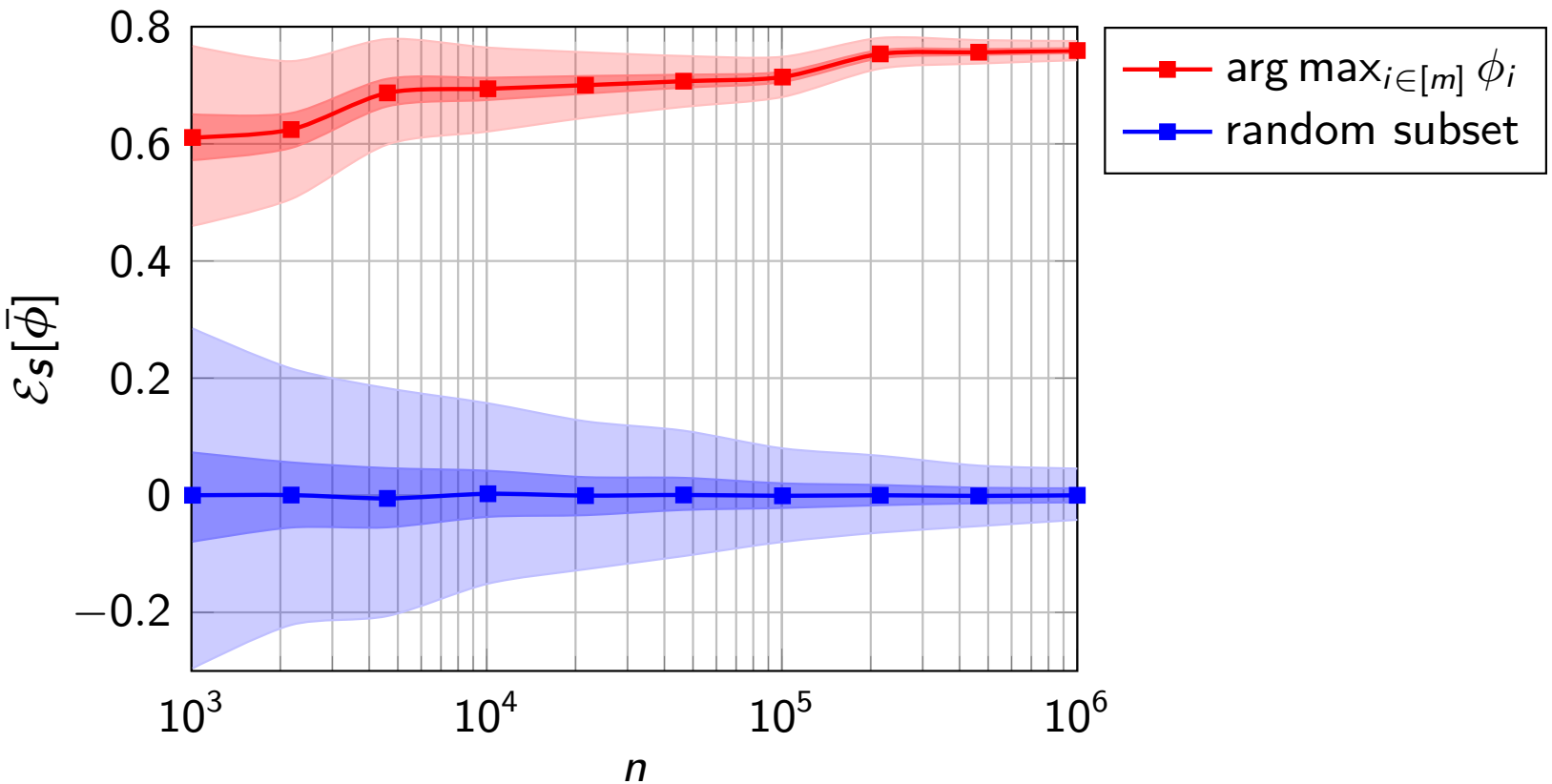
Unrealistic example (cont.)

