CSCI 699: Privacy Preserving Machine Learning - Week 3

Algorithms for Differentially Privacy and Machine Learning

Sai Praneeth Karimireddy, Sep 13 2024

Recap

- Differential privacy
  - $\forall y, \forall$ similar D, D' : $\dfrac{\Pr[Y = y \,|\, \mathscr{D} = D]}{\Pr[Y = y \,|\, \mathscr{D} = D']} \leq e^{\varepsilon}$
  - connection to tradeoff curves of attacker
  - If f is Δ_1-sensitive wrt ℓ_1 norm, Laplace mechanism
    - output ith coordinate = $f_i(D)$ + Lap(Δ_1/ε)

ε- DP $\Rightarrow$ $\mathcal{L}_{D,D'} \leq \varepsilon$ a.s.

Recap

- Approximate differential privacy

For $t \sim A(D)$ and $\mathscr{L}_{D,D'} = \ln\left(\frac{Pr[A(D) = t]}{Pr[A(D') = t]}\right)$,

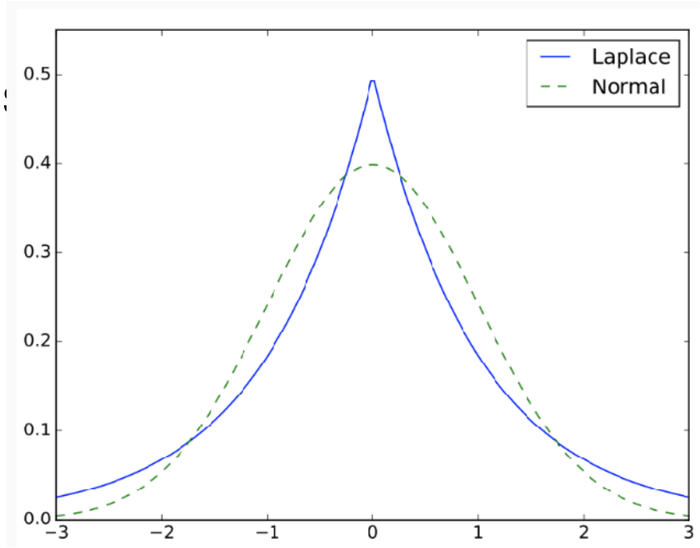
A satisfies (ε, δ)-DP iff for any neighboring datasets $D, D' \in \chi^n$

$$Pr\left[\mathscr{L}_{D,D'} \geq \varepsilon\right] \leq \delta$$

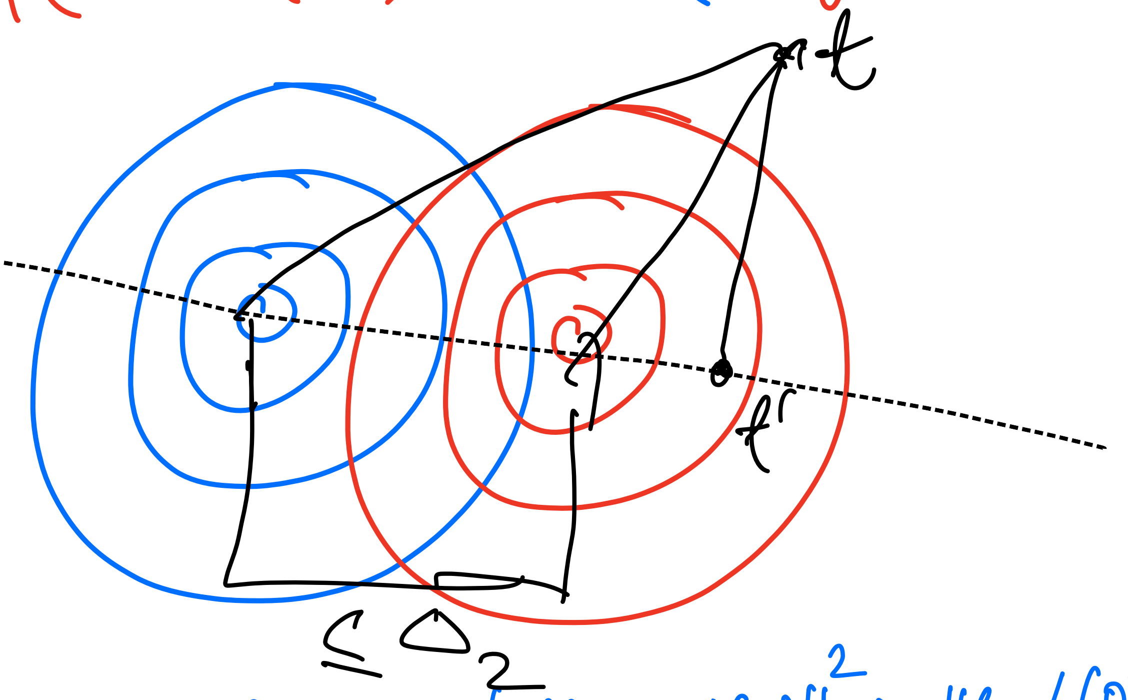
- $\delta \leq 1/n$ or 10^{-5}

Recap

- Gaussian mechanism
- If f is Δ_2-sensitive wrt ℓ_2 norm, Gaussian mechani
  - output $f(D) + \mathcal{N}(0\ ,\ \sigma^2 I_d)$
  - How large should σ^2 be for (ε, δ)-DP ?

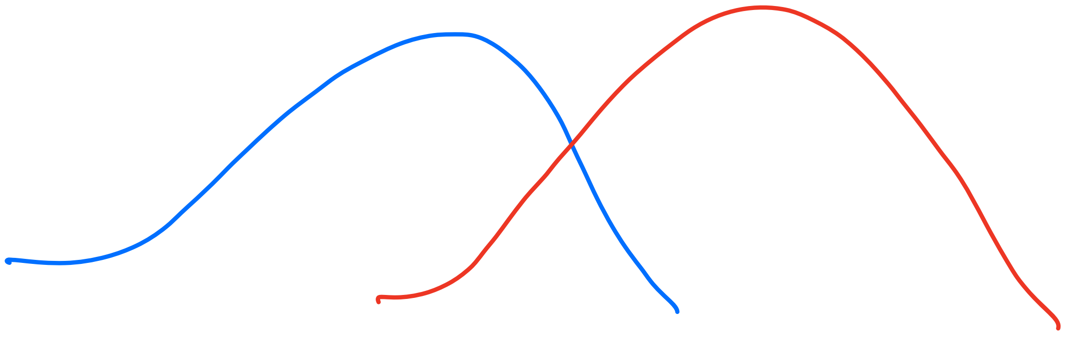


$$\log \frac{P(Y=t \mid D)}{P(Y=t \mid D')} = \log\left(\frac{\mathcal{N}(f(D), \sigma^2 I_d)}{\mathcal{N}(f(D'), \sigma^2 I_d)}\right)$$



$$\log \exp\left(\frac{1}{2\sigma^2}\left(\underline{-\|t-f(D)\|^2} + \underline{\|t-f(D')\|^2}\right)\right)$$

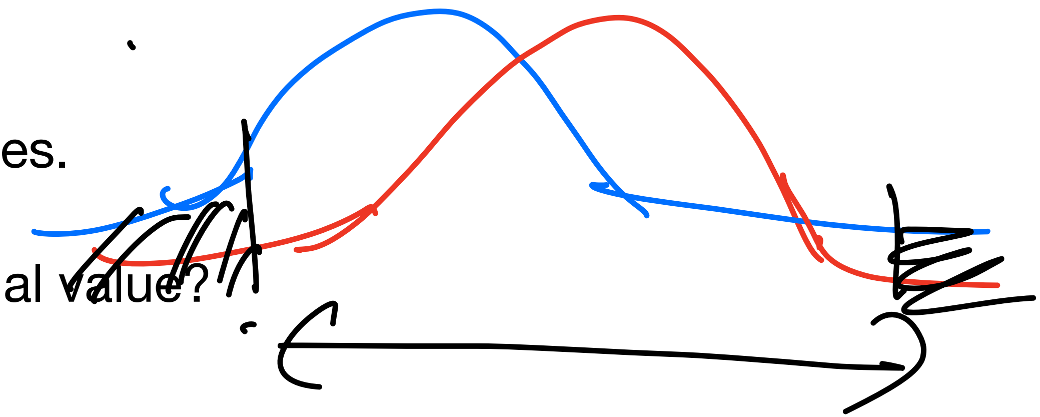
Project onto line joining centers



$$\leq \frac{1}{2\sigma^2}\left(\Delta_2\left(2t-\left(f(D)+f(D')\right)\right)\right)$$

Analytic vs. programatic Gaussian Mechanism

- Original paper [Dwork et al. 2006]:
  - $\sigma \geq \frac{\Delta_2 \sqrt{2 \ln(1.25/\delta)}}{\varepsilon}$ suffices.



