Overview

➔ **Why Privacy Matters**

Threat models for non-private ML

➔ **ML+DP**

Recap: DP setting, basic mechanisms

Private logistic regression

Private neural net weights with DPSGD

Learning Objectives

We want you to understand the following:

- DP+ML: Why? (security motivations)
  - Exploiting language models
  - Model inversion
- Basic tradeoffs and design choices at play
- DP+ML: How? (applying basic mechanisms)
- The Gaussian Mechanism and its applications:
  - Stochastic Gradient Descent
- Composition: how to analyze a sequence of mechanisms
- Private neural nets:
  - Implementation difficulties
  - Recent advances

Part 1: Security concerns

Slides in this section due to David Madras

Making Privacy Concrete

What is a Threat Model?

- Wikipedia: "**Threat modeling** is a process by which potential threats can be identified … all from a **hypothetical attacker's point of view**."

- Best way to talk about the **security** of our model is to specify a threat model – how might we be **vulnerable**?

- In this talk, I'll try to convince you **privacy is a real *security* threat** by presenting a concrete threat model

Let's Talk About *Model Inversion!*

- A trained ML model with parameters **w** is released to the public
  - **W** = training_procedure(X)
  - Training data X is hidden

- Can we *recover* some of X just through access to **w**?
  - X' = training_procedure^{-1}(**w**) <--- notational abuse
  - That would be bad

- Intersection of security and privacy

What Model Inversion Looks Like

Figure 1: An image recovered using a new model inversion attack (left) and a training set image of the victim (right). The attacker is given only the person's name and access to a facial recognition system that returns a class confidence score.

Two Examples We'll Discuss

- "The Secret Sharer: Measuring Unintended Neural Network Memorization & Extracting Secrets", Carlini et al., 2018

- "Model Inversion Attacks that Exploit Confidence Information and Basic Countermeasures", Fredrikson et al., 2015

Example 1. The Secret Sharer (Carlini et al.)

- Step 1: Find some training text containing sensitive information (e.g. credit card numbers)
  - "My credit card number is 3141-9265-3587-4001" $\in$ Training text
- Step 2: Train your language model without really thinking too hard
  - State-of-the-art log-likelihood!
- Step 3: Profit … for hackers
  - Sounds bad

Extracting Secrets

- “My credit card number is X” ∈ Training text
- Hacker is given black-box access to the model
- Prompt the model with ‘My credit card number is’ and generate!
  - **$$$$$$$$$$**

- Note: This isn’t exactly how they do it in the paper
  - Many annoying details in implementation

This Attack Kind of Works

		Number of Unique Phrases				
		1	10	50	100	500
# Insertions	1	80%	11%	2%	1%	0.1%
	2	100%	38%	18%	16%	1%
	5	100%	100%	100%	100%	98%
	10	100%	100%	100%	100%	100%

Table 4: Expected percentage of phrases that are uniquely extractable. Each inserted secret has the same format.

This Attack Kind of Works (Part II)

User	Secret Type	Exposure	Extracted?
A	CCN	52	✓
B	SSN	13	
C	SSN	16	
	SSN	10	
	SSN	22	
D	SSN	32	✓
F	SSN	13	
G	CCN	36	
	CCN	29	
	CCN	48	✓

Table 5: Summary of results on the Enron email dataset. Three secrets are extractable in under an hour; all are heavily memorized.

How to Defend?

- Maybe regularization?
  - Memorization relates to generalization
  - Authors try weight decay, dropout, and quantization – none work
  - The problem seems distinct from overfitting

- Maybe sanitization?
  - This makes sense: if you know what the secret looks like, just remove it before training
  - But you may not know all possible secret formats – this is heuristic

How to Defend?

- Differential Privacy!
  - Each token in the training text = “a record in the database”

	Optimizer	ε	Testing Loss	Estimated Exposure
With DP	RMSProp	0.65	1.69	1.1
	RMSProp	1.21	1.59	2.3
	RMSProp	5.26	1.41	1.8
	RMSProp	89	1.34	2.1
	RMSProp	2×10^8	1.32	3.2
	RMSProp	1×10^9	1.26	2.8
	SGD	∞	2.11	3.6
No DP	SGD	N/A	1.86	9.5
	RMSProp	N/A	1.17	31.0

Example 2: Targeted Model Inversion in Classifiers (Fredrikson et al.)

- Step 1. Train classifier parameters **w** on some secret dataset X
- Step 2. Release **w** to the public (white-box)
- Step 3. A hacker can recover parts of your training set by targeting specific individuals
  - That would be bad

Attacking a CNN

- Target: specific output (e.g. person's identity, sensitive feature)
- Start with some random input vector
- Use gradient descent in *input space* to maximize model's confidence in the target prediction

Attacking a Decision Tree using Auxiliary Information

- Given a trained decision tree where we know person X was in the the training set
- Assume we know $x_2 \ldots x_d$ for X, and want to find the value of x_1 (sensitive)