- In contrast to choosing a random subset, whose sample mean behaves as expected: $\phi_j \sim \mathcal{N}\left(0, \frac{1}{\ln(n)}\right)$, choosing the maximum among the sample means in the group has a particular p.d.f. that does not concentrate around the true mean of X.



Unrealistic example (cont.)

- Therefore, this (unrealistic) method for improving the sample convergence does not result in $\mathcal{E}_{\boldsymbol{S}}[\bar{\phi}]$ being close to $\mathcal{P}[\bar{\phi}]$, even though $\mathcal{E}_{\boldsymbol{S}}[\phi_j]$ is close to $\mathcal{P}[\phi_j]$ for every fixed ϕ_j.



Generalization via differential privacy

Generalization via DP for i.i.d. datasets

Let us define $\mathcal{A}$, $\boldsymbol{S}$, and $\boldsymbol{Y}$, where

- $\mathcal{A} : \mathcal{X}^n \to \mathcal{Y}$ and it is an ε-DP algorithm,
- $\boldsymbol{S}$ is a RV drawn from a distribution $\mathcal{P}^n$ over $\mathcal{X}^n$, and
- $\boldsymbol{Y} = \mathcal{A}(\boldsymbol{S})$ is the corresponding output distribution.

Assume that for each element $y \in \mathcal{Y}$ there is a subset $R(y) \subseteq \mathcal{X}^n$ such that $\max_{y \in \mathcal{Y}} \Pr\{\boldsymbol{S} \in R(y)\} \leq \beta$.

Then, for $\varepsilon \leq \sqrt{\frac{\ln(1/\beta)}{2n}}$ we have $\Pr\{\boldsymbol{S} \in R(\boldsymbol{Y})\} \leq 3\sqrt{\beta}$.

Interpretation: In the statement, $R(y)$ denotes the set of datasets for which y is a "bad" output of the algorithm. If setting $\mathcal{A}$ to output a fix y is a bad idea with probability at most β, then the output $\boldsymbol{Y} = \mathcal{A}(\boldsymbol{S})$ of an ε-DP algorithm has a bounded probability of being bad for the particular input $\boldsymbol{S}$.

Generalization via differential privacy (cont.)

A statistical query is defined by a function $\phi : \mathcal{X} \to [0,1]$ and a tolerance τ. For a distribution $\mathcal{P}$ over $\mathcal{X}$ a valid response to such a query is any value v such that $\left|v - \mathcal{P}[\phi]\right| \leq \tau$.

By Hoeffding's inequality, a fixed query function verifies that

$$\Pr\Big\{\left|\mathcal{P}[\phi] - \mathcal{E}_{\boldsymbol{S}}[\phi]\right| > \tau\Big\} \leq 2\exp\big(-2n\tau^2\big).$$

Corollary for statistical queries

Let $\mathcal{A}$ be an ε-DP algorithm that outputs a function from $\mathcal{X}$ to $[0,1]$. For a distribution $\mathcal{P}$ over $\mathcal{X}$, let $\boldsymbol{S}$ be a random variable distributed according to $\mathcal{P}^n$ and let $\phi = \mathcal{A}(\boldsymbol{S})$.

Then for any $\tau > 0$, setting $\varepsilon \leq \sqrt{\tau^2 - \frac{\ln(2)}{2n}}$ ensures that

$$\Pr\Big\{\left|\mathcal{P}[\phi] - \mathcal{E}_{\boldsymbol{S}}[\phi]\right| > \tau\Big\} \leq 3\sqrt{2}\exp\big(-n\tau^2\big).$$

Generalization via differential privacy (cont.)

Proof: Apply general theorem with

- $R(\phi) = \{S \in \mathcal{X}^n : \left|\mathcal{P}[\phi] - \mathcal{E}_S[\phi]\right| \geq \tau\}$ and
- $\beta = 2\exp\big(-2n\tau^2\big)$.