- But suboptimal. What is the optimal value?
- [Balle and Wang ICML '18]:
  - $u^* = \arg\min_u \Phi(-\sqrt{\varepsilon u}) - \Phi(-\sqrt{\varepsilon(u+2)})$
  - Use $\sigma = (\sqrt{2+u^*} + \sqrt{u^*}) \cdot \frac{\Delta_2}{\varepsilon}$
- Code: https://github.com/BorjaBalle/analytic-gaussian-mechanism

Post-processing and composition

Post-processing

- You can never undo the output of a DP-algorithm

Theorem

$A : \mathscr{X}^n \to \mathbb{R}^d$ is a (ε, δ)-DP algorithm and f is a mapping independent of $\mathscr{X}$, then $f \circ A$ is (ε, δ)-DP

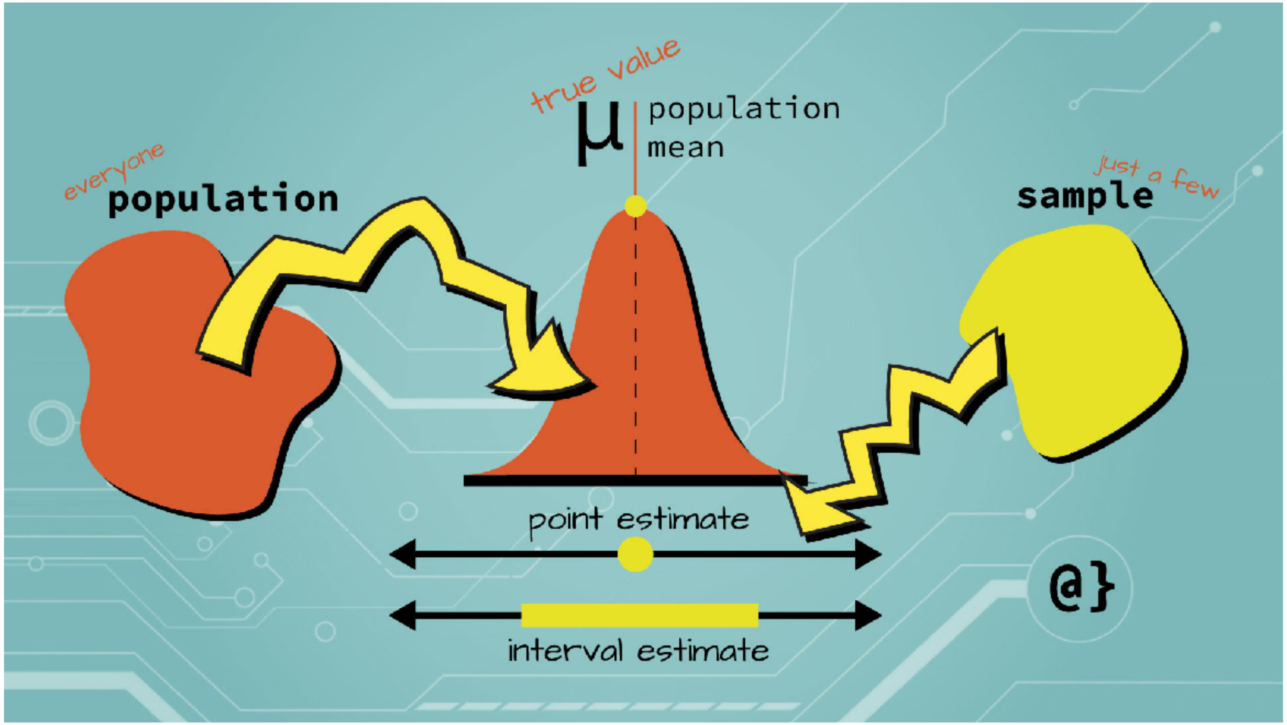
- Upshot: we can plug in our private gradients into any optimizer (e.g. AdamW).

Outline for today

How to make ML private?

- Making mean estimation private
- ML training
- Private ML training

Privacy vs. utility: mean



Binary Mean Estimation

Utility of exact mean

$x_i \sim \mathrm{Ber}(p)$

- We have n i.i.d samples $(x_1, \ldots, x_n)$ where $x_i \in \{0,1\}$.
- Estimate mean as $\underline{\hat{\mu}} = \frac{1}{n}\sum_{i=1}^{n} x_i$. What is the expected error?

$\mathbb{E}[\hat{u}] = p$

$$\mathbb{E}(\hat{u} - p)^2 = \frac{p(1-p)}{n} \boxed{= O\left(\frac{1}{n}\right)}$$

sensitivity $\Delta = \frac{1}{n}$

$$D = \{x_1, \ldots, x_n\}$$

$$f(D) = \frac{1}{n}\sum_{i=1}^{n} x_i \, , \quad f \text{ is } \text{'}\tfrac{1}{n}\text{'-sensitive}$$

output $\mu_{pr} = \frac{1}{n}\sum_{i=1}^{n} x_i + \text{Lap}\left(\frac{1}{n\varepsilon}\right)$

ε-DP

Cost of ε-DP.

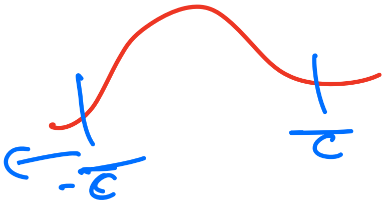
New error

$$\mathbb{E}\,(\mu_{pr} - p)^2 \leq \underbrace{\frac{1}{n}}_{\text{data}} + \underbrace{\frac{1}{n^2\varepsilon^2}}_{\text{extra privacy}}$$

$$x_i \sim N(\mu, \sigma^2)$$

$$\hat{\mu} = \underline{\frac{1}{n} \sum_i x_i}$$ → unbounded sensitivity

$$\mathbb{E}\,(\hat{\mu} - \mu)^2 = \frac{\sigma^2}{n}$$



$$\text{Clip}_\tau(x_i) = \text{sign}(x_i)\,\min(\tau, |x_i|)$$

$$\mu_\tau = \frac{1}{n} \sum_i \text{Clip}_\tau(x_i)$$

↳ sensitivity bounded $\frac{2\tau}{n}$!

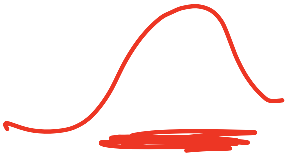
ε-DP

$$\mu_{\text{private}} = \frac{1}{n} \sum_i \text{Clip}_\tau(x_i) + \text{Lap}\left(\frac{2\tau}{n\epsilon}\right)$$

$$O\left(\frac{\sigma^2}{n} + \frac{\tau^2}{n^2\epsilon^2} + \text{Clipping}\right)$$

Error : $\mathbb{E}\,(\mu_{\text{private}} - \mu)^2 \le$ data + added noise + Clipping

(ε, δ) - DP



$$x_i \sim N(\overset{\text{unknow}}{\mu}, \underset{\text{known}}{\sigma^2})$$

① output $\frac{1}{n}\sum_i x_i$ is there an (ε, δ) s.t. it satisfies (ε, δ)-DP ?

② output $\frac{1}{n}\sum_i x_i + N(0, \rho^2)$.

Now can this be (ε, δ)-DP ?

Binary Mean Estimation

Utility of the private mean

- We have n i.i.d samples $(x_1, \ldots, x_n)$ where $x_i \in \{0,1\}$.
- Estimate mean as $\hat{\mu} = \frac{1}{n} \sum_{i=1}^{n} x_i +$ Lap(Δ/ε). What is Δ?
- Net error is $\frac{1}{n} + \frac{2}{n^2 \varepsilon^2}$.
- Error with Gaussian mechanism is similar.

Unbounded Mean Estimation

Utility of exact mean

- We have n i.i.d samples $(x_1, \dots, x_n)$ with $E\|x_i\|_2^2 \le \sigma^2$.
- Estimate mean as $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$. What is the expected error?
- What is the sensitivity?

$$\frac{1}{n}\sum_{i} \mathrm{Clip}_{\ell_1}(x_i) = \frac{x_i}{\|x_i\|_1}\min(\tau, \|x_i\|_1)$$

$$+ \mathrm{Lap}\left(\frac{2\tau}{n\varepsilon}\right)$$

$$O\left(\boxed{\frac{\sigma^2}{n}} + \frac{d\tau^2}{n^2\varepsilon^2} + \frac{\sigma^4}{\tau^2}\right)$$

Unbounded Mean Estimation

$$\tau^2 = O\left(\frac{\sigma^2 n \varepsilon}{\sqrt{d}}\right)$$

Bounding sensitivity

- We have n i.i.d samples $(x_1, \ldots, x_n)$ with $E\|x_i\|_2^2 \leq \sigma^2$.

- Let us clip x_i with a threshold of τ. The expected error is $\leq \frac{\sigma^4}{\tau^2} + \frac{\sigma^2}{n}$.

bad

$d \gg 1$

$$\text{Error} \leq O\left(\frac{\sigma^2}{n} + \frac{\sigma^2 \sqrt{d}}{n\varepsilon}\right)$$

comparable

priv cost

Unbounded Mean Estimation

Utility of private mean

- We have n i.i.d samples $(x_1, \ldots, x_n)$ with $E[x_i^2] \leq \sigma^2$.
- Output $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} \text{clip}_\tau(x_i) + \text{Lap}(2\tau/n\varepsilon)$.

Theorem

$\hat{\mu}$ with $\tau = \sigma\sqrt{n\varepsilon}/2$ satisfies ε-DP and has an error

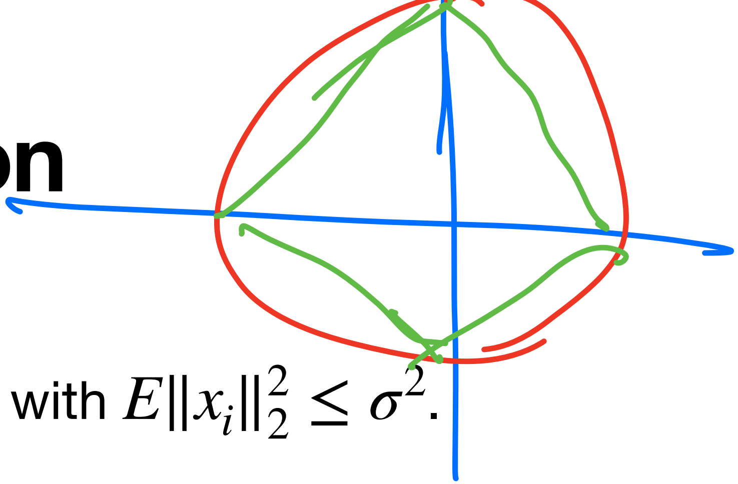
$$E[(\hat{\mu} - \mu)^2] \leq \frac{\sigma^2}{n} + \frac{4\sigma^2}{n\varepsilon}$$

Unbounded Mean Estimation

Utility of private mean

- We have n i.i.d samples $(x_1, \ldots, x_n)$ for $x_i \in \mathbb{R}^d$ with $E\|x_i\|_2^2 \leq \sigma^2$.
- Output $\frac{1}{n}\sum_{i=1}^{n} \text{clip}_\tau(x_i) + \mathcal{N}(0, \rho^2)$ for $\rho = 2\tau \log(2/\delta)/n\varepsilon$.
- Error?