The following estimator characterizes the probability that $\mathbf{x}_1 = v$ given that $\mathbf{x}$ traverses one of the paths $s_1, \ldots, s_m$ and $\mathbf{x}_K = \mathbf{v}_K$:

$$\begin{aligned}
&\Pr\left[\, \mathbf{x}_1 = v \mid (s_1 \vee \cdots \vee s_m) \wedge \mathbf{x}_K = \mathbf{v}_K \,\right] \\
&\quad \propto \sum_{i=1}^{m} \frac{p_i \phi_i(v) \cdot \Pr\left[\, \mathbf{x}_K = \mathbf{v}_K \,\right] \cdot \Pr\left[\, \mathbf{x}_1 = v \,\right]}{\sum_{j=1}^{m} p_j \phi_j(v)} \\
&\quad \propto \frac{1}{\sum_{j=1}^{m} p_j \phi_j(v)} \sum_{1 \le i \le m} p_i \phi_i(v) \cdot \Pr\left[\, \mathbf{x}_1 = v \,\right] \qquad (1)
\end{aligned}$$

Decision Tree Experiments

- Trying to uncover the values of sensitive answers like:

risk-taking behaviors [17]. To support the analysis, FiveThirtyEight commissioned a survey of 553 individuals from SurveyMonkey, which collected responses to questions such as: "Do you ever smoke cigarettes?", "Have you ever cheated on your significant other?", and of course, "How do you like your steak prepared?". Demographic characteristics such as

and 11 variables, including basic demographic information and responses to questions such as, "How happy are you in your marriage?" and "Have you watched X-rated movies in the last year?" We discarded rows that did not contain re-

Decision Tree Results

	FiveThirtyEight			GSS		
algorithm	**acc.**	**prec.**	**rec.**	**acc.**	**prec.**	**rec.**
whitebox	86.4	100.0	21.1	80.3	100.0	0.7
blackbox	85.8	85.7	21.1	80.0	38.8	1.0
random	50.0	50.0	50.0	50.0	50.0	50.0
baseline	82.9	0.0	0.0	82.0	0.0	0.0
ideal	99.8	100.0	98.6	80.3	61.5	2.3

Figure 4: MI results for for BigML models. All numbers shown are percentages.

Defending Against Model Inversion

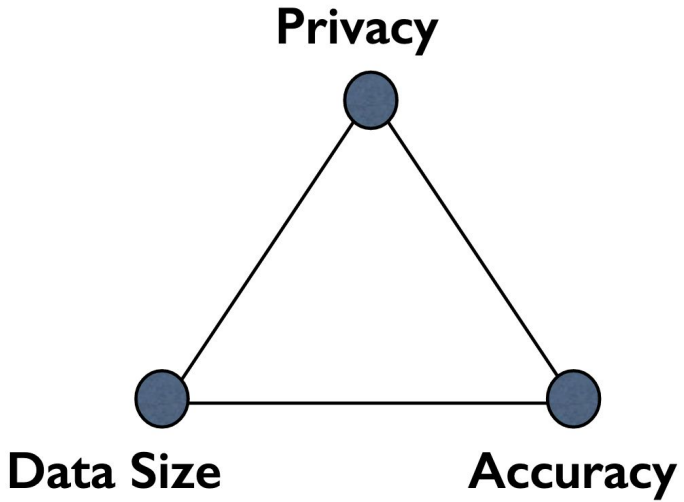
- Decision trees: split on sensitive features lower down
- CNNs: no concrete suggestions
  - But this paper came out before DP-SGD
- I think differential privacy would protect against both these attacks
- As always, consider tradeoffs with dataset size and accuracy

Conclusion

- If your software works, great! ☺
- If your software works but can be hacked –
  - Then your software doesn't work! ☹
- Hopefully, this presentation convinced you that privacy is a realistic security issue by providing a concrete threat model
- Not everyone needs to think about privacy all the time
  - But some people need to think about it some of the time
  - Or bad things will happen! ☺ ☺ ☺

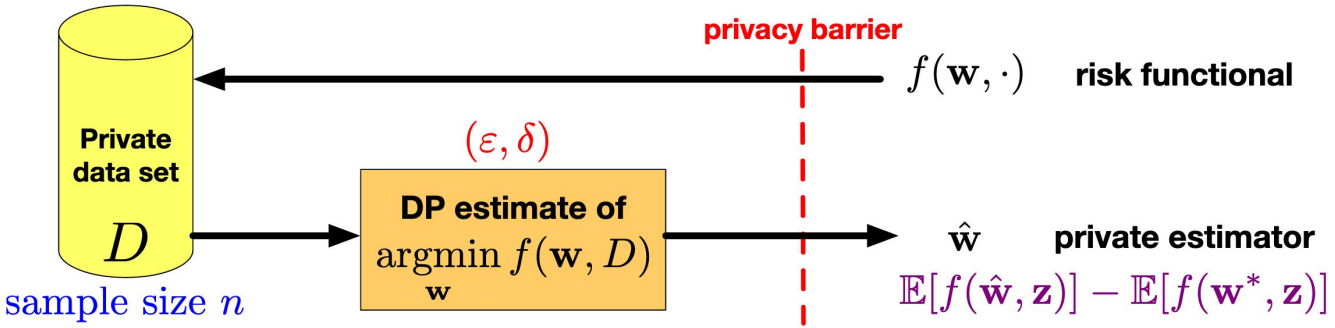
Part 2: Making ML Private

Tradeoffs in DP+ML



[Chaudhuri & Sarwate 2017 NIPS tutorial]

Tradeoffs in DP+ML



Statistical estimation: estimate a parameter or predictor using private data that has good expected performance on future data.

