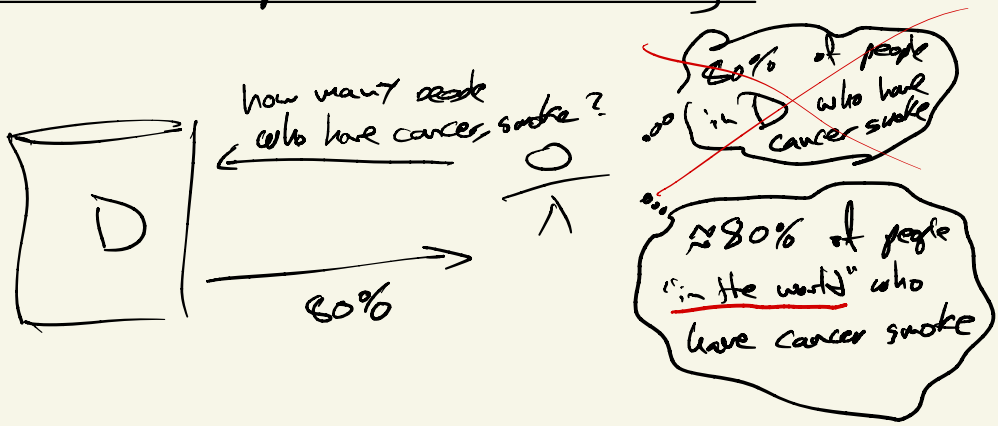
Differentially private learning



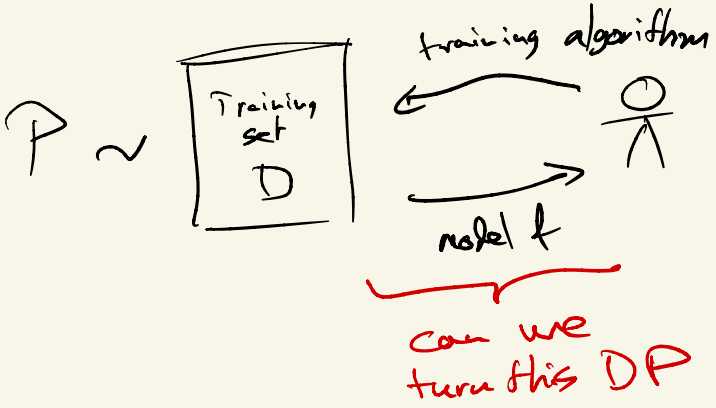
Generalization for simple stats

- D consists of n samples taken i.i.d from P
- we compute empirical mean $\bar{x} = \frac{1}{n}\sum x_i$
- in expectation, $|\bar{x} - \mu| = O(1/\sqrt{n})$
  - μ: $\mathbb{E}_{P}[x]$
  - $O(1/\sqrt{n})$: sampling error

Note:
- sampling error: $O(1/\sqrt{n})$
- noise added to $\bar{x}$ to get DP: $O(1/n)$

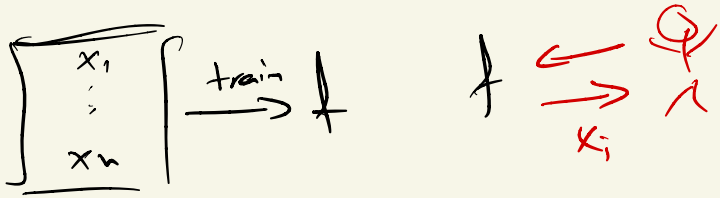
privacy comes for free ☺

Machine Learning



Why private learning?

- ML models can "overfit" to training data
- ML models can "memorize" some data



Can we prevent this?

Non-private learning

Setting: • distribution P over inputs X with labels Y

• Goal: learn a model $f_\theta : X \to Y$ ← parameters of the model

that "generalizes"

Defining generalization:

* loss function $\ell(\theta, x, y)$ measures the "error" of f_θ on input (x, y)

ex: $\ell(\theta, x, y) = \mathbb{1}\{f_\theta(x) = y\} \in \{0, 1\}$

* risk $L(\theta)$ is expected loss over P:

$$L(\theta) = \mathop{\mathbb{E}}_{(x,y)\sim P}[\ell(\theta, x, y)]$$

"test loss" / "test error"

We want a model with low risk!

* what do we get?

Training set $D = \{(x_1, y_1), \ldots (x_n, y_n)\}$

we have $(x_i, y_i) \overset{iid}{\sim} P$

empirical risk $\hat{L}_D(\theta) = \frac{1}{n}\sum_{i=1}^{n} \ell(\theta, x_i, y_i)$ "training loss" or error

Empirical risk minimization: (ERM)

- find θ^* that minimize the loss on the training set

ignore whether this is possible...

$$\theta^* = \underset{\theta \in \text{ALL MODELS}}{\operatorname{argmin}} \hat{L}_D(\theta)$$

ex: all 10-layer neural nets with 1M parameters and architecture...