Unbounded Mean Estimation



Utility of private mean

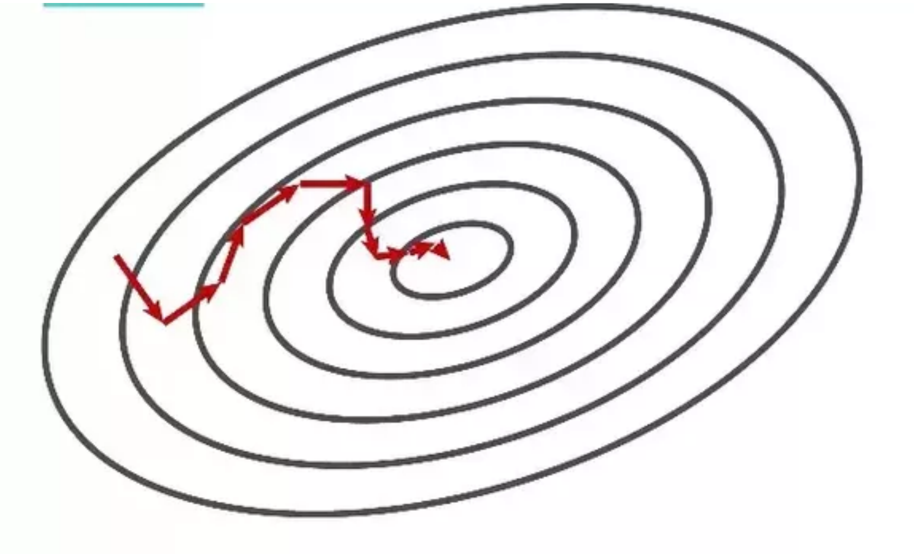
- We have n i.i.d samples $(x_1, \ldots, x_n)$ for $x_i \in \mathbb{R}^d$ with $E\|x_i\|_2^2 \leq \sigma^2$.
- Output $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} \text{clip}_\tau(x_i) + \mathcal{N}(0, \rho^2)$ for $\rho = 2\tau \log(2/\delta)/n\varepsilon$.
- Error?

Theorem

$\hat{\mu}$ with $\tau = O(\sigma\sqrt{n\varepsilon}/d^{1/4})$ satisfies (ε, δ)-DP and has an error

$$E[(\hat{\mu} - \mu)^2] \leq O\left(\frac{\sigma^2}{n} + \frac{\sigma^2\sqrt{d}\log(1/\delta)}{n\varepsilon}\right)$$

Training ML models: GD



Machine Learning

How to train a model?

- We are given i.i.d data: $(x_1, y_1), \ldots, (x_n, y_n)$.
- We have a parameterized family of predictors $f(x;\theta) : \mathscr{X} \to \mathscr{Y}$.
  - Linear models $f(x;\theta) = \boxed{\theta}^\top x$ → param
  - Neural Networks $f(x;\theta) = \theta_2^\top \cdot \text{Relu}(\theta_1^\top x)$

Machine Learning

How to train a model?

$$\theta^* = \arg\min_{\theta} \mathbb{E}_{x,y} \{ \ell(f(x;\theta), y)]$$

- We are given i.i.d data: $(x_1, y_1), \ldots, (x_n, y_n)$.

- We have a parameterized family of predictors $f(x;\theta) : \mathcal{X} \to \mathcal{Y}$.

    - Linear models $f(x;\theta) = \theta^\top x$

    - Neural Networks $f(x;\theta) = \theta_2^\top \cdot$ Relu$(\theta_1^\top x)$

- We want to find parameters which minimizes test-loss $L(\theta) = E_{(x,y)}[\ell(f(x;\theta), y)]$

-

Machine Learning

How to train a model?

D , D' (x_i', y_i')

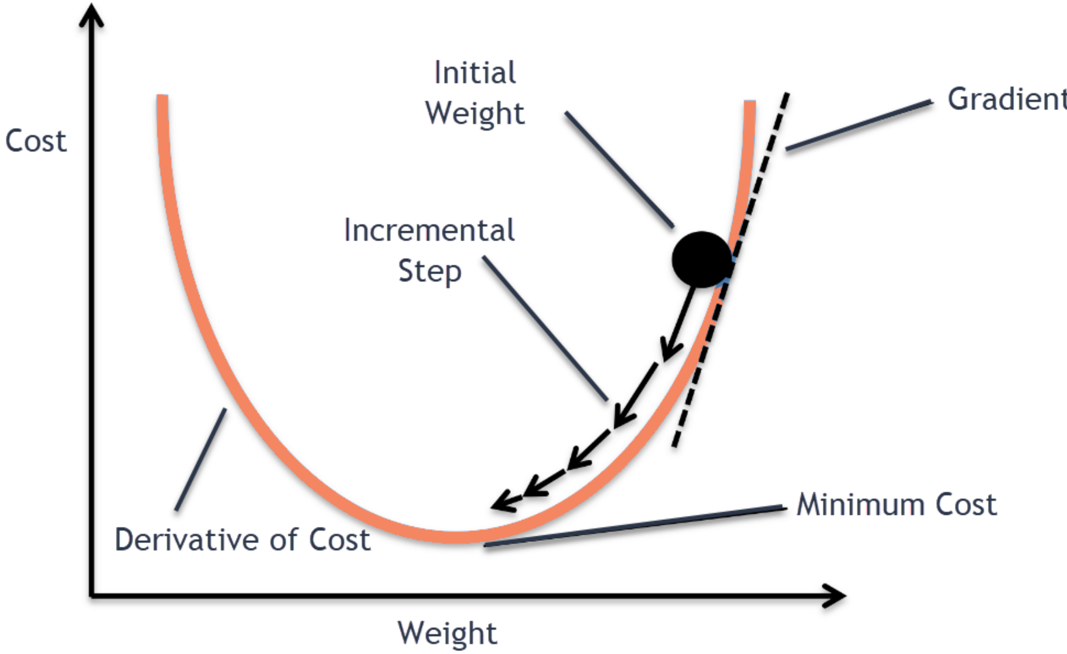
Alg , output

learnt $\hat{\theta}$

- We are given i.i.d data: $\boxed{\cancel{(x_1, y_1)}, \ldots, (x_n, y_n)}$
- We have a parameterized family of predictors $f(x;\theta) : \mathscr{X} \to \mathscr{Y}$.
    - Linear models $f(x;\theta) = \theta^\top x$
    - Neural Networks $f(x;\theta) = \theta_2^\top \cdot \text{Relu}(\theta_1^\top x)$
- We want to find parameters which minimizes test-loss $L(\theta) = E_{(x,y)}[\ell(f(x;\theta), y)]$
- We instead minimize training loss $\underline{\hat{L}(\theta)} = \frac{1}{n}\sum_{i=1}^{n}[\ell(f(x_i;\theta), y_i)]$

Understanding Gradient Descent

- We want to minimize our function $L(\theta)$
- Iterative algorithm. Starting from θ_t in step t,

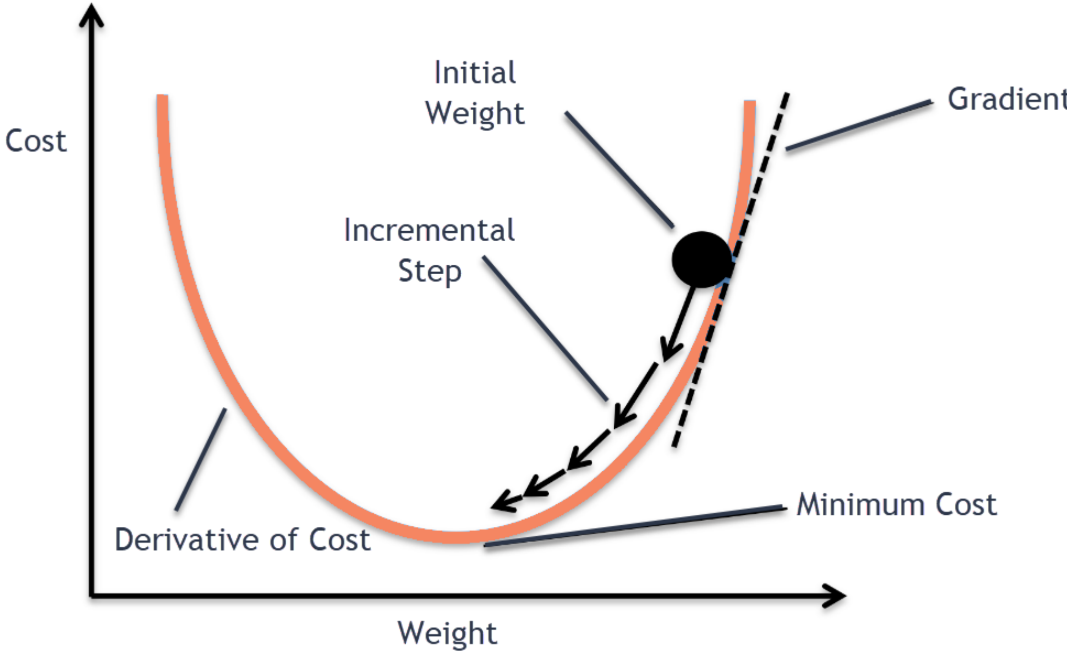


Understanding Gradient Descent

- We want to minimize our function $L(\theta)$
- Iterative algorithm. Starting from θ_t in step t,
- we create a local approximation

$$L(\theta_t + \Delta\theta) \approx L(\theta_t) + \nabla L(\theta_t)^\top \Delta\theta$$

- Move along "steepest" descent direction.