This results in

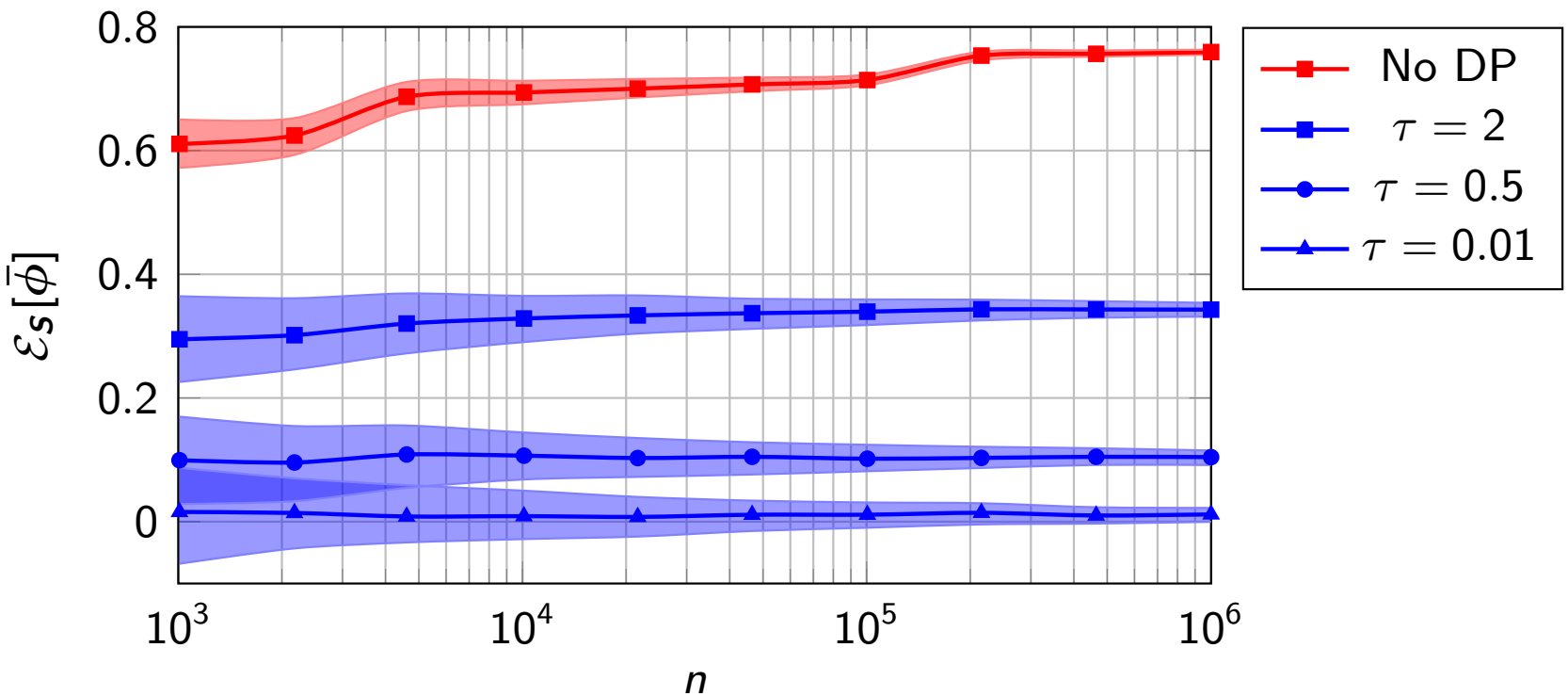
- $\varepsilon \leq \sqrt{\frac{\ln(1/\beta)}{2n}} = \sqrt{\tau^2 - \frac{\ln(2)}{2n}}$ and
- $$\begin{aligned}\Pr\{\boldsymbol{S} \in R(\boldsymbol{Y})\} &= \Pr\{\left|\mathcal{P}[\phi] - \mathcal{E}_{\boldsymbol{S}}[\phi]\right| > \tau\} \\ &\leq 3\sqrt{\beta} \\ &= 3\sqrt{2}\exp\big(-n\tau^2\big).\end{aligned}$$

Remark: Both papers show the same result but with different wording.

Unrealistic example (revisited)

We revisit the Gaussian example where now the $n/\ln(n)$ queries ϕ_j are answered by a DP mechanism following the previous guidelines.

Once the appropriate subsets are selected, the final query $\bar{\phi}$ is answered by a non-DP mechanism. This is possible since there will be no further adaptivity.