[Chaudhuri & Sarwate 2017 NIPS tutorial]

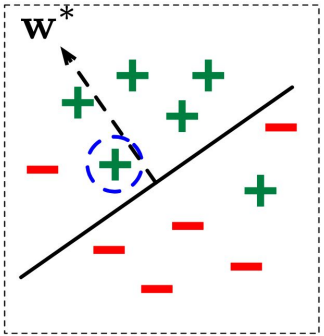
Goal: Good privacy-accuracy-sample size tradeoff

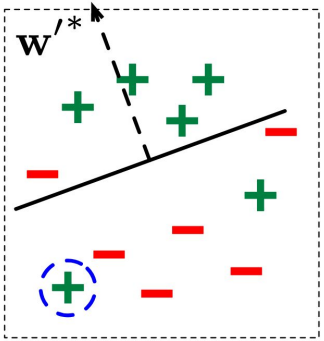
Private Empirical Risk Min.

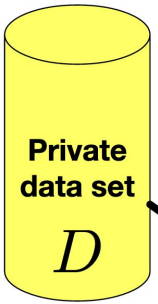
$$\mathbf{w}^* = \operatorname*{argmin}_{\mathbf{w}} \frac{1}{n} \sum_{i=1}^{n} \ell(\mathbf{w}, (\mathbf{x}_i, y_i)) + \lambda R(\mathbf{w})$$

- *Empirical Risk Minimization* (ERM) is a common paradigm for prediction problems.
- Produces a predictor **w** for a label/response *y* given a vector of features/covariates **x**.
- Typically use a convex loss function and regularizer to "prevent overfitting."

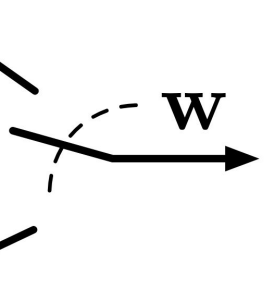
Private ERM







easy for adversary to tell the difference between D and D'



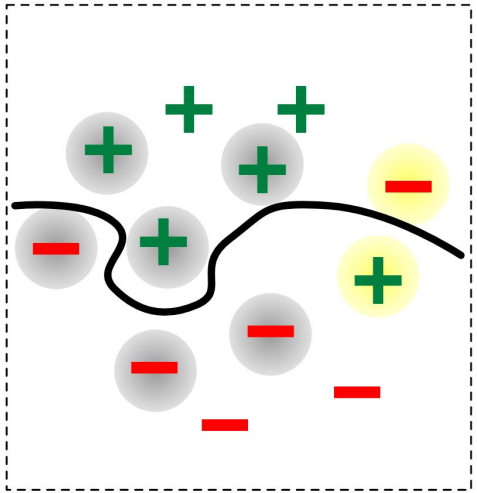
[CMS11, RBHT12]

[Chaudhuri & Sarwate 2017 NIPS tutorial]

Kernel Approaches Even Worse

- Kernel-based methods produce a classifier that is a function of the data points.
- Even adversary with black-box access to w could potentially learn those points.

$$\mathbf{w}(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i k(\mathbf{x}, \mathbf{x}_i)$$



[Chaudhuri & Sarwate 2017 NIPS tutorial]

[CMS11]

Privacy & Learning Compatible

- Good learning algorithms *generalize* to the population distribution, not individuals.
- *Stable learning algorithms* generalize [BE02].
- Differential privacy can be interpreted as a form of stability that also implies generalization [DFH+15,BNS+16].
- Two parts of the same story:
  Privacy implies generalization asymptotically.
  Tradeoffs between privacy-accuracy-sample size for finite *n*.

[Chaudhuri & Sarwate 2017 NIPS tutorial]

Revisiting ERM

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \ell(\mathbf{w}, (\mathbf{x}_i, y_i)) + \lambda R(\mathbf{w})$$

- Learning using (convex) optimization uses three steps:

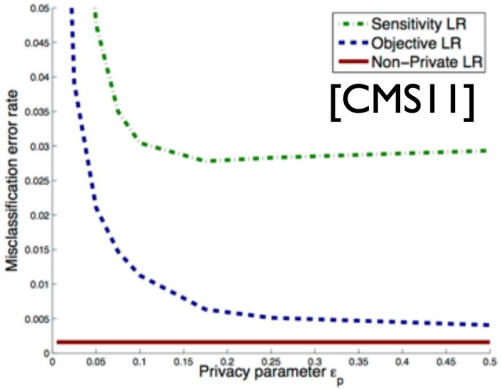
1. read in the data — **input perturbation**
2. form the objective function — **objective perturbation**
3. perform the minimization — **output perturbation**

We will discuss this one -->

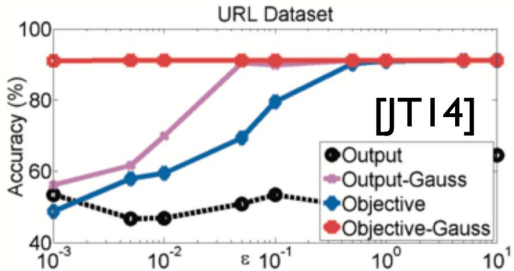
- We can try to introduce privacy in each step!