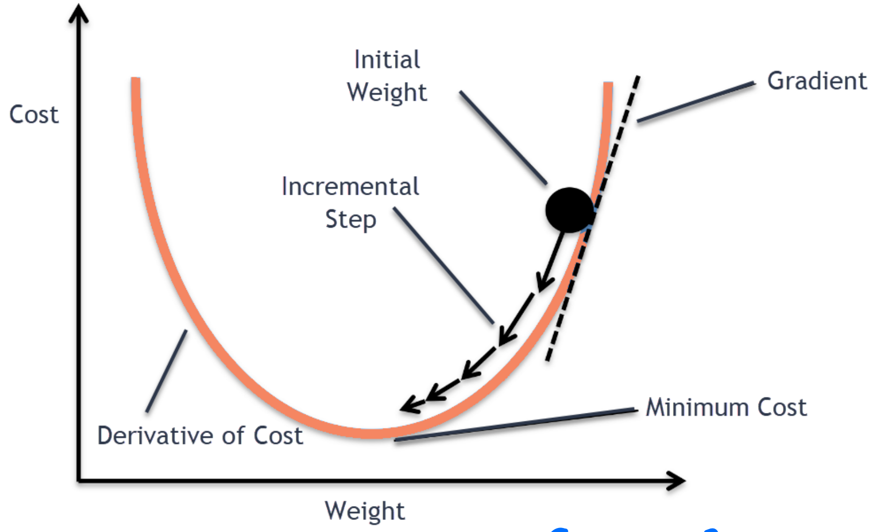


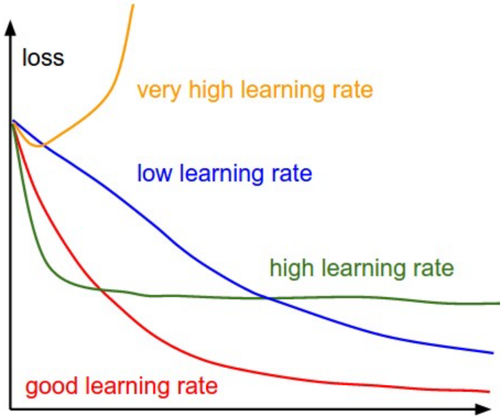
Understanding Gradient Descent

$z_i = (x_i, y_i)$

Algorithm

- Initialize θ_0
- For t=1, …, T
  - $\theta_t = \theta_{t-1} - \gamma_t \nabla L(\theta_{t-1})$
- How to decide γ_t?





$L(\theta) = \frac{1}{n} \sum_i L(\theta, z_i)$

$\nabla L(\theta) = \frac{1}{n} \sum_i \nabla L(\theta, z_i)$

linear clasifier

$$l(\theta) = \frac{1}{2}\left(\theta^T x - y\right)^2$$

$$\boxed{\nabla l(\theta)} = \left(\theta^T x - y\right) \boxed{x}$$

Output $\frac{1}{n}\sum_i \underbrace{\nabla L(\theta, z_i)}_{\text{Clip the gradient}}$

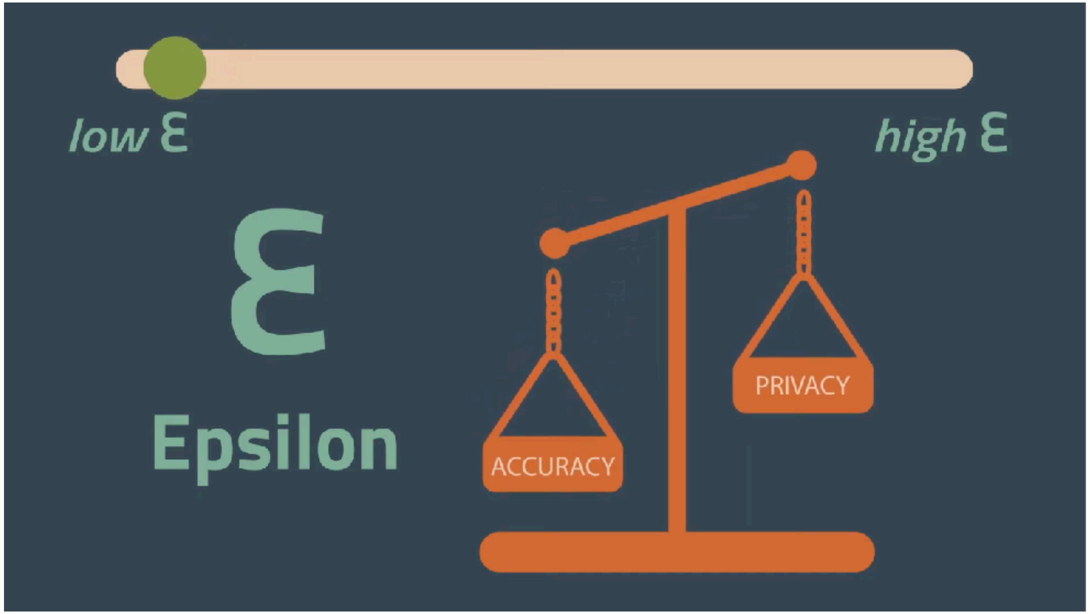
Sensitivity $= \frac{2C}{n}$

$$\tilde{\nabla} = \frac{1}{n}\sum_i \text{Clip}_C\left(\nabla L(\theta, z_i)\right) + \text{Lap}\left(\frac{2C}{n\varepsilon}\right) \Big/ N\left(0, \left(\frac{2C}{\varepsilon n}\log\frac{2}{\delta}\right)^2\right)$$

output $\boxed{\theta_0 - \tilde{\nabla}}$

full batch grad

Making Gradient Descent Private: Composition



Private full-batch gradient descent

Algorithm

- Starting from θ_0, at each time step we update
    - $\theta_t = \theta_{t-1} - \gamma \frac{1}{n} \sum_{i=1}^{n} \nabla_\theta \ell(f(x_i; \theta), y_i)$
- To make it private
    - $\theta_t = \theta_{t-1} - \gamma \frac{1}{n} \sum_{i=1}^{n} \text{Clip}_\tau \left(\nabla_\theta \ell(f(x_i; \theta), y_i) \right)$ + noise
    - Assume scalar for now. So noise = $Lap(??)$

Private full-batch gradient descent

One-step privacy

- Suppose we just run step of
  $\theta_t = \theta_{t-1} - \gamma \frac{1}{n} \sum_{i=1}^{n} \text{Clip}_\tau \left(\nabla_\theta \ell(f(x_i;\theta), y_i) \right) + Lap(??)$
- Sensitivity? How much noise?
- How to reason about what happens across time steps?

Post-processing and composition

Post-processing

- You can never undo the output of a DP-algorithm

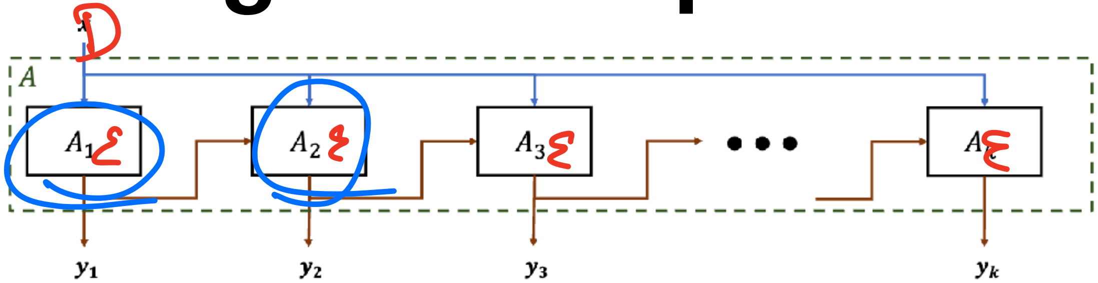
Theorem

$A : \mathscr{X}^n \to \mathbb{R}^d$ is a (ε, δ)-DP algorithm and f is a mapping independent of $\mathscr{X}$, then $f \circ A$ is (ε, δ)-DP

- Upshot: we can plug in our private gradients into any optimizer (e.g. AdamW).

Post-processing and composition

Composition



ε-DP

- What if the new function also depends on our data?

$(k \cdot \varepsilon)$-DP

Theorem

$A : \mathscr{X}^n \to \mathbb{R}^d$ is a $(\varepsilon_1, 0)$-DP algorithm and
$B : \mathscr{X}^n \to \mathbb{R}^d$ is a $(\varepsilon_2, 0)$-DP algorithm, then
$(A, B) : \mathscr{X}^n \to \mathbb{R}^d \times \mathbb{R}^d$ is $(\varepsilon_1 + \varepsilon_2, 0)$-DP

$$\log \frac{\Pr\{y_{\overset{i}{\cancel{k}}} = t \mid D, y_1, \dots, y_{i-1}]}{\Pr\{y_{\overset{i}{\cancel{k}}} = t \mid D', y'_1, \dots, y'_{i-1}]} \leq \varepsilon$$

$$\log \frac{\Pr\{(y_1, y_2, \dots, y_k) = (t_1, \dots, t_k) \mid D]}{\Pr[(y_1, \dots, y_k) = (t_1, \dots, t_k) \mid D']}$$

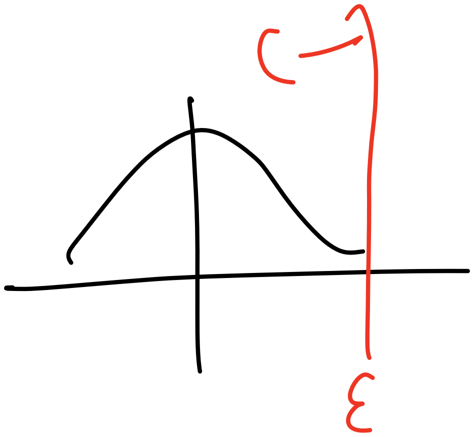
$$\boxed{\log \frac{\Pr[y_1 = t_1 \mid D]}{\Pr(y_1 = t_1 \mid D']}} + \log \frac{\Pr\{y_2 = t \mid D, y_1 = t_1]}{\Pr\{y_2 = t \mid D, y'_1 = t_1)} \leq ?$$

$$+ \quad \dots \quad + \log \frac{\Pr\{y_k = t_k \mid D, y_1, \dots, y_{k-1}]}{\log \Pr\{y_k = t_k \mid D', y'_1 \dots, y'_{k-1}]}$$

$$\leq \varepsilon$$

total probability

$$\leq k \cdot \varepsilon$$



Private full-batch gradient descent

Multi-step privacy

post process+ befor

- One step is $(\varepsilon,0)$-DP

$$\theta_t = \theta_{t-1} - \gamma \frac{1}{n} \sum_{i=1}^{n} \text{Clip}_\tau \left(\nabla_\theta \ell(f(x_i;\theta), y_i) \right) + Lap(2\tau/n\varepsilon)$$

- k-steps of full-batch gradient descent is $(k\varepsilon,0)$-DP.