Max-divergence

For two random variables $\boldsymbol{X}$ and $\boldsymbol{Y}$ over the same domain $\mathcal{X}$, the max-divergence of $\boldsymbol{X}$ from $\boldsymbol{Y}$ is defined[3] as

$$D_\infty(\boldsymbol{X}\|\boldsymbol{Y}) = \log \max_{x\in\mathcal{X}} \frac{\Pr\{\boldsymbol{X}=x\}}{\Pr\{\boldsymbol{Y}=x\}}$$

while the δ-approximate max-divergence is defined as

$$D_\infty^\delta(\boldsymbol{X}\|\boldsymbol{Y}) = \log \max_{\mathcal{O}\subseteq\mathcal{X},\Pr\{\boldsymbol{X}\in\mathcal{O}\}\geq\delta} \frac{\Pr\{\boldsymbol{X}\in\mathcal{O}\}-\delta}{\Pr\{\boldsymbol{Y}\in\mathcal{O}\}}.$$

A randomized algorithm $\mathcal{A}$ with domain $\mathcal{X}^n$ is (ε,δ)-DP if for all pairs of datasets that differ in a single element $S, S' \in \mathcal{X}^n$:

$$D_\infty^\delta\big(\mathcal{A}(S)\|\mathcal{A}(S')\big) \leq (\log e)\,\varepsilon.$$

[3] In the following, log and ln denote the logarithm on base 2 and e, respectively.

Max-information

Max-information

Let $\boldsymbol{X}$ and $\boldsymbol{Y}$ be jointly distributed random variables. Then, the max-information between $\boldsymbol{X}$ and $\boldsymbol{Y}$ is defined as

$$I_\infty(\boldsymbol{X};\boldsymbol{Y}) = D_\infty\big((\boldsymbol{X},\boldsymbol{Y})\|\boldsymbol{X}\times\boldsymbol{Y}\big),$$

while the β-approximate max-information is defined as:

$$I_\infty^\beta(\boldsymbol{X};\boldsymbol{Y}) = D_\infty^\beta\big((\boldsymbol{X},\boldsymbol{Y})\|\boldsymbol{X}\times\boldsymbol{Y}\big).$$

Some (simple) properties:

- If $I_\infty(\boldsymbol{X};\boldsymbol{Y}) = k$, then $\Pr\{\boldsymbol{X}=x \mid \boldsymbol{Y}=y\} \leq 2^k \Pr\{\boldsymbol{X}=x\}$.
- The max-information is an upper bound of the mutual information: $I(\boldsymbol{X};\boldsymbol{Y}) \leq I_\infty(\boldsymbol{X};\boldsymbol{Y})$.
- For $\beta \geq 0$, $I_\infty^\beta(\boldsymbol{X};\boldsymbol{Y}) = I_\infty^\beta(\boldsymbol{Y};\boldsymbol{X})$.

The max-information of an algorithm

Max-information of an algorithm

We say that a (randomized) algorithm $\mathcal{A}$ has β-approximate max-information of value k if for every distribution $\mathcal{D}$ over $\mathcal{X}^n$, $I_\infty^\beta(\boldsymbol{S}; \mathcal{A}(\boldsymbol{S})) \leq k$, where $\boldsymbol{S}$ is a dataset chosen randomly according to $\mathcal{D}$. We denote this by $I_\infty^\beta(\mathcal{A}, n) \leq k$.

For the particular case of $\beta = 0$, we can alternatively define the (pure) max-information of algorithm $\mathcal{A}$ as

$$I_\infty(\mathcal{A}, n) = \max_{S, S' \in \mathcal{X}^n} D_\infty\big(\mathcal{A}(S) \| \mathcal{A}(S')\big),$$

where S and S' are any two datasets.

The max-information of an algorithm (cont.)