[Chaudhuri & Sarwate 2017 NIPS tutorial]

Typical Empirical Results



(a) Regularized logistic regression, KDDCup99

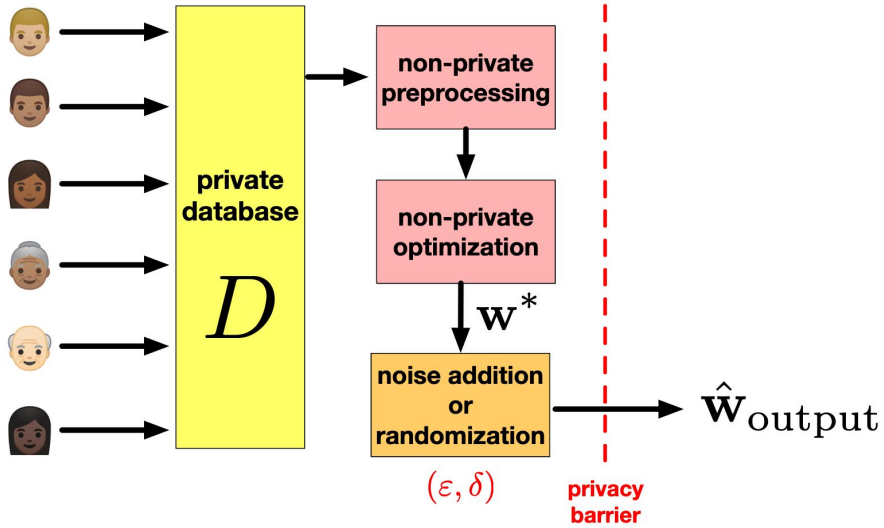


In general:

- **Objective** perturbation empirically outperforms output perturbation.
- **Gaussian** mechanism with (ε, δ) guarantees outperform **Laplace** - like mechanisms with ε-guarantees.
- **Loss vs. non-private methods** is very dataset-dependent.

[Chaudhuri & Sarwate 2017 NIPS tutorial]

Output Perturbation



- Compute the minimizer and add noise.
- Does not require re-engineering baseline algorithms

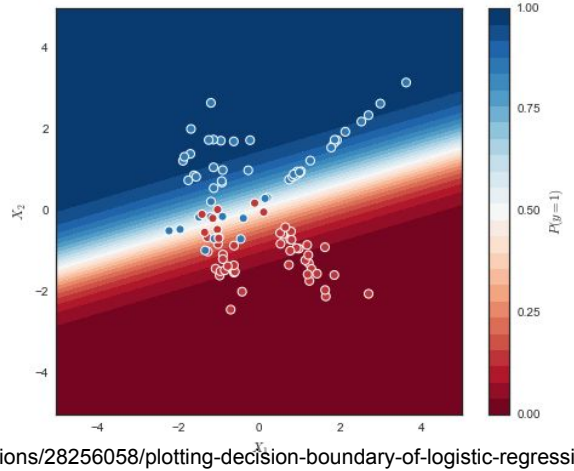
Noise depends on the sensitivity of the argmin.

[CMS11, RBHT12]

Private Logistic Regression

$$L(w) = \sum_{i=1}^{N} \text{CrossEntropy}(\sigma(w^T x_i), y_i) + \frac{1}{2}\lambda w^T w$$

What is $\Delta L(w)$, the sensitivity of the loss?



https://stackoverflow.com/questions/28256058/plotting-decision-boundary-of-logistic-regression

Private Logistic Regression

$$L(w) = \sum_{i=1}^{N} \underbrace{\text{CrossEntropy}(\sigma(w^T x_i), y_i)}_{\text{convex}} + \underbrace{\frac{1}{2}\lambda w^T w}_{\lambda-\text{strongly convex}}$$

$h(w)$ is convex iff $\nabla^2 h(w) \succcurlyeq 0$ (its Hessian is positive semi-definite)

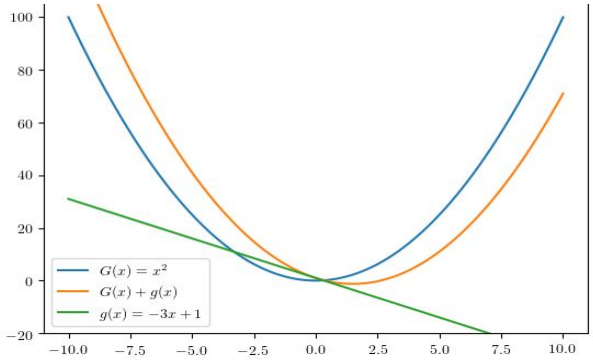
$h(w)$ is $\lambda-$strongly convex iff $(\nabla h(x) - \nabla h(y))^T (x - y) \geq \lambda ||x - y||_2^2$

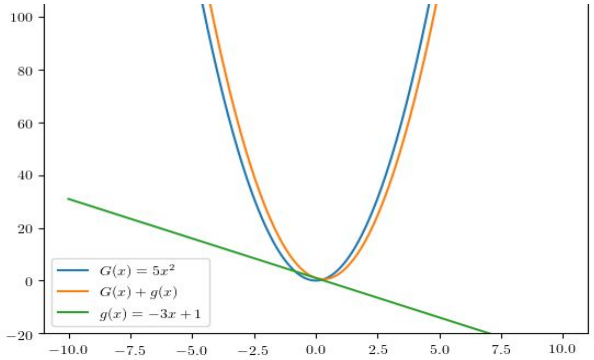
The sum of a convex function and a λ-strongly convex function is λ-strongly convex.