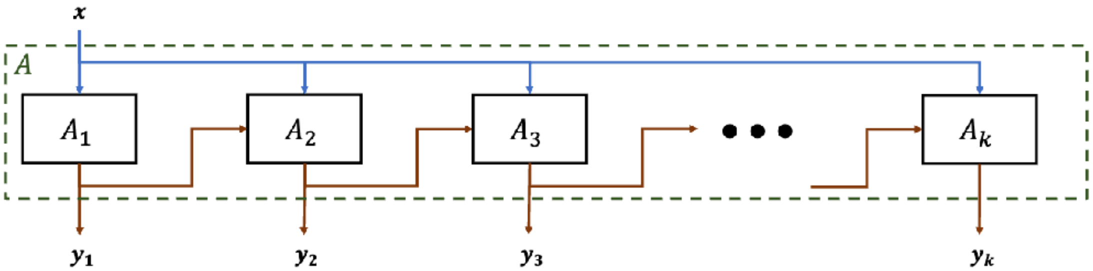
composition.

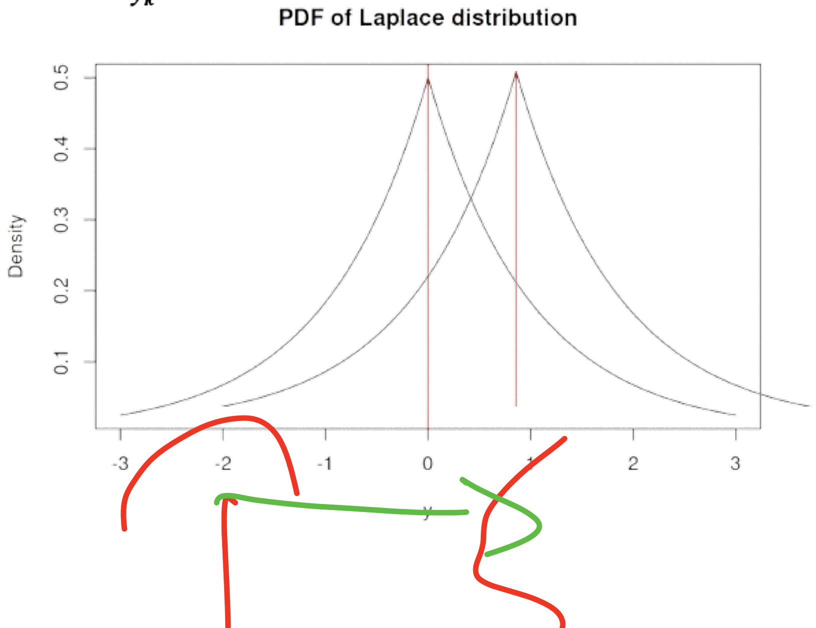
more iterations => less privacy

- We can do better!

Private full-batch gradient descent

Advanced composition





- Let us compute the privacy random variable:

$$R = \log\left(\frac{Pr[A(D) = t]}{Pr[A(D') = t]}\right) \qquad \text{for } t \sim A(D)$$

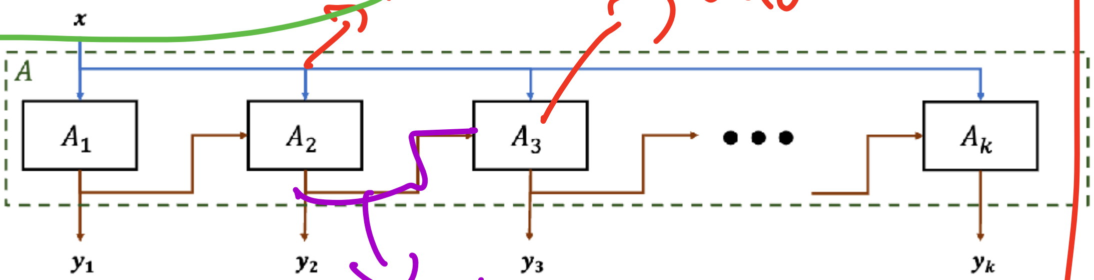
- $R \in [-\varepsilon, \varepsilon]$ and has mean 0.

Private full-batch gradient descent

Advanced composition

ind noise

add ind noise



dependence

- Privacy random variable of composition:

$$R = \sum_{i=1}^{k} \log\left(\frac{Pr[A_i(D) = t_i]}{Pr[A_i(D') = t_i]}\right) = \sum_{i=1}^{k} R_i$$

- $R_i \in [-\varepsilon, \varepsilon]$, 0-mean, conditionally independent.

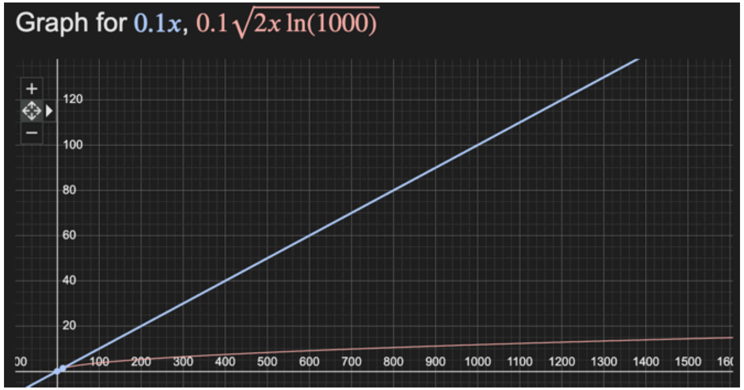
Private full-batch gradient descent

Aside: Azuma's inequality

Azuma's inequality

Given $X_1, \ldots, X_n$ where $E[X_i \,|\, \text{past}] = 0$, $|X_i| \leq \varepsilon_i$. Then,

$$Pr[\textstyle\sum_{i=1}^{k} X_i \geq \Delta] \leq \exp(-\Delta^2/2\textstyle\sum_{i=1}^{k} \varepsilon_i^2)$$

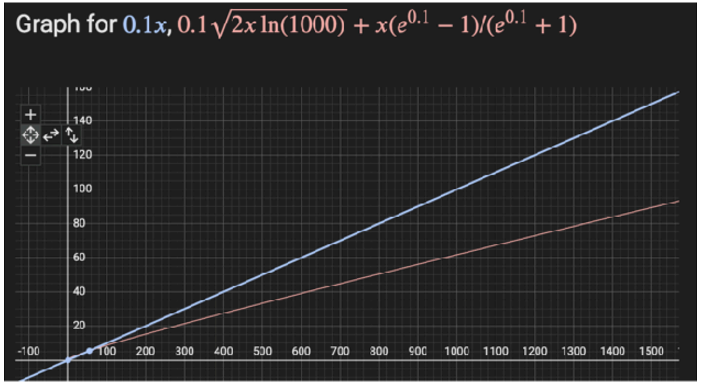

Graph for $0.1x$, $0.1\sqrt{2x\ln(1000)}$

$(\varepsilon \cdot k, 0)$

- $R_i \in [-\varepsilon, \varepsilon]$, 0-mean, conditionally independent.
- $Pr[\sum_{i=1}^{k} R_i \geq \varepsilon\sqrt{2k\log(1/\delta)}] \leq \delta$ i.e. we have $(\varepsilon\sqrt{2k\ln(1/\delta)}, \delta)$-DP!

Private full-batch gradient descent

Advanced composition



Graph for $0.1x$, $0.1\sqrt{2x\ln(1000)} + x(e^{0.1} - 1)/(e^{0.1} + 1)$

Theorem. Advanced Composition

A combination of $A_1 \circ A_2 \circ A_k$, each of which is (ε, δ)-DP is $(\tilde{\varepsilon}, \tilde{\delta})$-DP where

$$\tilde{\varepsilon} = \varepsilon\sqrt{2k\ln(1/\delta')} + k\frac{e^{\varepsilon} - 1}{e^{\varepsilon} + 1} \quad \text{and} \quad \tilde{\delta} = k\delta + \delta'$$

For any choice of δ'.

