CS 7775

Seminar in Computer Security: Machine Learning Security and Privacy

Matthew Jagielski
Google DeepMind

October 26, 2023

Deep Learning with Differential Privacy

Martín Abadi, Andy Chu, Ian GoodFellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, Li Zhang

Motivation for Differential Privacy

- **Anonymization is not enough!**
  - Netflix Prize
    - Contestants were given a dataset of user ratings where identifying information was anonymized and competed to create a recommendation engine
    - Researchers were able to de-anonymize the dataset by linking user ratings to public IMDB ratings
- Ideally, we need a way to learn general trends of the dataset without revealing any individual's private information/contribution

Intuition Behind Differential Privacy

- Suppose we have an algorithm/model, M, which outputs the probability that a student has an F in CY7725
- M_D is trained on dataset D, and when we make the query M_D(John), the output is 0.55
- M_{D+John} is trained on dataset D+{John}, and when we make the query M_{D+John}(John), the output is 0.57
- Outputs similar → “membership inference” is hard → not much leakage

[Mukul Rathi]

Intuition Behind Differential Privacy

- What if M_{D+John}(John) outputs 0.80?
- Then we have high confidence that John is failing CY7725
- Higher accuracy on our test set at the cost of privacy

[Mukul Rathi]

How do we prevent this privacy leakage?

Differential Privacy

- Let D, D’ be neighboring datasets ($||D - D'||_1 = 1$, differ in one row), O be some potential output of an algorithm A

- Then, differential privacy considers the “privacy loss”:

$$\ln\left(\frac{\mathbb{P}[A(D') \in O]}{\mathbb{P}[A(D) \in O]}\right) \leq \epsilon$$

Differential Privacy

- Let D, D’ be neighboring datasets ($||D - D'||_1 = 1$, differ in one row), O be some potential output of an algorithm A

- Then, differential privacy considers the “privacy loss”:

$$\ln\left(\frac{\mathbb{P}[A(D') \in O]}{\mathbb{P}[A(D) \in O]}\right) \leq \epsilon$$

“Privacy loss”

Bound on privacy loss, the “privacy budget”

Differential Privacy

- Let D, D’ be neighboring datasets ($||D - D'||_1 = 1$, differ in one row), O be some potential output of an algorithm A

- Alternatively, ε-differential privacy (epsilon-DP) enforces:

$$\mathbb{P}[A(D') \in O] \leq \exp(\varepsilon)\mathbb{P}[A(D) \in O]$$

“Approximate” Differential Privacy

- Let D, D’ be neighboring datasets ($||D - D'||_1 = 1$, differ in one row), O be some potential output of an algorithm A

- “Approximate” (ε, δ)-differential privacy enforces:

$$\mathbb{P}[A(D') \in O] \leq \exp(\varepsilon)\mathbb{P}[A(D) \in O] + \delta$$

Basic DP Algorithms

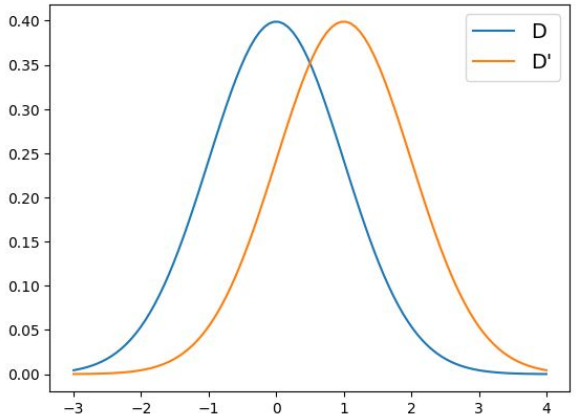
- Randomized Response [Warner65]
  - “Have you ever had an abortion?”
  - Input: {Yes, No}
  - Flip a coin
    - If heads, answer correctly
    - If tails, answer randomly
  - What ε does this satisfy?

$$\mathbb{P}[A(D') \in O] \leq \exp(\varepsilon)\mathbb{P}[A(D) \in O]$$

See https://en.wikipedia.org/wiki/Randomized_response for more

Basic DP Algorithms

- Gaussian Mechanism
  - Input: Database of {Yes, No} responses for each of n people
  - Goal: Count how many answered “Yes”
  - Sensitivity: Each person changes one count
  - Algorithm
    - Sample Gaussian Noise v ~ N(0, σ^2)
    - Return (true count + v)



See http://www.gautamkamath.com/CS860notes/lec5.pdf for more

Basic DP Algorithms

- Gaussian Mechanism
  - Input: Database of {Yes, No} responses for each of n people
  - Goal: Count how many answered “Yes”
  - Sensitivity: Each person changes one count
  - Algorithm
    - Sample Gaussian Noise v ~ N(0, σ^2)
    - Return (true count + v)
  - What ε, δ does this satisfy?

$$\mathbb{P}[A(D') \in O] \leq \exp(\varepsilon)\mathbb{P}[A(D) \in O] + \delta$$

$$p(v) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{v^2}{2\sigma^2}\right) \qquad p(v') = \frac{1}{\sigma\sqrt{2\pi}}\exp\left(-\frac{(v+1)^2}{2\sigma^2}\right)$$

See http://www.gautamkamath.com/CS860notes/lec5.pdf for more

Properties

- **Composition:** If we run k (ε, δ)-DP algorithms:
  - The resulting algorithm will be ($k\varepsilon, k\delta$)-DP
- **Advanced composition:** If $k < 1/\varepsilon^2$ the resulting algorithm is ($O\left(\sqrt{k\log(1/\delta')}\right)\varepsilon, k\delta + \delta'$)-DP for all $\delta' > 0$
- **Amplification:** If we sample a fraction q, of the data, our algorithm becomes ($q\varepsilon, q\delta$)-DP

DP-SGD Algorithm

1. Compute gradient of random sample

2. Clip gradient to some ℓ_2 norm C ("bound sensitivity of the gradient")

3. Add noise calibrated proportional to the clipping norm, C and noise scale sigma

4. Update

Algorithm 1 Differentially private SGD (Outline)

Input: Examples $\{x_1, \ldots, x_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N}\sum_i \mathcal{L}(\theta, x_i)$. Parameters: learning rate η_t, noise scale σ, group size L, gradient norm bound C.