Private Logistic Regression

Lemma 7 *Let $G(\mathbf{f})$ and $g(\mathbf{f})$ be two vector-valued functions, which are continuous, and differentiable at all points. Moreover, let $G(\mathbf{f})$ and $G(\mathbf{f}) + g(\mathbf{f})$ be λ-strongly convex. If $\mathbf{f}_1 = \operatorname{argmin}_{\mathbf{f}} G(\mathbf{f})$ and $\mathbf{f}_2 = \operatorname{argmin}_{\mathbf{f}} G(\mathbf{f}) + g(\mathbf{f})$, then*

$$\|\mathbf{f}_1 - \mathbf{f}_2\| \leq \frac{1}{\lambda} \max_{\mathbf{f}} \|\nabla g(\mathbf{f})\| .$$





Private Logistic Regression

$h(w)$ is $\lambda-$strongly convex iff $(\nabla h(x) - \nabla h(y))^T (x - y) \geq \lambda ||x - y||_2^2$

The sum of a convex function and a λ-strongly convex function is λ-strongly convex.

$$L(w) = \frac{1}{N} \sum_{i \in \{1..N\}} \text{CrossEntropy}(\sigma(w^T x_i), y_i) + \frac{1}{2} \lambda w^T w$$

$$L'(w) = \left(\frac{1}{N} \sum_{i \in \{1..N\} \setminus j} \text{CrossEntropy}(\sigma(w^T x_i), y_i) \right) + \text{CrossEntropy}(\sigma(w^T x_{j'}), y_{j'}) + \frac{1}{2} \lambda w^T w$$

$$L'(w) = L(w) + g(w)$$

$$g(w) = \frac{1}{N} \left(\text{CrossEntropy}(\sigma(w^T x_{j'}), y_{j'}) - \text{CrossEntropy}(\sigma(w^T x_j), y_j) \right)$$

Assuming $||x|| \leq 1$, we have $\nabla g(w) \leq \frac{2}{N}$

$$\rightarrow \Delta L = \frac{2}{N\lambda}$$

Private Logistic Regression

Algorithm:

sensitivity $\Delta L = \frac{2}{N}$

1) solve $w^* = \arg\min_w L(w)$

2) draw η with $p(\eta) \propto \exp(-\frac{\Delta L}{\epsilon}||\eta||)$ (vector analogue of Laplace draw)

3) return $w = w^* + \eta$

Private Non-Convex Learning

For non-convex losses:

- We don't know whether our optimizer will (asymptotically) find the global optimum
- We can not bound the loss function at the optimum or anywhere else

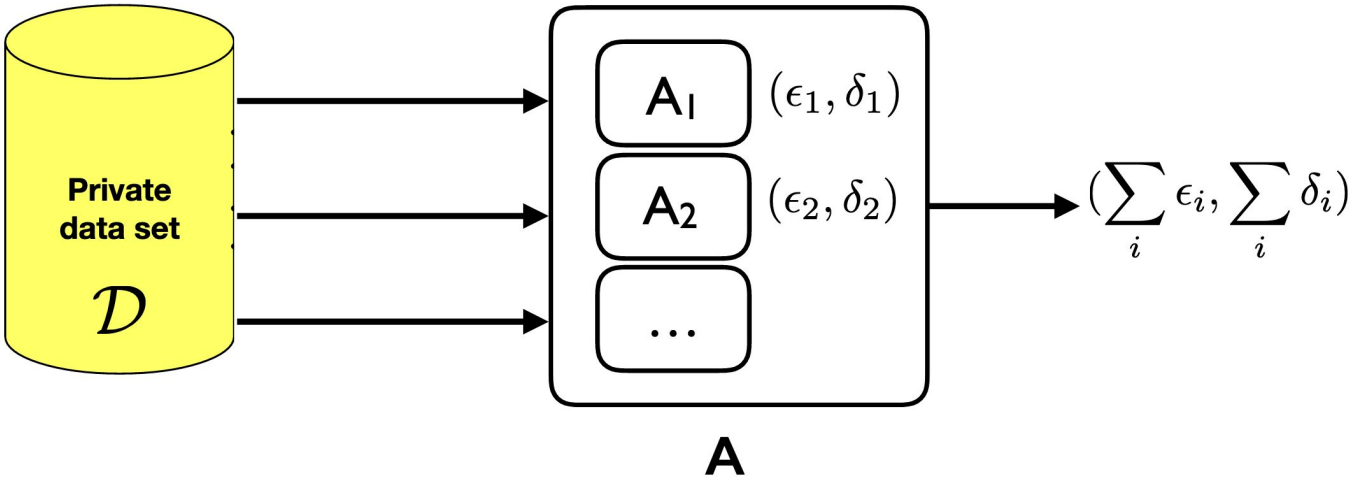
Instead we will make every step of the iterative optimization algorithm private, and somehow account for the overall privacy loss at the end

Private Non-Convex Learning

To discuss DPSGD algorithm, we need to understand

- Basic composition of mechanisms
- “Advanced” composition in an adaptive setup
- Privacy amplification by subsampling
- Per-example gradient computation

Recall Composition Theorem



Total privacy loss is the sum of privacy losses

(Better composition possible — coming up later)

Advanced Composition

What is composition?