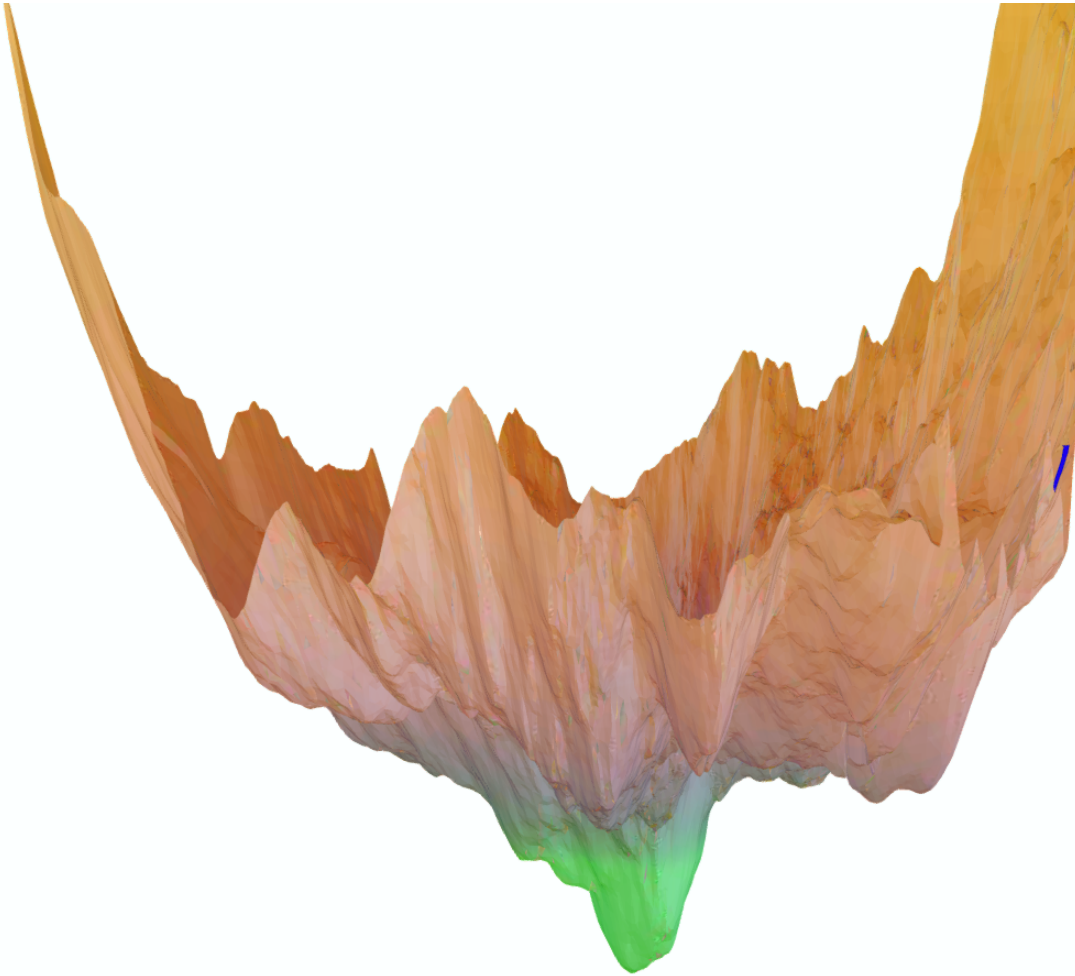
Private full-batch gradient descent

Multi-step privacy

Hyperparameters
→ k
→ τ

- One step is $(\varepsilon, 0)$-DP
  $$\theta_t = \theta_{t-1} - \gamma \frac{1}{n} \sum_{i=1}^{n} \text{Clip}_\tau \left(\nabla_\theta \ell(f(x_i; \theta), y_i) \right) + Lap(2\tau / n\varepsilon)$$
- k-steps of full-batch gradient descent is $(\underbrace{\varepsilon\sqrt{2k \ln(1/\delta)}}_{\text{want} = 10}, \delta)$-DP.
- How about with Gaussian-noise and vectors?

Optimization for Deep Learning



Stochastic Gradient Descent

Convergence analysis

- How do we compute $\nabla L(\theta_t)$?
- We are only given data samples: $(x_1, y_1), \ldots$ i.e. we cannot compute $L(\theta) = E_{(x,y)}[\ell(f(x;\theta), y)]$

Stochastic Gradient Descent

Convergence analysis

- How do we compute $\nabla L(\theta_t)$?
- We are only given data samples: $(x_1, y_1), \ldots$ i.e. we cannot compute $L(\theta) = E_{(x,y)}[\ell(f(x;\theta), y)]$
- SGD says no problem. Just use sample gradient. Initialize θ_0
- For t=1, …, T
  - Sample a data point (x_t, y_t)
  - $\theta_t = \theta_{t-1} - \gamma_t \nabla_\theta \ell(f(x_t; \theta_{t-1}), y_t) = \theta_{t-1} - \gamma_t \nabla \ell_t(\theta_{t-1})$

Gradient Descent Variants

using data more "fully"
= more leakage

only "glances"
= less ~~leakage~~ leakage

?

- we are given n samples $(x_1, y_1), \ldots, (x_n, y_n)$
- We have a few options:
  - Exact gradient: $\nabla_\theta E_{x,y}[\ell(f(x;\theta), y)]$
  - Stochastic gradient: for a random sample (x_i, y_i), $\nabla_\theta \ell(f(x_i;\theta), y_i)$
  - Full-batch gradient: $\frac{1}{n}\sum_{i=1}^{n} \nabla_\theta \ell(f(x_i;\theta), y_i)$
  - Mini-batch gradient: for a sample $\mathscr{B}$, $\frac{1}{|\mathscr{B}|}\sum_{i\in\mathscr{B}} \nabla_\theta \ell(f(x_i;\theta), y_i)$

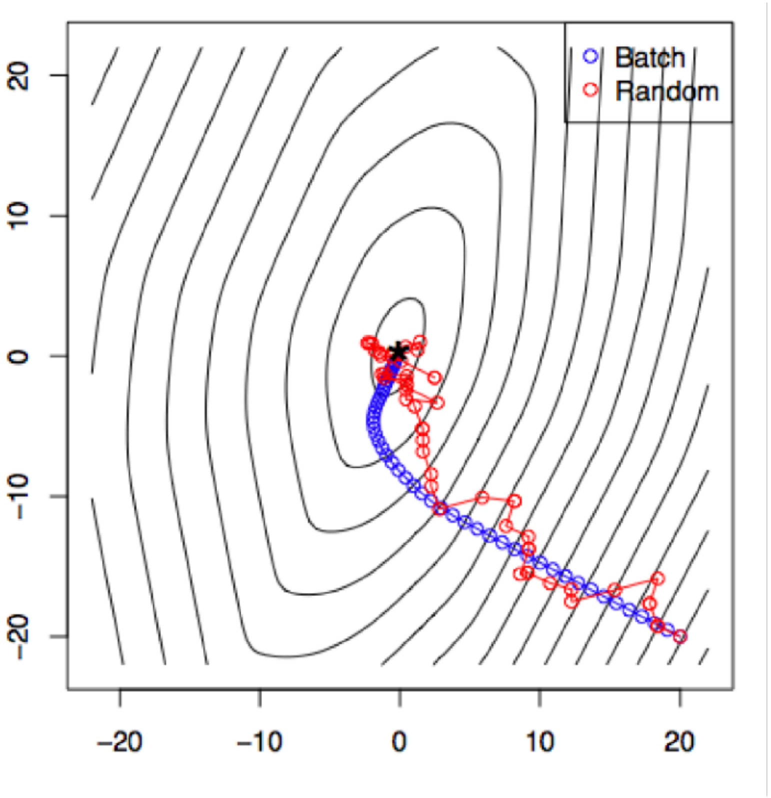
→ practical

less samples to "hide" among

larger sensitivity

Understanding Gradient Descent

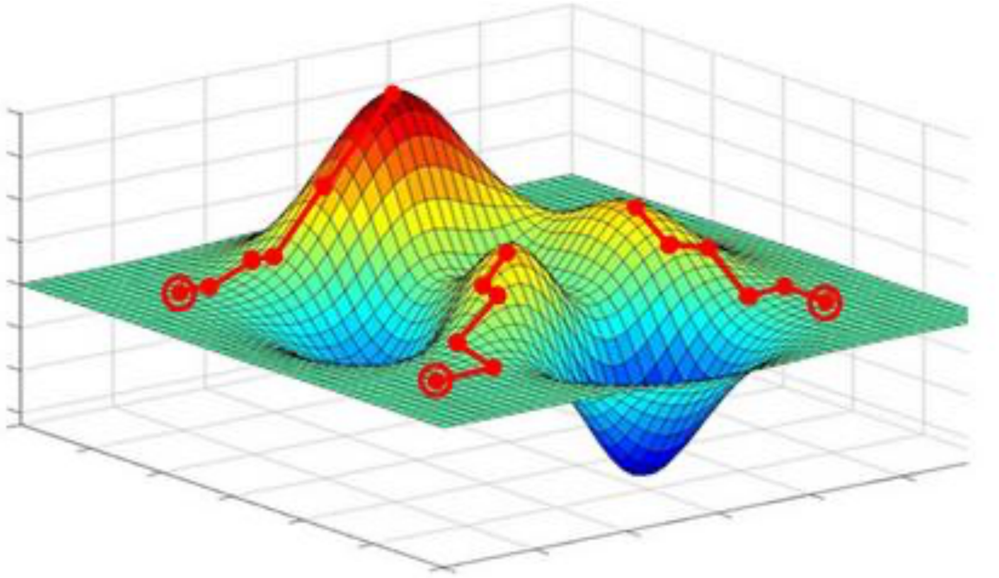
Convergence analysis



Optimization in Deep Learning

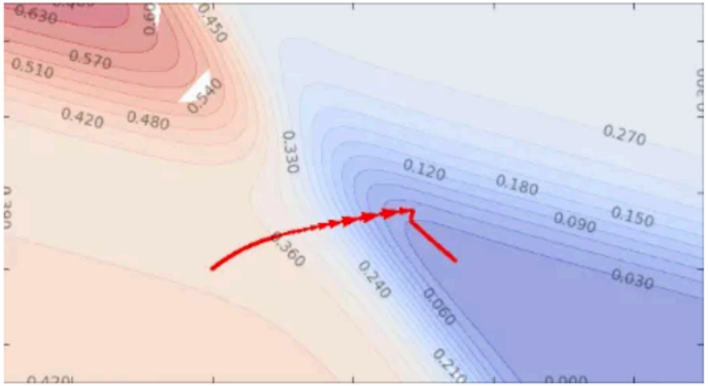
Initialization

- Initialization matters!
- Always start with a pretrained model if you can.

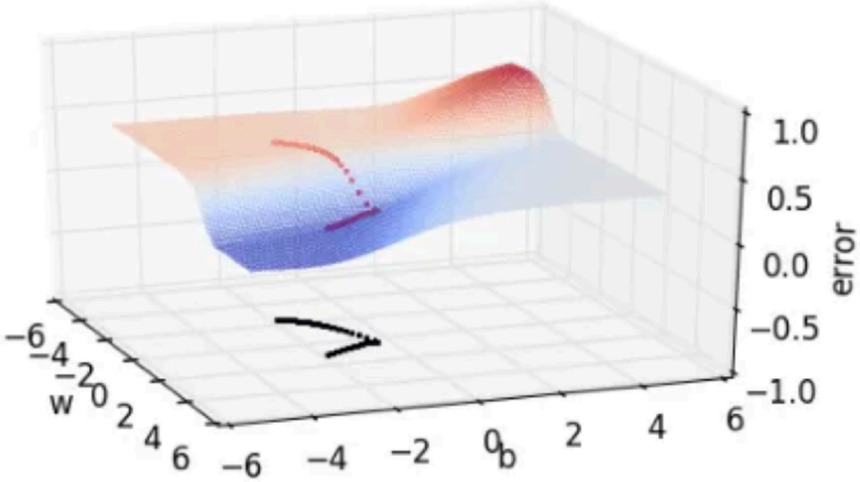


Optimization in Deep Learning

Momentum



Contour Map with GD applied on it. Arrows represent faster fall in error value. Flat surfaces represented by constant color regions.