Initialize θ_0 randomly

for $t \in [T]$ **do**

Take a random sample L_t with sampling probability L/N

Compute gradient

For each $i \in L_t$, compute $\mathbf{g}_t(x_i) \leftarrow \nabla_{\theta_t}\mathcal{L}(\theta_t, x_i)$

Clip gradient

$\bar{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i) / \max\left(1, \frac{\|\mathbf{g}_t(x_i)\|_2}{C}\right)$

Add noise

$\tilde{\mathbf{g}}_t \leftarrow \frac{1}{L}\left(\sum_i \bar{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})\right)$

Descent

$\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{\mathbf{g}}_t$

Output θ_T and compute the overall privacy cost (ε, δ) using a privacy accounting method.

Analysis

- **Basic Approach**
  - Use basic composition
    - $(T\varepsilon, T\delta)$-DP
  - Amplification from subsampling gives
    - $(qT\varepsilon, qT\delta)$-DP

Algorithm 1 Differentially private SGD (Outline)

Input: Examples $\{x_1, \ldots, x_N\}$, loss function $\mathcal{L}(\theta) = \frac{1}{N}\sum_i \mathcal{L}(\theta, x_i)$. Parameters: learning rate η_t, noise scale σ, group size L, gradient norm bound C.

Initialize θ_0 randomly

for $t \in [T]$ **do**

Take a random sample L_t with sampling probability L/N

Compute gradient

For each $i \in L_t$, compute $\mathbf{g}_t(x_i) \leftarrow \nabla_{\theta_t}\mathcal{L}(\theta_t, x_i)$

Clip gradient

$\bar{\mathbf{g}}_t(x_i) \leftarrow \mathbf{g}_t(x_i) / \max\left(1, \frac{\|\mathbf{g}_t(x_i)\|_2}{C}\right)$

Add noise

$\tilde{\mathbf{g}}_t \leftarrow \frac{1}{L}\left(\sum_i \bar{\mathbf{g}}_t(x_i) + \mathcal{N}(0, \sigma^2 C^2 \mathbf{I})\right)$

Descent

$\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{\mathbf{g}}_t$

Output θ_T and compute the overall privacy cost (ε, δ) using a privacy accounting method.

Analysis

- **Using Advanced Composition**
  - $\left(O\left(\sqrt{T\log(1/\delta')}\right)\varepsilon, T\delta+\delta'\right)$-DP
  - Privacy budget scales with $\sqrt{T}$ instead of T, nice!

Algorithm 1 Differentially private SGD (Outline)

Input: Examples $\{x_1,\ldots,x_N\}$, loss function $\mathcal{L}(\theta)=\frac{1}{N}\sum_i\mathcal{L}(\theta,x_i)$. Parameters: learning rate η_t, noise scale σ, group size L, gradient norm bound C.

Initialize θ_0 randomly

for $t\in[T]$ **do**

Take a random sample L_t with sampling probability L/N

Compute gradient

For each $i\in L_t$, compute $\mathbf{g}_t(x_i)\leftarrow\nabla_{\theta_t}\mathcal{L}(\theta_t,x_i)$

Clip gradient

$\bar{\mathbf{g}}_t(x_i)\leftarrow\mathbf{g}_t(x_i)/\max\left(1,\frac{\|\mathbf{g}_t(x_i)\|_2}{C}\right)$

Add noise

$\tilde{\mathbf{g}}_t\leftarrow\frac{1}{L}\left(\sum_i\bar{\mathbf{g}}_t(x_i)+\mathcal{N}(0,\sigma^2C^2\mathbf{I})\right)$

Descent

$\theta_{t+1}\leftarrow\theta_t-\eta_t\tilde{\mathbf{g}}_t$

Output θ_T and compute the overall privacy cost (ε,δ) using a privacy accounting method.

Does advanced composition give us the best bound?

Analysis

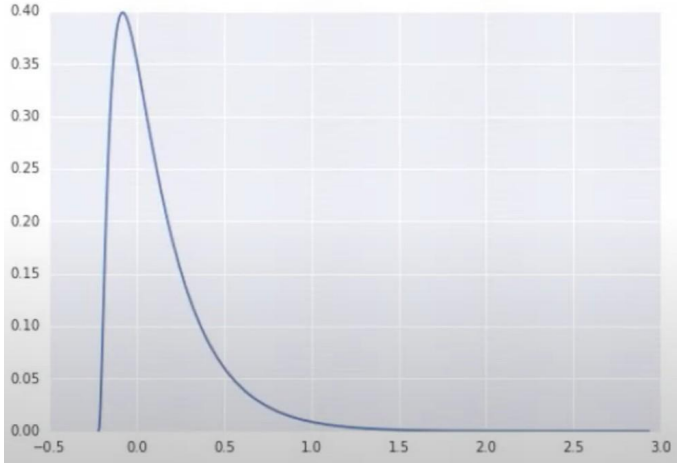
The Moments Accountant

- Treat privacy loss as a random variable

$$c(o; \mathcal{M}, \text{aux}, d, d') \triangleq \log \frac{\Pr[\mathcal{M}(\text{aux}, d) = o]}{\Pr[\mathcal{M}(\text{aux}, d') = o]}.$$