- "Generalization gap": "overfitting"

$$|\text{test error} - \text{train error}|$$

$$= |L(\theta^*) - \hat{L}_D(\theta^*)|$$

Note: if we can fit the training set exactly ($\hat{L}_D(\theta^*) = 0$), then the generalization gap is just $L(\theta^*)$

- How to control the generalization gap?
  regularization, data augmentation, etc.

How to turn ERM private?

- Output perturbation — mostly of theoretical interest

  apply a standard training algorithm $T(D)$ to get a model θ, and then "add noise" to θ.

  ↳ not clear what this means? e.g. adding noise to θ

  ↳ we need to know the sensitivity of the ERM algorithm

  ↳ this is possible for "simple" models (linear/logistic regression)

- Objective perturbation — mostly theoretical interest

  instead of solving $\operatorname{argmin}_\theta \hat{L}_D(\theta)$

  solve $\operatorname{argmin}_\theta \hat{L}_D(\theta) + \underbrace{b}^T\theta$ (b: random)

  ↳ for some models, we can show this is DP!!

  ↳ we are solving ERM exactly, just with a different objective

- Gradient perturbation

  Specific training algorithm: gradient descent

Private Projected Gradient Descent

$\theta_0 \xleftarrow{\$} \Omega$ ← set of parameters

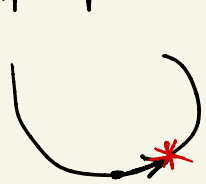
for $t = 0 \dots T-1$:

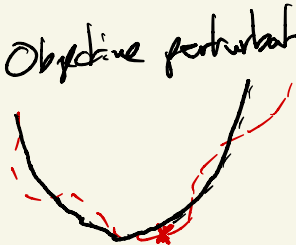
$g \leftarrow \nabla_{\theta_t} \hat{L}_D(\theta_t) = \frac{1}{n} \sum_{i=1}^{n} \nabla_{\theta_t} \ell(\theta_t, x_i, y_i)$ ← "full batch"

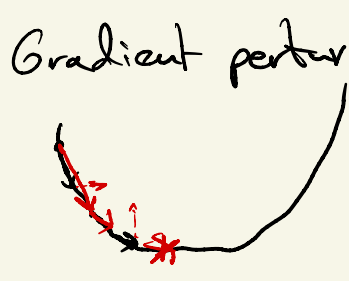
$\theta_{t+1} \leftarrow \theta_t - \eta \cdot (g + N(0, \sigma^2 \cdot I_{d \times d}))$ ← step size

($\theta_{t+1} \leftarrow$ Project θ_{t+1} onto Ω)

return θ_T







Utility: in convex setting, in worst-case the training error of private ERM is $O(\sqrt{d}/\varepsilon)$ (larger than for standard ERM)

(d: dimension of the model (# of learnable parameters))

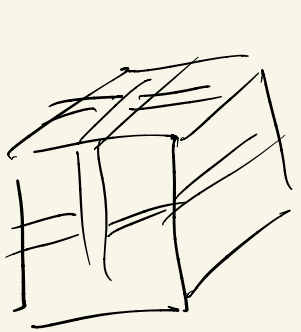
Private Deep learning

train models where $d \gg n$ (d: # of parameters, n: # of samples)

and no convexity

Theory: this won't generalize $\Rightarrow$ Wrong

DP Theory: this can't be learned with privacy $\Rightarrow$ Wrong



DP-SGD

"a miracle"

Crucial step: making the batch gradient private

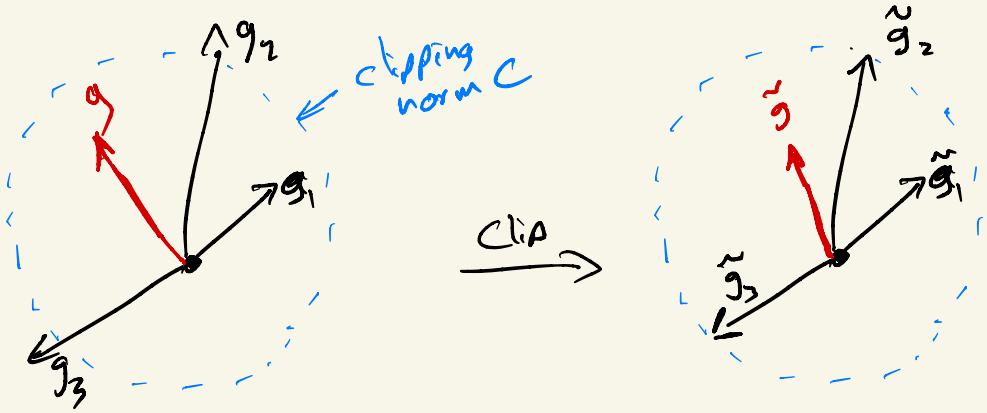
* Sample a batch of examples B
* calculate the mean gradient

$$g = \frac{1}{|B|} \sum_{i=1}^{|B|} \underbrace{\nabla_\theta L(\theta, x_i, y_i)}_{g_i}$$

$$= \frac{1}{m} \sum_{i=1}^{m} g_i$$

how large can this vector be (in ℓ_2 norm)?