Post-processing

Let $\mathcal{A} : \mathcal{X}^n \to \mathcal{Y}$ and $\mathcal{B} : \mathcal{Y} \to \mathcal{Y}'$ be two randomized algorithms. Then, the algorithm $\mathcal{B} \circ \mathcal{A}$ with domain $\mathcal{X}^n$ and range $\mathcal{Y}'$ satisfies that, for every random variable $\boldsymbol{S}$ over $\mathcal{X}^n$ and every $\beta \geq 0$,

$$I_\infty^\beta\big(\boldsymbol{S}; \mathcal{B} \circ \mathcal{A}(\boldsymbol{S})\big) \leq I_\infty^\beta\big(\boldsymbol{S}; \mathcal{A}(\boldsymbol{S})\big).$$

Composition of algorithms

Let $\mathcal{A} : \mathcal{X}^n \to \mathcal{Y}$ be an algorithm such that $I_\infty^{\beta_1}(\mathcal{A}, n) \leq k_1$, and let $\mathcal{B} : \mathcal{X}^n \times \mathcal{Y} \to \mathcal{Z}$ be an algorithm such that, for every $y \in \mathcal{Y}$, $I_\infty^{\beta_2}\big(\boldsymbol{S}; \mathcal{B}(\boldsymbol{S}, y)\big) \leq k_2$.

Let $\mathcal{C} : \mathcal{X}^n \to \mathcal{Z}$ be defined such that $\mathcal{C}(S) = \mathcal{B}(S, \mathcal{A}(S))$. Then,

$$I_\infty^{\beta_1+\beta_2}(\mathcal{C}, n) \leq k_1 + k_2.$$

For a longer composition, we have that $I_\infty^{\sum_j \beta_j}(\mathcal{C}', n) \leq \sum_j k_j$.

The max-information of an algorithm (cont.)

Usefulness of max-information

Let $\boldsymbol{S}$ be a random dataset in $\mathcal{X}^n$ and $\mathcal{A}$ be an algorithm with range $\mathcal{Y}$ such that for some $\beta \geq 0$, $I_\infty^\beta(\mathcal{A}, n) = k$. Then for any event $\mathcal{O} \subseteq \mathcal{X}^n \times \mathcal{Y}$,

$$\Pr\{(\boldsymbol{S}, \mathcal{A}(\boldsymbol{S})) \in \mathcal{O}\} \leq 2^k \Pr\{\boldsymbol{S} \times \mathcal{A}(\boldsymbol{S}) \in \mathcal{O}\} + \beta.$$

In particular,

$$\Pr\{(\boldsymbol{S}, \mathcal{A}(\boldsymbol{S})) \in \mathcal{O}\} \leq 2^k \max_{y \in \mathcal{Y}} \Pr\{(\boldsymbol{S}, y) \in \mathcal{O}\} + \beta.$$

As before, this allows us to bound the probability of any joint event between the random dataset and random output of the algorithm, i.e., an adaptive behavior, with respect to the independent selection of $\boldsymbol{S}$ and $\mathcal{A}(\boldsymbol{S})$.

The max-information of DP algorithms

The (approximate) max-information of $\mathcal{A}$, i.e., $I_\infty^\beta\big(\boldsymbol{S}; \mathcal{A}(\boldsymbol{S})\big) = I_\infty^\beta(\mathcal{A}, n)$, is a measure of how much the random dataset $\boldsymbol{S}$ affects the distribution of the algorithm's output. In other words, how much information from $\boldsymbol{S}$ is leaked to the output of $\mathcal{A}$.

Pure max-information of pure DP algorithms

Let $\mathcal{A}$ be an ε-DP algorithm. Then $I_\infty(\mathcal{A}, n) \leq (\log e)\,\varepsilon n$.

Proof: Given that any two datasets S and S' might differ in at most n elements, for every output y of $\mathcal{A}$ we have that

$$\Pr\{\boldsymbol{Y} = y \mid \boldsymbol{S} = S\} \leq \exp(\varepsilon n) \Pr\{\boldsymbol{Y} = y \mid \boldsymbol{S} = S'\},$$

which implies that $D_\infty\big(\mathcal{A}(S) \| \mathcal{A}(S')\big) \leq (\log e)\,\varepsilon n$.

Therefore,

$$I_\infty(\mathcal{A}, n) = \max_{S, S' \in \mathcal{X}^n} D_\infty\big(\mathcal{A}(S) \| \mathcal{A}(S')\big) \leq (\log e)\,\varepsilon n. \qquad \blacksquare$$

The max-information of DP algorithms (cont.)