- For subsampled Gaussian, privacy loss has a quick drop-off and a very long tail

- We can use **concentration inequalities** to bound privacy loss



log(privacy loss)

Analysis

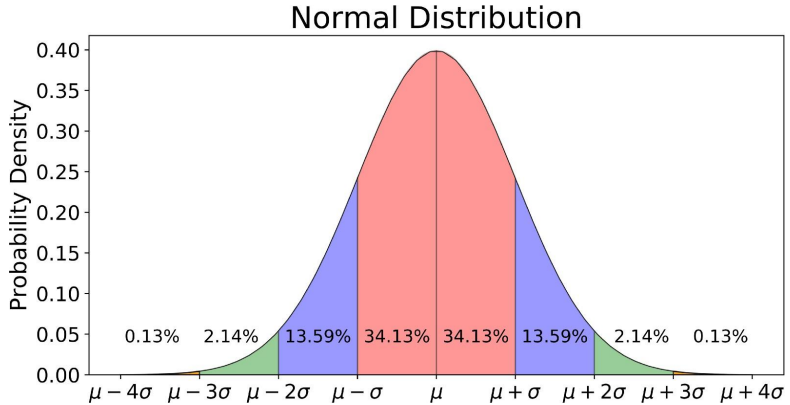
Standard Concentration Inequalities

- Markov's Inequality:

$$\mathbb{P}[X \geq a] \leq \frac{\mathbb{E}[X]}{a}$$

- Chebyshev's Inequality:

$$\mathbb{P}[|X - \mathbb{E}[X]| \geq a] \leq \mathbb{E}\left[\frac{|X-\mu|^2}{a^2}\right] = \frac{\mathrm{Var}[X]}{a^2}$$



Analysis

Concentration Inequalities

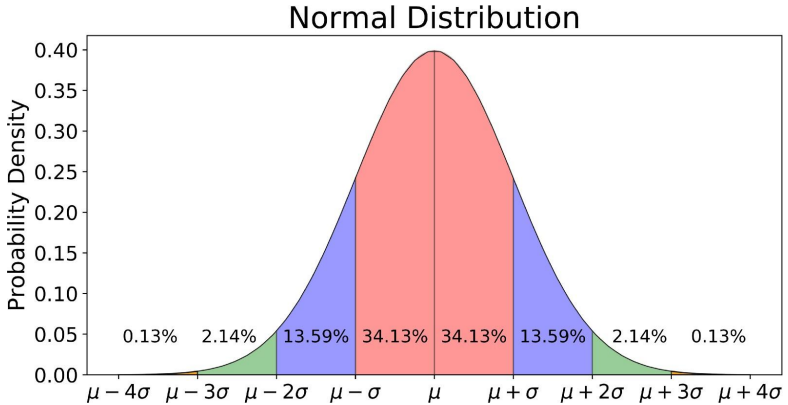
- Can handle higher moments with the moment generating function:

$$\mathbb{P}[|X - \mathbb{E}[X]| > a] \leq \mathbb{E}[\exp(\lambda X)] \exp(-\lambda a)$$

- And prove DP with their Theorem

*[**Tail bound**] For any $\varepsilon > 0$, the mechanism $\mathcal{M}$ is (ε, δ)-differentially private for*

$$\delta = \min_{\lambda} \exp(\alpha_{\mathcal{M}}(\lambda) - \lambda\varepsilon)\,.$$



Analysis

- **Concentration Inequalities**
  - Then, we can optimize λ to obtain the tightest result for specific parameters
  - (For practical reasons, the authors only keep track of the first 32 moments)
- Using this approach, the authors find that the algorithm is ($O\left(q\varepsilon\sqrt{T}\right), \delta$)-DP

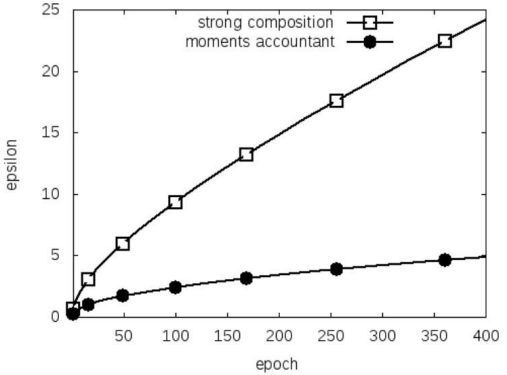


Figure 2: The ε value as a function of epoch E for $q = 0.01$, $\sigma = 4$, $\delta = 10^{-5}$, using the strong composition theorem and the moments accountant respectively.

Empirical Results

- Using a simple feed-forward neural network trained on MNIST the authors achieved the following results
  - No privacy 98.3% accuracy
  - $\epsilon = 8, \quad \delta = 10^{-5}$ $\longrightarrow$ 97% accuracy (Compared to 80% in previous work)
  - $\epsilon = 2, \quad \delta = 10^{-5}$ $\longrightarrow$ 95% accuracy
  - $\epsilon = 0.5, \delta = 10^{-5}$ $\longrightarrow$ 90% accuracy

Empirical Results

- Using a simple feed-forward neural network trained on CIFAR10 the authors achieved the following results
  - No privacy 80% accuracy
  - $\epsilon = 8, \quad \delta = 10^{-5}$ $\rightarrow$ 73% accuracy
  - $\epsilon = 2, \quad \delta = 10^{-5}$ $\rightarrow$ 67% accuracy

Strengths

- The authors show that differentially private deep learning can achieve good accuracy with a reasonable amount of privacy leakage
- Moments accountant is a strong tool for computing privacy of iterative algorithms
  - It gives us more room for accuracy without the privacy loss found using advanced composition