- Repeated use of DP algorithms on the same database
- Repeated use of DP algorithms on the different databases that nevertheless may contain shared information about individuals

We can show that the privacy loss over all possible outcomes has a Markov structure, which hints at a better composition

Advanced Composition

Theorem 3.20 (Advanced Composition)**.** For all $\varepsilon, \delta, \delta' \geq 0$, the class of (ε, δ)-differentially private mechanisms satisfies $(\varepsilon', k\delta + \delta')$-differential privacy under k-fold adaptive composition for:

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

Privacy by Subsampling

Lemma 3 (Amplification via sampling) If A is 1 –differentially private, then for any $\epsilon \in (0,1)$, $A'(\epsilon, \cdot)$ is 2ϵ –differentially private.

Suppose A is a 1 –differentially private algorithm that expects data sets from a domain D as input. Consider a new algorithm A', which runs A on a random subsample of $\approx \epsilon n$ points from its input:

Proof: Fix an event S in the output space of A', and two data sets x, x' that differ by a single individual, say $x = x' \cup \{i\}$.

Consider a run of A' on input x. If i is not included in the sample T, then the output is distributed the same as a run of A' on $x' = x \setminus \{i\}$, since the inclusion of i in the sample is independent of the inclusion of other elements. On the other hand, if i is included in the sample T, then the behavior of A on T is only a factor of e off from the behavior of A on $T \setminus \{i\}$. Again, because of independence, the distribution of $T \setminus \{i\}$ is the same as the distribution of T conditioned on the omission of i. For a set $T \subseteq D$, let p_T denote the distribution of $A(T)$. In symbols, we have that for any event S:

$$p_x(S \mid i \notin T) = p_{x'}(S) \quad \text{and} \quad p_x(S \mid i \in T) \in e^{\pm 1} p_{x'}(S).$$

We can put the pieces together, using the fact that i is in T with probability only ϵ:

$$\begin{aligned} p_x(S) &= (1-\epsilon) \cdot p_x(S \mid i \notin T) + \epsilon \cdot p_x(S \mid i \in T) \\ &\leq (1-\epsilon) \cdot p_{x'}(S) + \epsilon \cdot e \cdot p_{x'}(S) \\ &= (1 + \epsilon(e-1)) p_{x'}(S) \\ &\leq \exp(2\epsilon) \cdot p_{x'}(S) \end{aligned}$$

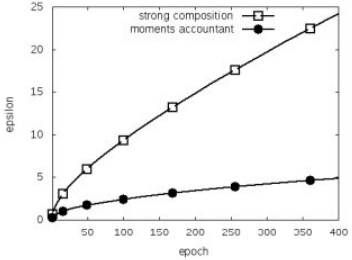
We can get a similar lower bound:

$$\begin{aligned} p_x(S) &= (1-\epsilon) \cdot p_x(S \mid i \notin T) + \epsilon \cdot p_x(S \mid i \in T) \\ &\geq (1-\epsilon) \cdot p_{x'}(S) + \epsilon \cdot \tfrac{1}{e} \cdot p_{x'}(S) \\ &= (1 - \epsilon(1 - e^{-1})) \cdot p_{x'}(S) \\ &\geq \exp(-\epsilon) \cdot p_{x'}(S) \end{aligned}$$

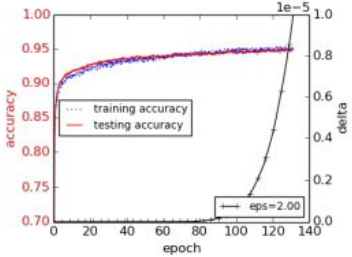
The last inequality uses the fact that $\epsilon \leq 1$. □

https://adamdsmith.wordpress.com/2009/09/02/sample-secrecy/

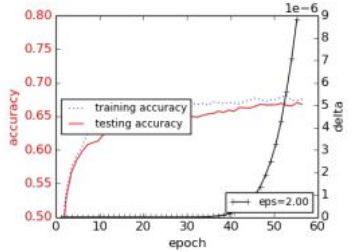
Moments accountant improves bounds



MNIST epoch vs accuracy/privacy



CIFAR-10 epoch vs accuracy/privacy



Differentially Private SGD

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.

[Abadi et al. 2016]

Guarantees final parameters don't depend too much on individual training examples

Gaussian noise added to the parameter update at every iteration

Privacy loss accumulates over time

The "moments accountant" provides better empirical bounds on (ε,δ)