The previous result applied to general distributions of datasets but we can obtain a tighter bound for datasets with i.i.d. samples.

Approx. max-information of DP algorithms for i.i.d. datasets

Let us define $\mathcal{A}$, $\boldsymbol{S}$, and $\boldsymbol{Y}$, where

- $\mathcal{A} : \mathcal{X}^n \to \mathcal{Y}$ and it is an ε-DP algorithm,
- $\boldsymbol{S}$ is a RV drawn from a distribution $\mathcal{P}^n$ over $\mathcal{X}^n$, and
- $\boldsymbol{Y} = \mathcal{A}(\boldsymbol{S})$ is the corresponding output distribution.

Then for any $\beta > 0$, $I_\infty^\beta(\mathcal{A}, n) \leq (\log e)\big(\varepsilon^2 n/2 + \varepsilon\sqrt{n \ln(2/\beta)/2}\big)$.

For small ε and fixed $\beta > 0$, the bound on $I_\infty^\beta(\mathcal{A}, n)$ is $\mathcal{O}(\varepsilon\sqrt{n})$.

For comparison, remember that the composition of k ε-DP mechanisms results in a (ε', δ')-DP mechanism with $\varepsilon' = \mathcal{O}(\varepsilon\sqrt{k})$.

Generalization via max-information

Let $\mathcal{A}$ be an algorithm that outputs a function $f : \mathcal{X}^n \to \mathbb{R}$ of sensitivity c and define the "bad event" $\mathcal{O}_\tau$ when the empirical estimate of f is far from its expected value, i.e.,

$$\mathcal{O}_\tau = \left\{(S, f) : \left|f(S) - \mathcal{D}[f]\right| \geq \tau\right\},$$

where $\mathcal{D}$ is a general distribution over $\mathcal{X}^n$.

By McDiarmid's inequality, if the samples are i.i.d. ($\mathcal{D} = \mathcal{P}^n$), then for every possible function f we have that

$$\Pr\{(\boldsymbol{S}, f) \in \mathcal{O}_\tau\} = \Pr\left\{\left|f(\boldsymbol{S}) - \mathcal{P}^n[f]\right| \geq \tau\right\} \leq 2\exp\left(-\frac{2\tau^2}{nc^2}\right).$$

Generalization via max-information (cont.)

Generalization via max-information for i.i.d. datasets

Let us define $\mathcal{A}$, $\boldsymbol{S}$, and $\boldsymbol{f}$, where

- $\mathcal{A}$ is an algorithm that outputs a c-sensitive function $f : \mathcal{X}^n \to \mathbb{R}$,
- $\boldsymbol{S}$ is a random dataset drawn according to $\mathcal{P}^n$, and
- $\boldsymbol{f} = \mathcal{A}(\boldsymbol{S})$ is the corresponding random output of $\mathcal{A}$.

If for $\beta \geq 0$ and $\tau > 0$, $I_\infty^\beta(\mathcal{A}, n) \leq (\log e)\frac{\tau^2}{nc^2}$, then

$$\Pr\Big\{\big|\boldsymbol{f}(\boldsymbol{S}) - \mathcal{P}^n[\boldsymbol{f}]\big| \geq \tau\Big\} \leq 2\exp\Big(-\frac{\tau^2}{nc^2}\Big) + \beta.$$

In particular, if $\mathcal{A}$ is a $\big(\frac{\tau^2}{n^2c^2}\big)$-DP algorithm, then

$$\Pr\Big\{\big|\boldsymbol{f}(\boldsymbol{S}) - \mathcal{P}^n[\boldsymbol{f}]\big| \geq \tau\Big\} \leq 2\exp\Big(-\frac{\tau^2}{nc^2}\Big).$$

Generalization via max-information (cont.)

If f is a statistical query $\phi : \mathcal{X} \to [0,1]$, we have that:

- $f(S) = \frac{1}{n}\sum_{i=1}^{n} \phi(x_i) = \mathcal{E}_S[\phi]$,
- $c = \frac{1}{n}$, and
- $\mathcal{P}^n[f] = \frac{1}{n}\sum_{i=1}^{n} \mathcal{P}^n[\phi(x_i)] = \mathcal{P}[\phi]$.