Limitations

- A reasonable tradeoff between privacy and accuracy does not seem to hold for more complicated datasets, like CIFAR10
- The authors only consider a simple neural network in their experiments

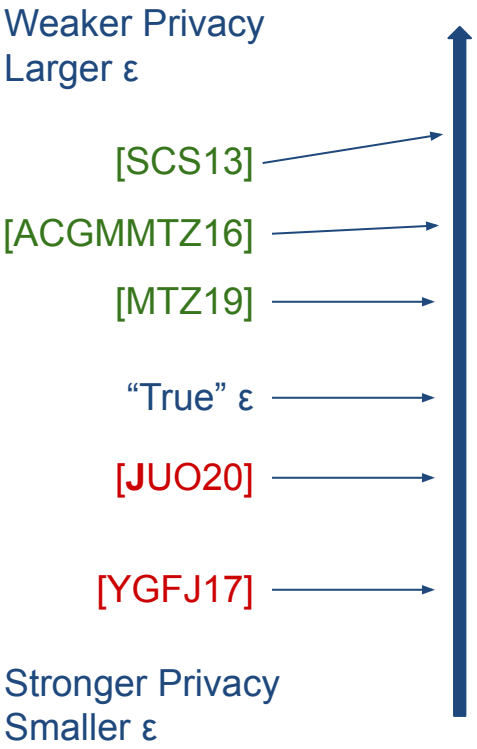
Auditing DP ML: How Private is DP-SGD?

Matthew Jagielski, Jonathan Ullman, Alina Oprea

Problem

- Proving DP is hard
- Implementing DP algorithms correctly is hard
- In practice, unclear epsilon we should be comfortable with

How Private is DP-SGD?



We use “poisoning attacks” to tighten the gap!

[SCS13] - https://cseweb.ucsd.edu/~kamalika/pubs/scs13.pdf
[ACGMMTZ16] - https://arxiv.org/abs/1607.00133
[MTZ19] - https://arxiv.org/abs/1908.10530

[JUO20] - https://arxiv.org/abs/2006.07709
[YGFJ17] - https://arxiv.org/abs/1709.01604

Statistically Measuring Differential Privacy

- Given algorithm A, adjacent datasets D_0 and D_1 and outputs O:

$$\Pr[\mathcal{A}(D_0) \in \mathcal{O}] \leq e^{\varepsilon} \Pr[\mathcal{A}(D_1) \in \mathcal{O}]$$

- Begin by computing probabilities:

$$p_0 = \Pr[\mathcal{A}(D_0) \in \mathcal{O}] \qquad p_1 = \Pr[\mathcal{A}(D_1) \in \mathcal{O}]$$

- Lower bound on Epsilon:

$$\varepsilon_{LB} = \ln(p_0/p_1)$$

Statistically Measuring Differential Privacy

- To approximate p_0 and p_1, use Monte Carlo Estimation
- Clopper Pearson confidence intervals: Produce epsilon lower bound with probability 99%

Algorithm 2: Empirically Measuring ε

Data: Algorithm $\mathcal{A}$, datasets D_0, D_1 at distance k, output set $\mathcal{O}$, trial count T, confidence level α

$\mathrm{ct}_0 = 0, \mathrm{ct}_1 = 0$

For $i \in [T]$

  If $\mathcal{A}(D_0) \in \mathcal{O}$ $\mathrm{ct}_0 = \mathrm{ct}_0 + 1$

  If $\mathcal{A}(D_1) \in \mathcal{O}$ $\mathrm{ct}_1 = \mathrm{ct}_1 + 1$

$\hat{p}_0 = \textsc{ClopperPearsonLower}(\mathrm{ct}_0, T, \alpha/2)$

$\hat{p}_1 = \textsc{ClopperPearsonUpper}(\mathrm{ct}_1, T, \alpha/2)$

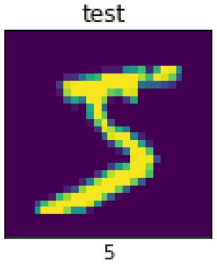
Return $\varepsilon_{LB} = \ln(\hat{p}_0/\hat{p}_1)/k$

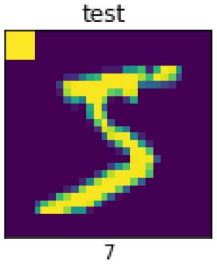
How do we pick O and D0, D1?

- D0, D1 should differ in 1 record, but result in different models
  - The record should significantly change the model
- Idea: Use poisoning attacks!
- Standard poisoning attacks will help a lot, we also have a better version
- For any poisoning attack, can measure its success as an “output set”

Backdoor Attack

- Introduce a Backdoor pattern on images during training phase with the desired target label.
- Apply the backdoor pattern on test data and check if the test samples get misclassified to the target label.