Differentially Private SGD

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

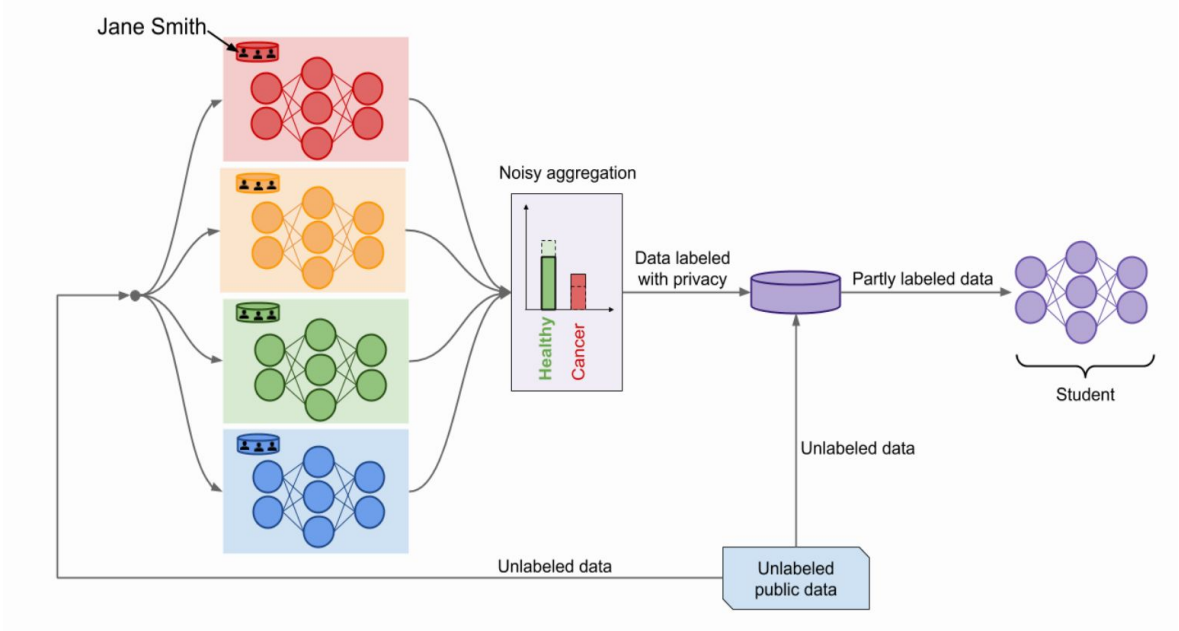
$\theta_{t+1} \leftarrow \theta_t - \eta_t \tilde{\mathbf{g}}_t$

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

← when can we efficiently compute per-example gradients?

[Abadi et al. 2016]

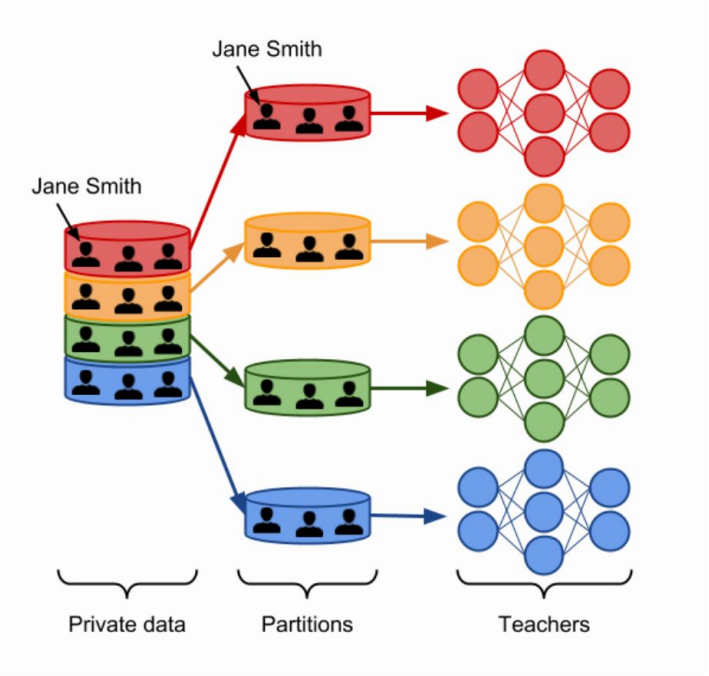
PATE



Private Aggregation of Teacher Ensembles [Papernot et al 2017, Papernot et al 2018]