Therefore, the previous result may be rewritten as follows:

If $\mathcal{A}$ is a τ^2-DP algorithm, then $I_\infty(\mathcal{A}, n) \leq (\log e)\, n\tau^2$ and

$$\Pr\Big\{\big|\mathcal{E}_{\boldsymbol{S}}[\phi] - \mathcal{P}[\phi]\big| \geq \tau\Big\} \leq 2\exp\big(-n\tau^2\big).$$

Compare to our former bound that for $\mathcal{A}$ being an ε-DP algorithm, setting $\varepsilon \leq \sqrt{\tau^2 - \frac{\ln(2)}{2n}}$ ensures that

$$\Pr\Big\{\big|\mathcal{E}_{\boldsymbol{S}}[\phi] - \mathcal{P}[\phi]\big| \geq \tau\Big\} \leq 3\sqrt{2}\exp\big(-n\tau^2\big).$$

Generalization via max-information (cont.)

Given that the previous result requires a very restrictive privacy level of $\varepsilon = \tau^2$, we can try to use the other (tighter) bound on generalization via max-information.

Generalization via max-information for i.i.d. datasets (take 2)

Let us define $\mathcal{A}$, $\boldsymbol{S}$, and $\boldsymbol{f}$ as before. If $\mathcal{A}$ is $\left(\frac{\tau}{nc}\right)$-DP algorithm, then

$$\Pr\left\{\left|\boldsymbol{f}(\boldsymbol{S}) - \mathcal{P}^n[\boldsymbol{f}]\right| \geq \tau\right\} \leq 2\exp\left(-\frac{3}{4}\frac{\tau^2}{nc^2}\right),$$

for large $\frac{\tau^2}{nc^2}$ (more than 10 seems OK).

Proof: Let $\beta = 2\exp\left(-\frac{\tau^2}{nc^2}\right)$, then using the tighter bound on the approx. max-information we have $I_\infty^\beta(\mathcal{A}, n) \leq (\log e)\frac{\tau^2}{nc^2}\left(\frac{1}{2} + \frac{1}{\sqrt{2}}\right)$.

Applying this to the bound on the prob. of joint events we obtain $\Pr\left\{\left|\boldsymbol{f}(\boldsymbol{S}) - \mathcal{P}^n[\boldsymbol{f}]\right| \geq \tau\right\} \leq 2\exp\left(\left(\frac{1}{\sqrt{2}} - \frac{3}{2}\right)\frac{\tau^2}{nc^2}\right) + 2\exp\left(-\frac{\tau^2}{nc^2}\right)$. ■

Reusable holdout

In this last part, we present one differentially private algorithm to answer adaptively chosen statistical queries on a holdout set: the Thresholdout algorithm which is based on the "Sparse Vector Algorithm."

The algorithm accepts any statistical query $\phi : \mathcal{X} \to [0, 1]$ and its goal is to provide an estimate of $\mathcal{P}[\phi]$ using a holdout set for validation.

Thresholdout

Thresholdout is given access to:

- the training dataset S_t,
- the holdout dataset S_h, and
- a budget limit B.

Given a function ϕ, Thresholdout checks if

$$\left|\mathcal{E}_{S_t}[\phi] - \mathcal{E}_{S_h}[\phi]\right| \leqslant T + \eta,$$

where T is a fixed threshold and η is Laplace noise:

- If below the threshold, then the algorithm returns $\mathcal{E}_{S_t}[\phi]$.
- If above the threshold, then the algorithm returns $\mathcal{E}_{S_h}[\phi] + \xi$, for another Laplacian noise variable ξ, and the budget B is reduced by one.

Thresholdout (cont.)

Given a budget limit $B > 0$ and a noise rate $\sigma > 0$, the Thresholdout algorithm is:

- $\left(\frac{2B}{\sigma n}\right)$-differentially private, or
- $\left(\frac{2\sqrt{2B\ln(2/\delta)}}{\sigma n}, \delta\right)$-differentially private for any $\delta > 0$.