Auditing DP

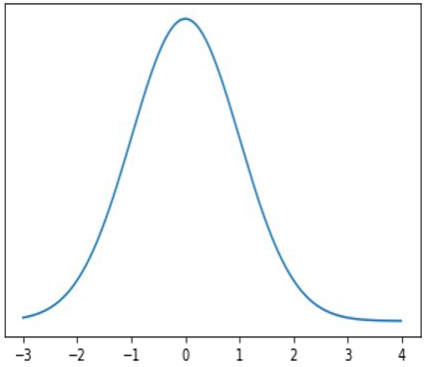
Original Database

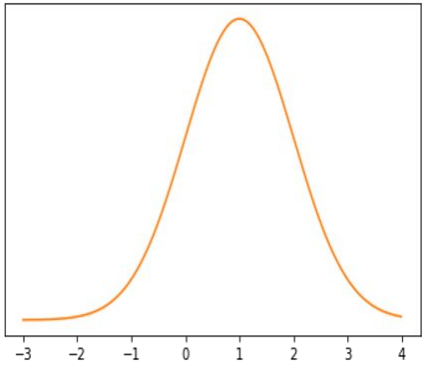
Train many models

Generate and add poisoning

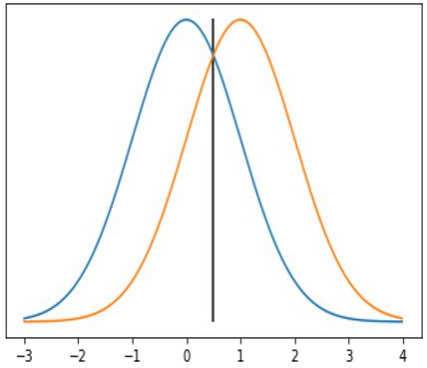
Poisoned Database

Train many models



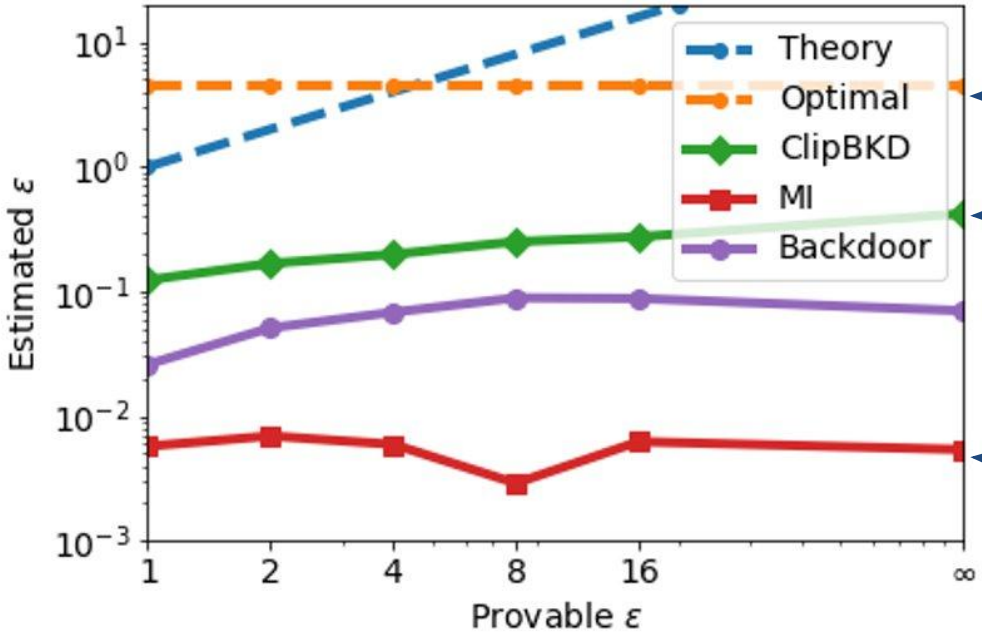


Measure poisoning effectiveness



Can we identify whether the dataset was poisoned?

The Gap Tightens!



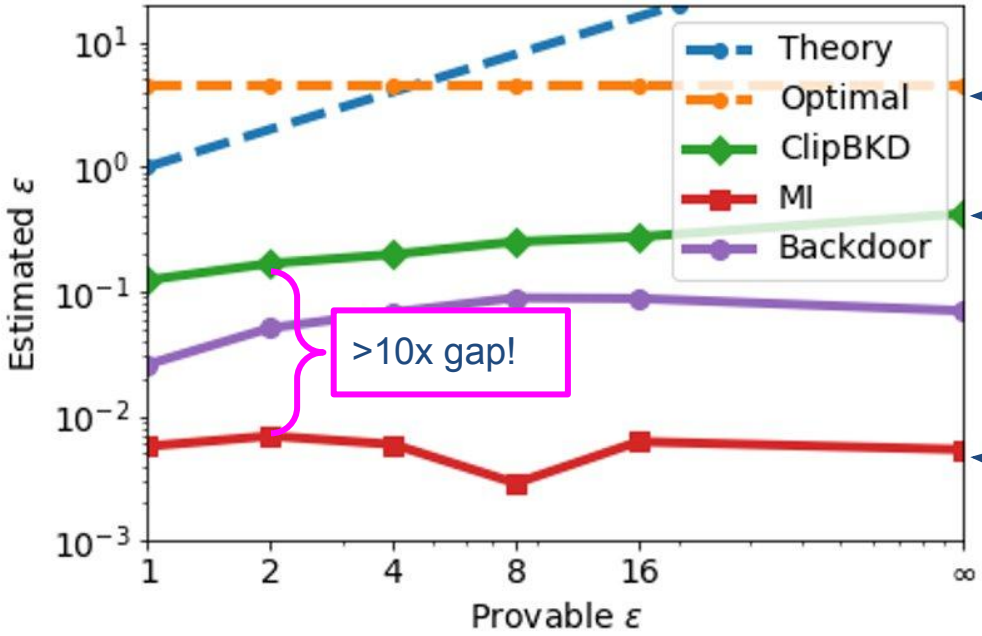
Best Possible Attack (blue and orange)

Our Technique

Membership Inference [YGFJ17]

[YGFJ17] - https://arxiv.org/abs/1709.01604

The Gap Tightens!