Key idea: instead of adding noise to gradients, add noise to *labels*

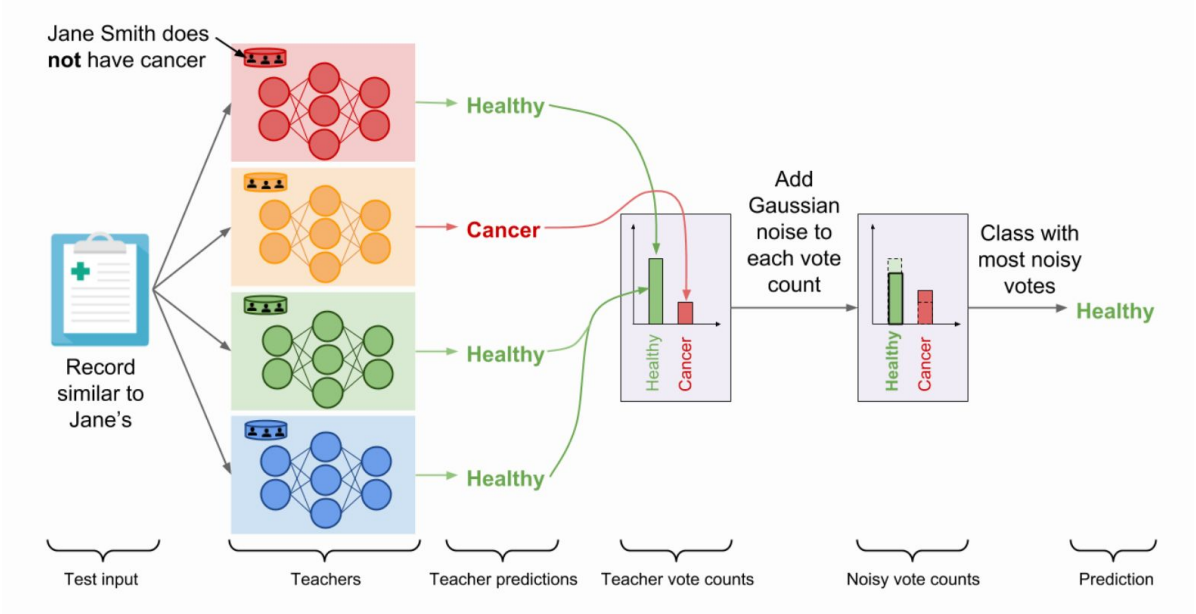
PATE



Start by partitioning private data into disjoint sets

Each teacher trains (non-privately) on its corresponding subset

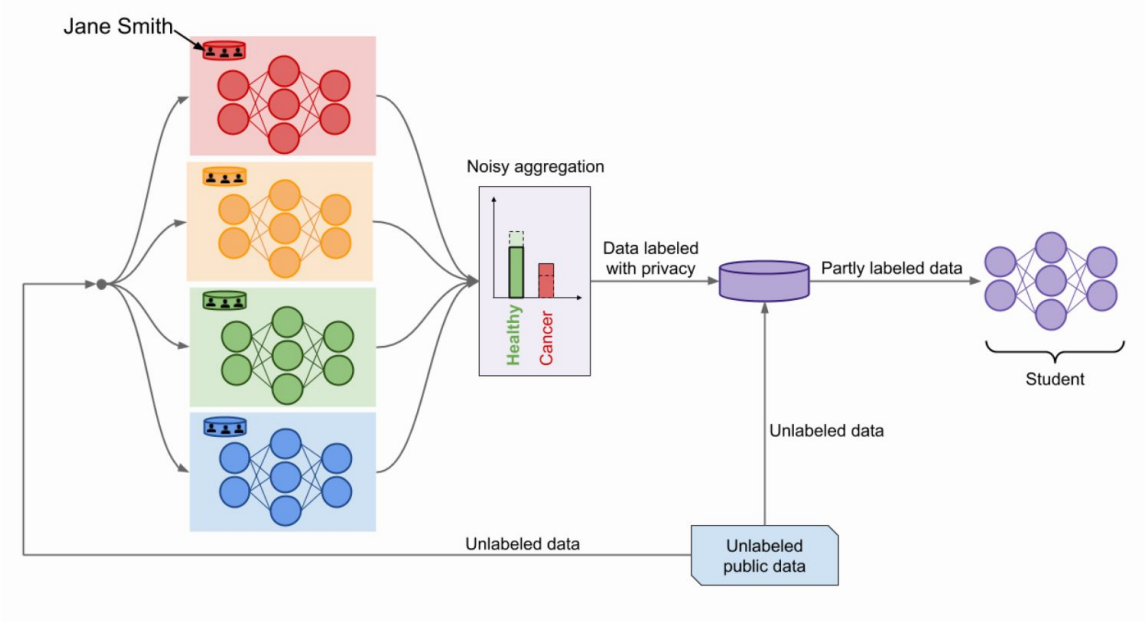
PATE



Private predictions can now be generated via the exponential mechanism, where the "score" is computed with an election amongst teachers - output the noisy winner

We now have private inference, but we lose privacy every time we predict. We would like the privacy loss to be constant at test time.

PATE

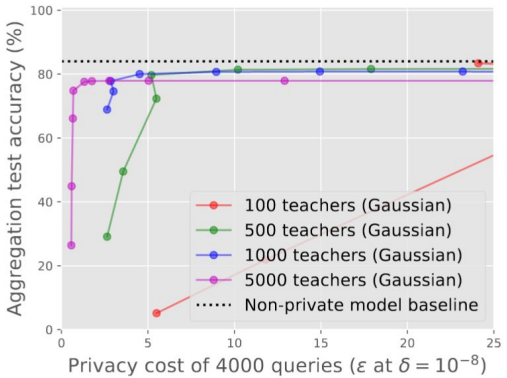


We can instead use the noisy labels provided by the teachers to train a student

We leak privacy during training but at test time we lose no further privacy (due to post-processing thm)

Because the student should use as few labels as possible, unlabeled public data is leveraged in a semi-supervised setup.

PATE



Dataset	Aggregator	Queries answered	Privacy bound ε	Accuracy Student	Accuracy Baseline
MNIST	LNMax (Papernot et al., 2017)	100	2.04	98.0%	99.2%
	LNMax (Papernot et al., 2017)	1,000	8.03	98.1%	
	Confident-GNMax (T=200, σ_1=150, σ_2=40)	286	**1.97**	**98.5%**	
SVHN	LNMax (Papernot et al., 2017)	500	5.04	82.7%	92.8%
	LNMax (Papernot et al., 2017)	1,000	8.19	90.7%	
	Confident-GNMax (T=300, σ_1=200, σ_2=40)	3,098	**4.96**	**91.6%**	
Adult	LNMax (Papernot et al., 2017)	500	2.66	83.0%	85.0%
	Confident-GNMax (T=300, σ_1=200, σ_2=40)	524	**1.90**	**83.7%**	
Glyph	LNMax	4,000	4.3	72.4%	82.2%
	Confident-GNMax (T=1000, σ_1=500, σ_2=100)	10,762	2.03	**75.5%**	
	Interactive-GNMax, two rounds	4,341	**0.837**	73.2%	

https://arxiv.org/pdf/1802.08908.pdf