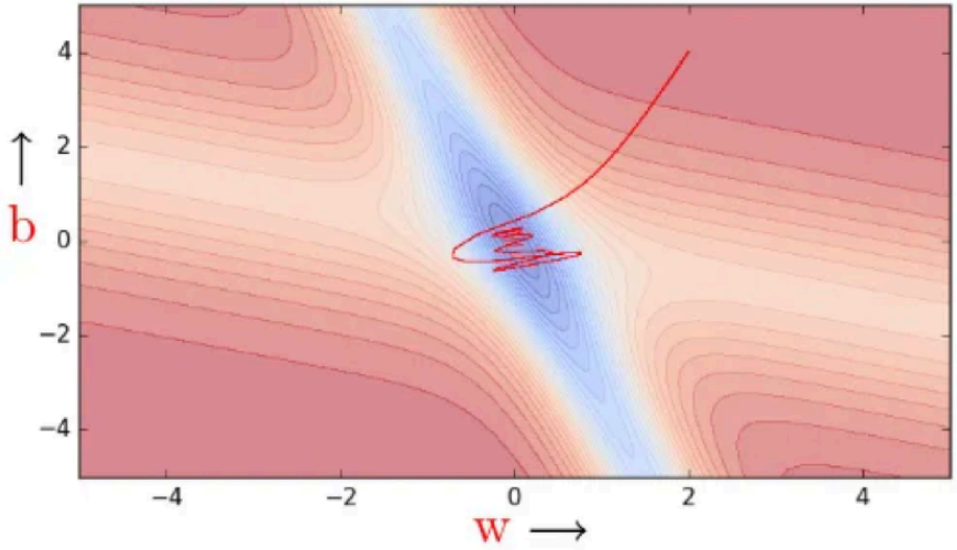


Gradient descent slows down a lot when it encounters large flat sections

Optimization in Deep Learning

Momentum

- Add momentum to speed it up
- $m_t = m_{t-1} + \beta \nabla L(\theta_{t-1})$
- $\theta_t = \theta_{t-1} - \gamma m_t$



It oscillates in the valley of minima. Take lot U-turns, still converges faster than Vanilla GD.

Physical intuition - biking down a hill is faster than walking down

Optimization in Deep Learning

Adaptivity

- We need to keep changing step size since smoothness keeps changing all the time

- Make the step-size adaptive - AdaGrad

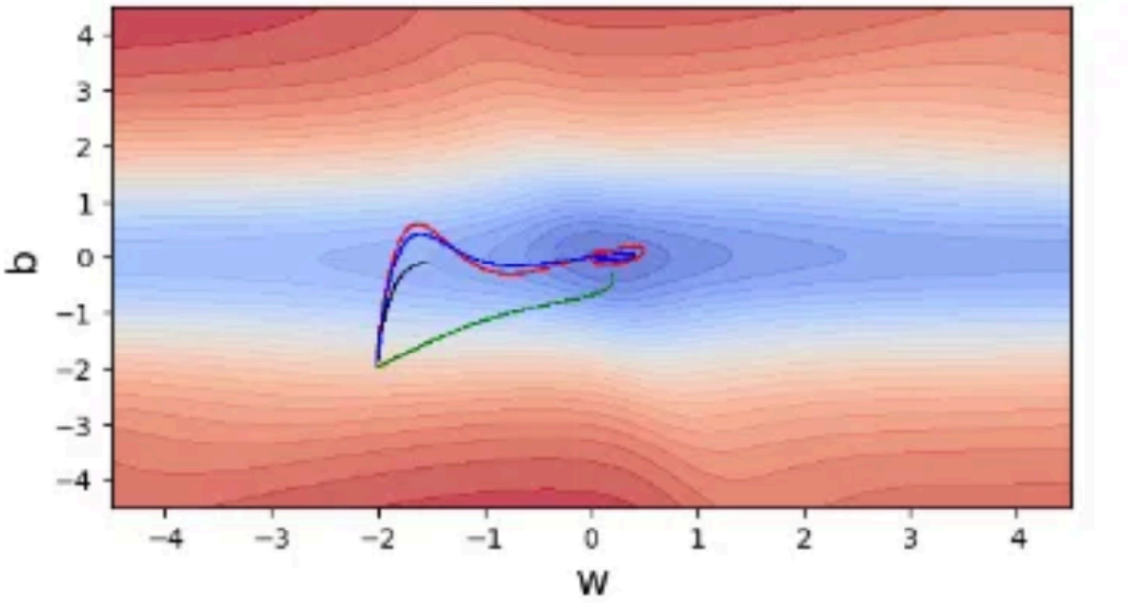
- $v_t = v_{t-1} + \beta_2(\nabla L(\theta_{t-1}))^2$ - running estimate of second moment

- $\theta_t = \theta_{t-1} - \gamma \nabla L(\theta_{t-1}) / \sqrt{v_t + \epsilon}$ - normalize the updates.

Tran et al. 2024 “Empirical Tests of Optimization Assumptions in Deep Learning”

Optimization in Deep Learning

AdaGrad

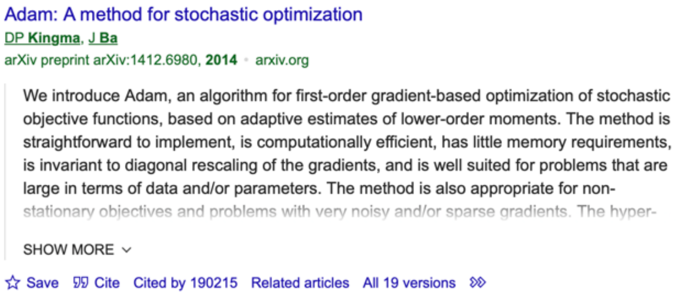


- Progress of AdaGrad in green is much more consistent across steps.
- But it is slower than momentum methods (blue / red)

Optimization in Deep Learning

Adam

- Combine everything - adaptivity + momentum = Adam
- $m_t = m_{t-1} + \beta \nabla L(\theta_{t-1} - \gamma m_{t-1})$ - first moment estimate
- $v_t = v_{t-1} + \beta_2 (\nabla L(\theta_{t-1}))^2$ - second moment estimate
- $\theta_t = \theta_{t-1} - \gamma m_t / \sqrt{v_t + \epsilon}$

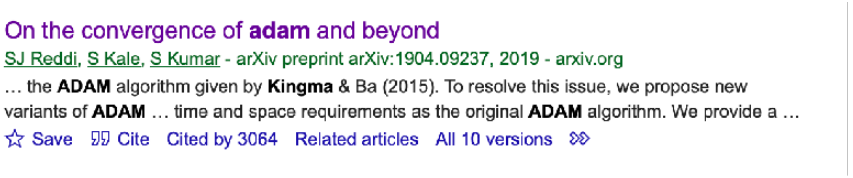


Adam: A method for stochastic optimization

DP Kingma, J Ba

arXiv preprint arXiv:1412.6980, **2014** · arxiv.org

We introduce Adam, an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. The method is straightforward to implement, is computationally efficient, has little memory requirements, is invariant to diagonal rescaling of the gradients, and is well suited for problems that are large in terms of data and/or parameters. The method is also appropriate for non-stationary objectives and problems with very noisy and/or sparse gradients. The hyper-



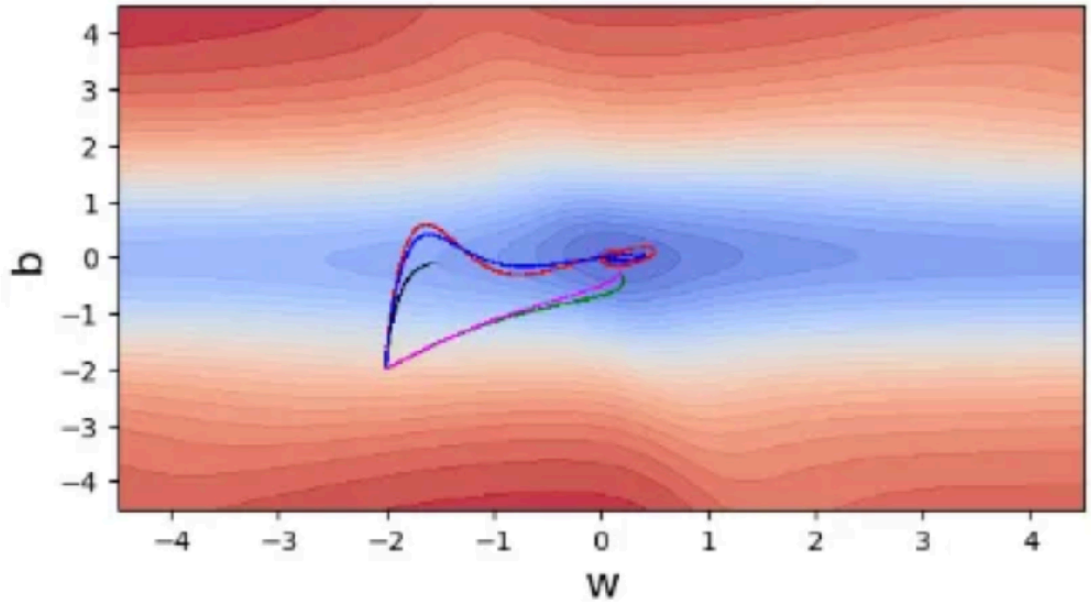
On the convergence of **adam** and beyond

SJ Reddi, S Kale, S Kumar - arXiv preprint arXiv:1904.09237, 2019 - arxiv.org

… the **ADAM** algorithm given by **Kingma** & Ba (2015). To resolve this issue, we propose new variants of **ADAM** … time and space requirements as the original **ADAM** algorithm. We provide a …

Optimization in Deep Learning

Adam



- AdamW (a minor variant) is likely the best default optimizer
- In LLMs, additionally learning rate warm up is used for 5 epochs.