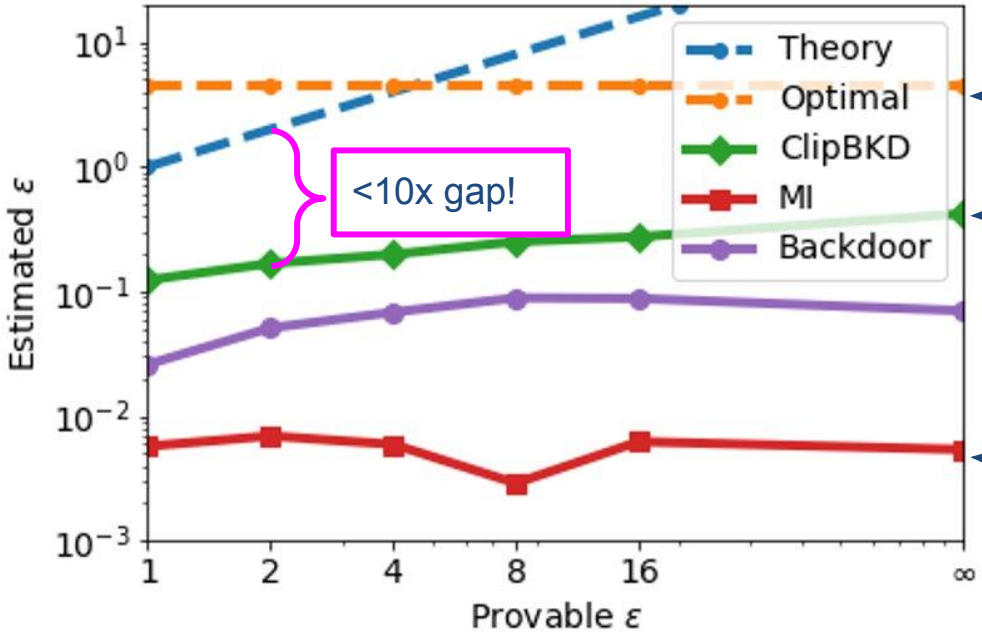
Best Possible Attack (blue and orange)

Our Technique

Membership Inference [YGFJ17]

[YGFJ17] - https://arxiv.org/abs/1709.01604

The Gap Tightens!



Best Possible Attack (blue and orange)

Our Technique

Membership Inference [YGFJ17]

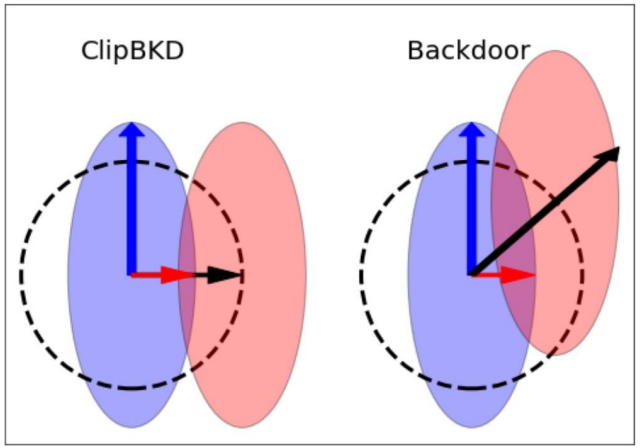
[YGFJ17] - https://arxiv.org/abs/1709.01604

Clipping-Aware Backdoor

- Gradient of model parameters with respect to a poisoning point is

$$\nabla_w \ell(w, b; x_p, y_p) = \ell'(w \cdot x_p + b, y_p) x_p.$$

- For Backdoor attacks:
  - Large $|\ell'(w \cdot x_p + b, y_p)|$
  - Relationship gets broken after clipping



- Clipping-Aware Backdoor:
  - Choose x_p such that it minimizes $Var_{(x,y)\in D}[\ell'(w \cdot x_p + b, y_p) x_p \cdot \ell'(w \cdot x + b, y) x] \leq Var_{(x,y)\in D}[x_p \cdot x].$

You train *how many* models?

Privacy Auditing with One (1) Training Run

https://arxiv.org/abs/2305.08846

Thomas Steinke, Milad Nasr, Matthew Jagielski

Google

This Work: *Can we do privacy auditing using a single run?*

- Idea: Consider multiple examples instead of a single example.
  - Group privacy?
  - Randomly include/exclude examples independently to avoid cost of group privacy.
  - Exploit connection between DP and generalization [Dwork, Feldman, Hardt, Pitassi, Reingold, Roth 2015].
- Flip m coins to decide which examples to include/exclude.
- Attempt to "guess" the coins from the output.
  - Intuitively ε-DP implies guess accuracy $\leq e^{\varepsilon}/(e^{\varepsilon}+1)$.

Algorithm 1 Auditor with One Training Run

1: **Data:** $x \in \mathcal{X}^n$ consisting of m auditing examples (a.k.a. canaries) $x_1, \cdots, x_m$ and $n-m$ non-auditing examples $x_{m+1}, \cdots, x_n$.
2: **Parameters:** Algorithm to audit $\mathcal{A}$, number of examples to randomize m, number of positive k_+ and negative k_- guesses.
3: For $i \in [m]$ sample $S_i \in \{-1, +1\}$ uniformly and independently. Set $S_i = 1$ for all $i \in [n] \setminus [m]$.
4: Partition x into $x_{\text{IN}} \in \mathcal{X}^{n_{\text{IN}}}$ and $x_{\text{OUT}} \in \mathcal{X}^{n_{\text{OUT}}}$ according to S, where $n_{\text{IN}} + n_{\text{OUT}} = n$. Namely, if $S_i = 1$, then x_i is in x_{IN}; and, if $S_i = -1$, then x_i is in x_{OUT}.
5: Run $\mathcal{A}$ on input x_{IN} with appropriate parameters, outputting w.
6: Compute the vector of scores $Y = (\textsc{Score}(x_i, w) : i \in [m]) \in \mathbb{R}^m$.
7: Sort the scores Y. Let $T \in \{-1, 0, +1\}^m$ be $+1$ for the largest k_+ scores and -1 for the smallest k_- scores. (I.e., $T \in \{-1, 0, +1\}^m$ maximizes $\sum_i^m T_i \cdot Y_i$ subject to $\sum_i^m |T_i| = k_+ + k_-$ and $\sum_i^m T_i = k_+ - k_-$.)
8: **Return:** $S \in \{-1, +1\}^m$ indicating the true selection and the guesses $T \in \{-1, 0, +1\}^m$.