* Clip the gradients to enforce a maximal ℓ_2 norm



Observations: 1) this biases the gradient

2) implementing this in practice (efficiently) is non-trivial

3) how to choose C?

- large: $\tilde{g}$ is less biased, but you will need to add more noise for DP
- small: $\tilde{g}$ is very biased, but less noise
- in practice: C is a hyperparameter

DP-SGD (Abadi et al. 2016)

$\theta_0 \xleftarrow{\$} \Omega$

for $t = 0 \dots T-1$ do

Sample a batch $S \subseteq [n]$ of size m

$g_i \leftarrow \nabla_{\theta_t} \ell(\theta_t, x_i, y_i) \quad \forall i \in S$

$g_i \leftarrow C \cdot g_i / \max(C, \|g_i\|_2)$

$\tilde{g} \leftarrow \frac{1}{m} \sum_{i \in S} g_i$ ← those all have ℓ_2 norm $\leq C$

$\theta_{t+1} \leftarrow \theta_t - \eta \left(\tilde{g} + N(0, \sigma^2 \cdot I_{d \times d}) \right)$

Analysis

1) "Naïve"

- the sensitivity of the gradient $\tilde{g}$ is $\frac{2C}{m}$
- we can add Gaussian noise

  $\sigma^2 = 2(2C)^2 \log(1.25/\delta) / m^2\varepsilon^2$ to get (ε, δ)-DP

  for <u>one</u> gradient step

✗ Basic composition: $(\varepsilon T, \delta T)$-DP

✗ Advanced composition: $(\varepsilon\sqrt{T\log(1/\delta)}, \delta T)$-DP

↳ a lot better, but still not enough!!

Amplification by subsampling

<u>Lemma</u>: let $\mathcal{M}$ be (ε, δ)-DP. Let $\mathcal{M}'(D)$ be the algo that subsamples each element of D w.p p to build $\tilde{D}$ and then outputs $\mathcal{M}(\tilde{D})$

↑ Then $\mathcal{M}'$ is (ε', δ')-DP for $\varepsilon' \approx p \cdot \varepsilon$

$\delta' = p \cdot \delta$

Let $p = \frac{m}{n}$ ← batch size
↖ # of training samples

Then DP-SGD is $O(\varepsilon p\sqrt{T\log(1/\delta)}, \delta pT)$
DP

Even better composition ?

Insight: • advanced composition applies to any (ε, δ)-DP algorithm
• can we get a better analysis for the subsampled Gaussian mechanism?
YES !

How? "Tedious" math (variants or generalizations of DP: Rényi-DP, CDP, Gaussian-DP)

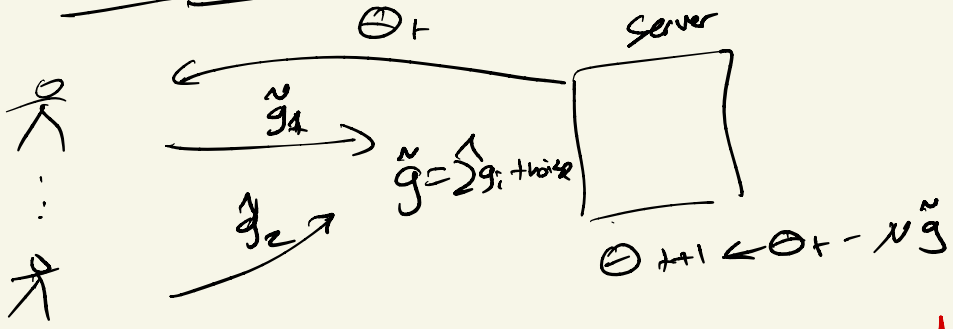
Result: $O(\varepsilon p\sqrt{T}, \delta)$
↑ save a $\log(1/\delta)$ term
↑ save a pT term

DP-SGD in practice

Local vs central mode of DP.

Central

"Local" or decentralized (Federated learning)



Q: · who computes the noisy sum? → MPC / → Trusted server / → multiple non-colluding servers

- how to do random subsampling?
  - ↳ what if clients aren't always online?
  - ↳ how to keep the subsample secret from the server(s)?