Sketch of proof: Thresholdout is an instance of the Sparse Vector Algorithm (when we check if above or below the threshold) together with the Laplace mechanism (when we release the average on S_h).

Our previous analysis of Sparse was for counting queries, which are 1-sensitive, but now we deal with empirical averages, which are $\frac{1}{n}$-sensitive queries.

Thresholdout (cont.)

In our instance, the holdout set S_h is the private data set, and each function ϕ corresponds to the following query:

$$f_\phi(S_h) = \left|\mathcal{E}_{S_h}[\phi] - \mathcal{E}_{S_t}[\phi]\right|.$$

Thresholdout then is equivalent to the following procedure:

- We run the sparse vector algorithm with $c = B$, queries f_ϕ for each function ϕ, and noise rate 2σ.
- Whenever an above-threshold query is found, we release its value using the Laplace mechanism with noise rate σ.

Each mechanism is simultaneously:

- $\left(\frac{B}{\sigma n}\right)$-differentially private, or
- $\left(\frac{\sqrt{2B\ln(2/\delta)}}{\sigma n}, \frac{\delta}{2}\right)$-differentially private[4] for any $\delta > 0$. ■

[4] This value differs from the one in the article by a factor of 2. The reason is that I am being consistent with our definition of the best ε used in the *advanced* composition of B mechanisms.

Take-away messages

- Differentially private algorithms can be used to obtain bounds on the probability of overfitting in adaptive data analysis given functions that "generalizes well" in a non-adaptive setting.
- The max-information of an algorithm is an interesting concept that generalizes DP in the context of data analysis.
  - The max-information exhibits similar properties as DP.
  - DP algorithms have bounded max-information.
- Thresholdout is a specific DP mechanism that allows the reuse of a holdout dataset with rigorous generalization guarantees.

Outline

Notes on the unrealistic example

Let $X_i \sim \mathcal{N}(0, \sigma)$ and let $Y = \max_{i \in [n]} X_i$, then

$$\Pr\{Y \leq y\} = \Phi\left(\frac{y}{\sigma}\right)^n,$$

where Φ is the CDF of the standard Gaussian distribution.

Consequently, the PDF of Y is

$$f_Y(y) = \frac{n}{\sigma}\phi\left(\frac{y}{\sigma}\right)\Phi\left(\frac{y}{\sigma}\right)^{n-1},$$

where ϕ is the PDF of the standard Gaussian distribution.

Notes on the unrealistic example (cont.)

To create the Laplace mechanism for this example, I did as follows:

- Each query ϕ_j is the sample mean of the jth subset, i.e., $\phi_j \sim \mathcal{N}(0, 1/\ln(n))$.
- The subsets are disjoint, and thus the $n/\ln(n)$ queries are disjoint. This helps reduce the amount of noise added.
- With probability $1 - 2Q(3) \approx 0.9973$, each sample satisfies $|X_i| \leq 3$. Therefore, I assumed that $\Delta\phi_j = \frac{3}{\ln(n)}$ (with high probability) and neglected the effects of $|X_i| > 3$.
- Given that $\Pr\{|\mathcal{E}_{\boldsymbol{S}}[\phi_j]| \geq t\} = 2Q(t\sqrt{\ln(n)}) = \beta$, according to the generalization result, I took $\varepsilon = \sqrt{\frac{\ln(1/\beta)}{2\ln(n)}}$ and added independent noises $\text{Lap}\left(\frac{\Delta\phi_j}{\varepsilon}\right)$ to every element of the subset before choosing the maximum. Only the index was reported.

References

- C. Dwork and A. Roth, "The algorithmic foundations of differential privacy," *Foundations and Trends in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.
- C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, and A. Roth, "Preserving Statistical Validity in Adaptive Data Analysis," *arXiv:1411.2664 [cs]*, Mar. 2016.
- C. Dwork, V. Feldman, M. Hardt, T. Pitassi, O. Reingold, and A. Roth, "Generalization in Adaptive Data Analysis and Holdout Reuse," *arXiv:1506.02629 [cs]*, Jun. 2015.