This Work: *Can we do privacy auditing using a single run?*

ℙ[number of correct guesses ≥ v]

ℙ[ε-DP randomized response ≥ v]

+O(□)

Theore 1 (Main Result). *Let* $\in \{-1,+\}^m \times \{-1,0,+$ ^{m} *be the output of Alg m 1. Assu e the algorithm audit $\mathcal{A}$ satisfies (ε,δ)-D et $r := k_+ + k_- = \|T\|_1$ be the number of guesses. Then, r all $v \in \mathbb{R}$,*

$$\mathbb{P}\left[\sum_i^m \max\{0, T_i \cdot S_i\} \geq v\right] \leq \mathbb{P}\left[\check{W} \geq v\right] + \delta \cdot m \cdot \alpha,$$

where $\check{W} \leftarrow \mathsf{Binomial}\left(r, \frac{e^\varepsilon}{e^\varepsilon+1}\right)$ *and*

$$\alpha = \max_{i \in [m]} \frac{2}{i} \cdot \mathbb{P}\left[v > \check{W} \geq v - i\right].$$

ℙ[*A* is ε-DP randomized response]=1-2m□/i
ℙ[*A* adds i correct guesses]=2m□/i

Algorithm 1 Auditor with One Training Run

1: **Data:** $x \in \mathcal{X}^n$ consisting of m auditing examples (a.k.a. canaries) $x_1, \cdots, x_m$ and $n-m$ non-auditing examples $x_{m+1}, \cdots, x_n$.
2: **Parameters:** Algorithm to audit $\mathcal{A}$, number of examples to randomize m, number of positive k_+ and negative k_- guesses.
3: For $i \in [m]$ sample $S_i \in \{-1,+1\}$ uniformly and independently. Set $S_i = 1$ for all $i \in [n] \setminus [m]$.
4: Partition x into $x_{\text{IN}} \in \mathcal{X}^{n_{\text{IN}}}$ and $x_{\text{OUT}} \in \mathcal{X}^{n_{\text{OUT}}}$ according to S, where $n_{\text{IN}} + n_{\text{OUT}} = n$. Namely, if $S_i = 1$, then x_i is in x_{IN}; and, if $S_i = -1$, then x_i is in x_{OUT}.
5: Run $\mathcal{A}$ on input x_{IN} with appropriate parameters, outputting w.
6: Compute the vector of scores $Y = (\textsc{Score}(x_i, w) : i \in [m]) \in \mathbb{R}^m$.
7: Sort the scores Y. Let $T \in \{-1,0,+1\}^m$ be $+1$ for the largest k_+ scores and -1 for the smallest k_- scores. (I.e., $T \in \{-1,0,+1\}^m$ maximizes $\sum_i^m T_i \cdot Y_i$ subject to $\sum_i^m |T_i| = k_+ + k_-$ and $\sum_i^m T_i = k_+ - k_-$.)
8: **Return:** $S \in \{-1,+1\}^m$ indicating the true selection and the guesses $T \in \{-1,0,+1\}^m$.

Obtaining ε Lower Bounds

Theorem 3.1 (Main Result). *Let* $(S, T) \in \{-1, +1\}^m \times \{-1, 0, +1\}^m$ *be the output of Algorithm 1. Assume the algorithm to audit* $\mathcal{A}$ *satisfies* (ε, δ)*-DP. Let* $r := k_+ + k_- = \|T\|_1$ *be the number of guesses. Then, for all* $v \in \mathbb{R}$,

$$\mathbb{P}\left[\sum_i^m \max\{0, T_i \cdot S_i\} \geq v\right] \leq \mathbb{P}\left[\check{W} \geq v\right] + \delta \cdot m \cdot \alpha,$$

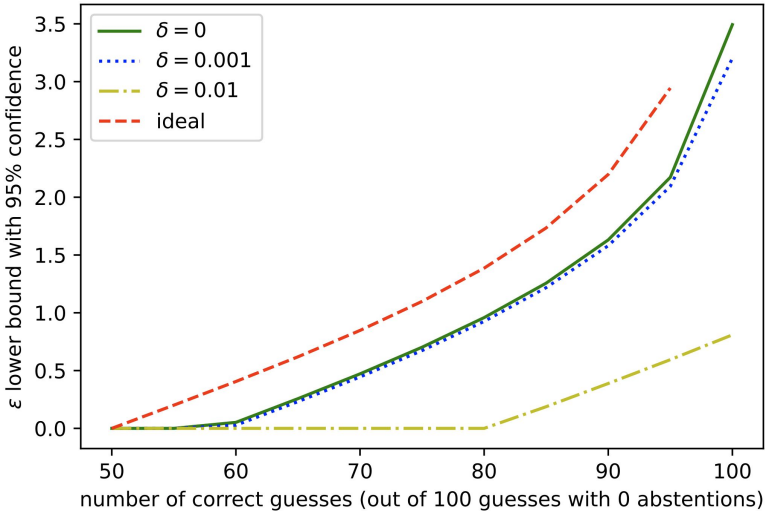
where $\check{W} \leftarrow \text{Binomial}\left(r, \frac{e^\varepsilon}{e^\varepsilon + 1}\right)$ *and*