What about privacy?

Can we make SGD private?

- For t=1, …, T
    - Sample a data point (x_t, y_t)
    - $\theta_t = \theta_{t-1} - \gamma_t \nabla_\theta \ell(f(x_t; \theta_{t-1}), y_t) = \theta_{t-1} - \gamma_t \nabla \ell_t(\theta_{t-1})$

Private stochastic gradient descent

Algorithm

- Starting from θ_0, at each time step
    - sample (x_i, y_i) randomly from $(x_1, y_1), \ldots, (x_n, y_n)$
    - $\theta_t = \theta_{t-1} - \gamma \nabla_\theta \ell(f(x_i; \theta), y_i)$
- To make it private
    - $\theta_t = \theta_{t-1} - \gamma \text{Clip}_\tau \left(\nabla_\theta \ell(f(x_i; \theta), y_i) \right) + \text{noise}$
    - Assume scalar for now. So noise = $Lap(??)$

Private stochastic gradient descent

One-step privacy

- Suppose we just run step:
  - $\theta_t = \theta_{t-1} - \gamma \text{Clip}_\tau \left(\nabla_\theta \ell(f(x_t; \theta), y_t) \right) + Lap(2\tau/\varepsilon)$
- No improvement due to n
- Important note: use poisson sampling! Not uniform.
- This makes analyzing what happens to each data-point independent.

Privacy amplification via subsampling

- Given a dataset $D \in \mathscr{X}^n$, and $m \in [n]$
- We define S to be a random m-subsample of D
- Is releasing S private?
- Now suppose A is ε-DP on D. What is the privacy of A composed with subsampling?

Privacy amplification via subsampling

input D

A (Random subset of D)

Theorem. Subsampling Amplification

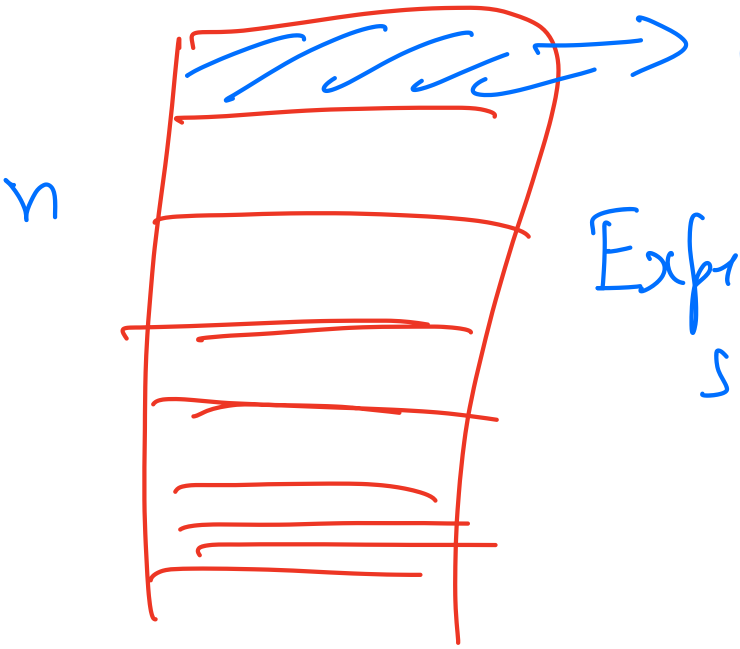
Composing an (ε, δ)-DP A with a sampling rate of q results in an $(\tilde{\varepsilon}, \tilde{\delta})$-DP algorithm where

$= \frac{1}{n}$

$$\tilde{\varepsilon} = \log(1 - q + qe^{\varepsilon}) = O(q\varepsilon) \quad \text{and} \quad \tilde{\delta} = q\delta$$

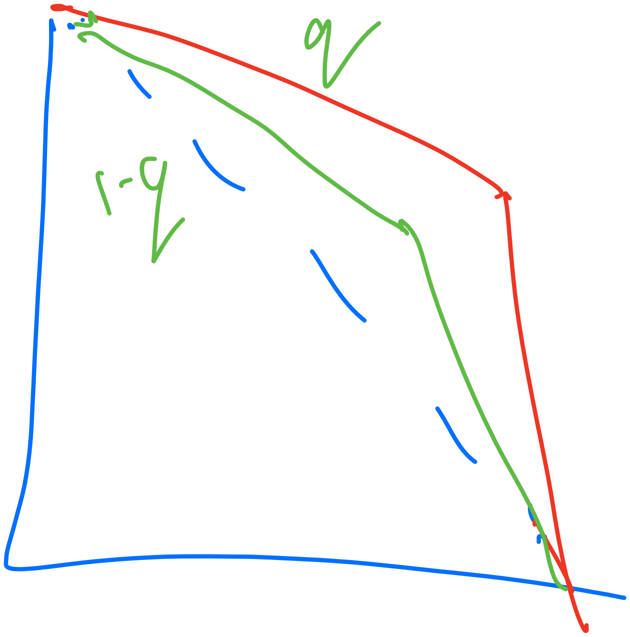
poisson sampling only!

Poisson Sampling



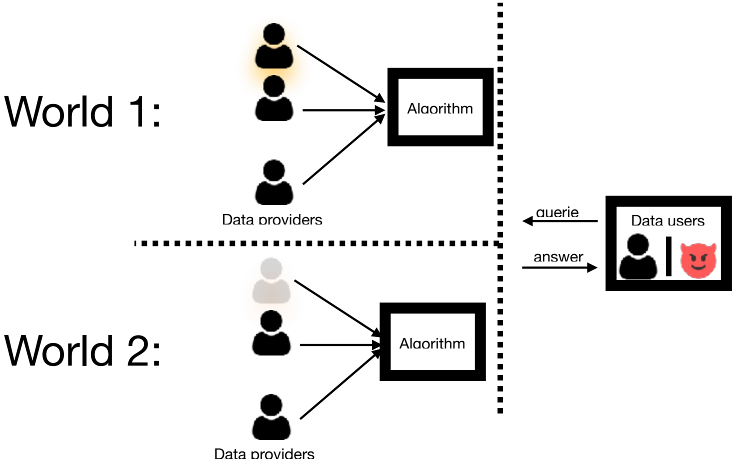
pick with q

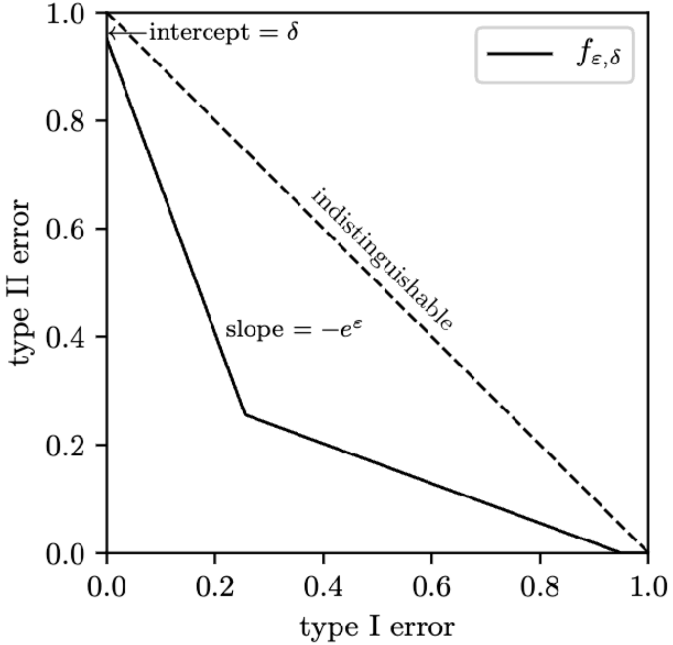
Expected batch size (qn)



Recall

Membership Inference definition of privacy





- Claim: $\beta + (1 - q + qe^{\varepsilon})\alpha \geq 1 - \delta$
- and, $(1 - q + qe^{\varepsilon})\beta + \alpha \geq 1 - \delta$, where $\alpha =$ type I error, $\beta =$ type II error

Private stochastic gradient descent

One-step privacy

- Suppose we just run step:
  - $$\theta_t = \theta_{t-1} - \gamma \overbrace{\text{Clip}_\tau\left(\nabla_\theta \ell(f(x_t;\theta), y_t)\right) + Lap(2\tau/\overset{n\tilde{\varepsilon}}{\varepsilon})}^{\tilde{g}}$$

$n\tilde{\varepsilon} = \varepsilon$

$\tilde{\varepsilon} = \frac{\varepsilon}{n}$

- We have $q = 1/n$. So, we have $\tilde{\varepsilon} = \log(1 - 1/n + e^{\varepsilon}/n) = O(\varepsilon/n)$
- Adding in advanced composition, k rounds of SGD satisfies $(O(\underbrace{\varepsilon/n}_{\tilde{\varepsilon}}\sqrt{k\ln(1/\delta)}), \delta)$-DP

$(\tilde{\varepsilon}\sqrt{k\ln(1/\delta)}, \delta)$ DP

one more hyperparameter

↳ Batch size

Private stochastic gradient descent

One-step privacy

ok to use AdamW by post-processing!

- Suppose we just run step:
    - $\theta_t = \theta_{t-1} - \gamma \text{Clip}_\tau \left(\nabla_\theta \ell(f(x_t;\theta), y_t) \right) + Lap(2\tau/\varepsilon)$
- We have $q = 1/n$. So, we have $\tilde{\varepsilon} = \log(1 - 1/n + e^{\varepsilon}/n) = O(\varepsilon/n)$
- Adding in advanced composition, k rounds of SGD satisfies $(O(\varepsilon/n\sqrt{k \ln(1/\delta)}), \delta)$-DP
- Hyper-parameter - tune sampling rate q in practice

① I don't know my batch size!

- very large → mem errors
- very small → GPU empty

sometimes use standard data loaders.

② Clipping is per data gradient

$$\frac{1}{|B|}\sum_{i\in B} \text{Clip}_C(\nabla \ell(\theta, z_i))$$

⇒ grad accumulation

③ More Hyper-params.