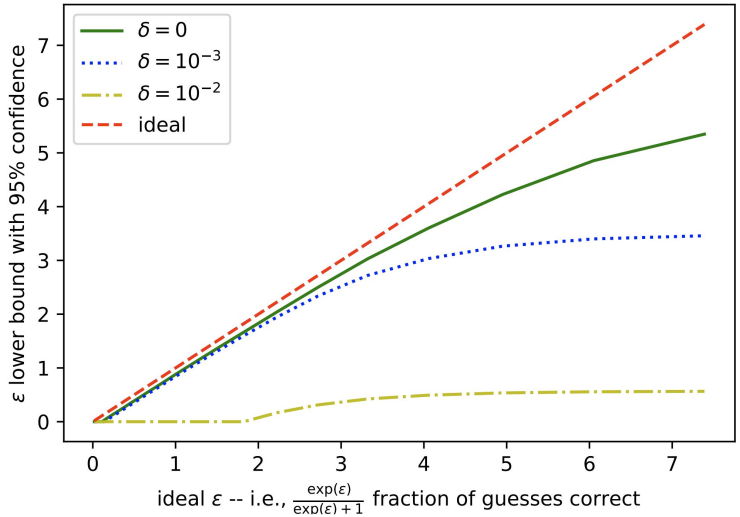
"p-value"

$$\alpha = \max_{i \in [m]} \frac{2}{i} \cdot \mathbb{P}\left[v > \check{W} \geq v - i\right].$$

Hypothesis test: If too many correct guesses, can reject hypothesis that A is (ε,δ)-DP.

Confidence interval = lower bound on ε:
Pick largest ε we can reject at given confidence.





Experimental Results

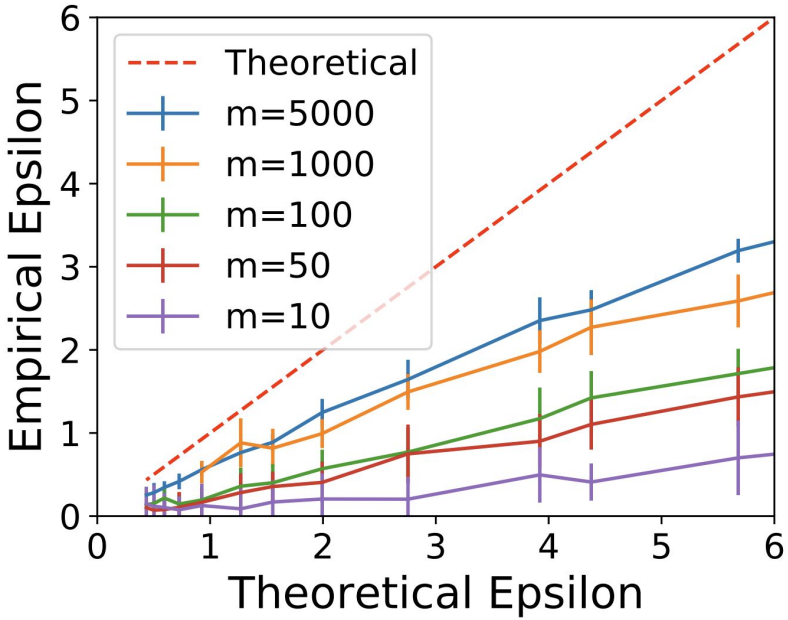


Figure 3. Effect of the number of auditing examples (m) in the white-box setting. By increasing the number of the auditing examples we are able to achieve tighter empirical lower bounds.

Adversary sees intermediate model weights (à la federated learning)

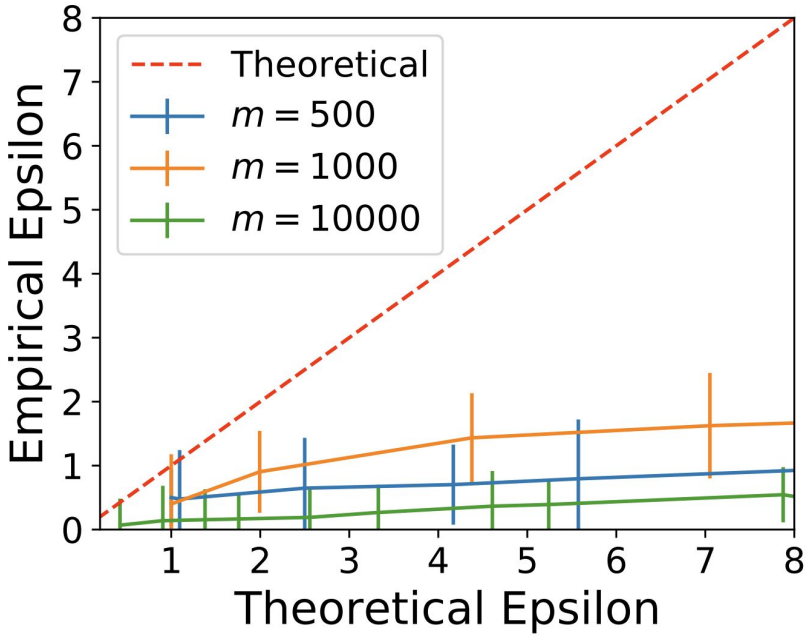


Figure 6. Effect of the number of auditing examples (m) in the black-box setting. Black-box auditing is very sensitive to the number of auditing examples.

Adversary only sees final model weights (or can only query the loss)

Conclusion

- Our Contribution: Privacy auditing with one run instead of 100s.
- Win: Less computation.
- Limitations: Weaker lower bounds.
  - Seems inherent with one run.
- Future Work:
  - Better attacks ⇒ stronger lower bounds.
  - A few runs? (>1 but <100s).