

CS208: Applied Privacy for Data Science

Machine Learning under DP

School of Engineering & Applied Sciences
Harvard University

March 8, 2022

ML Inputs and Loss Functions

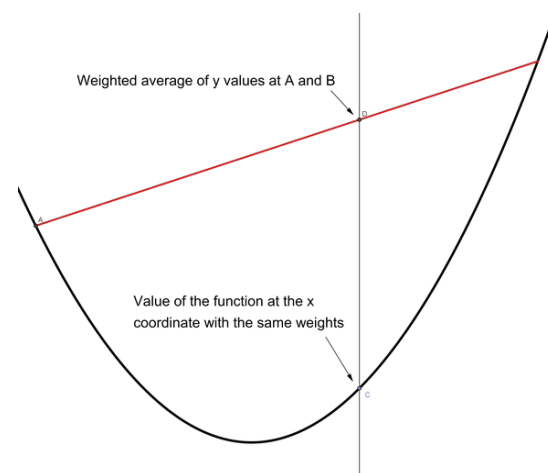
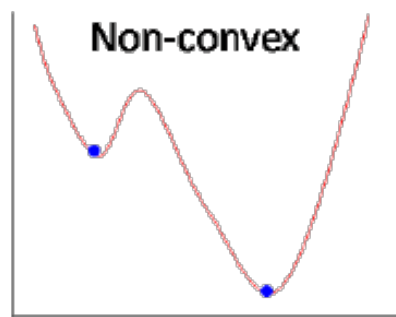
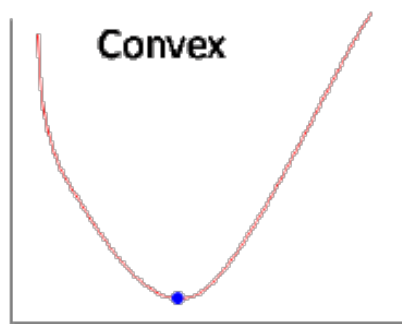
- **Data:** $(x_1, y_1), \dots, (x_n, y_n) \sim \mathcal{P}$
 - Examples $x_i \in \mathcal{X}$ d -dimensional, discrete or continuous
 - Labels $y_i \in \mathcal{Y}$ 1-dimensional, discrete or continuous
 - \mathcal{P} typically unknown
- **A loss function:**
 - $\ell : \Theta \times \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$ $\ell(\theta|x_i, y_i)$ measures ``loss''
 - Define $L: \Theta \rightarrow \mathbb{R}$ $L(\theta) = \frac{1}{n} \sum_{i=1}^n \ell(\theta|x_i, y_i)$
 - Example: Squared loss: $\ell(\theta|x, y) = |(\beta_1 x + \beta_0) - y|^2$.
- **Goal:** output $\hat{\theta} \in \Theta$ s.t.
$$L(\hat{\theta}) \approx \min L(\theta)$$

Convexity

- **Def:** L is **convex** if for all points \vec{a}, \vec{b} , we have

$$L\left(\frac{\vec{a} + \vec{b}}{2}\right) \leq \frac{L(\vec{a}) + L(\vec{b})}{2}.$$

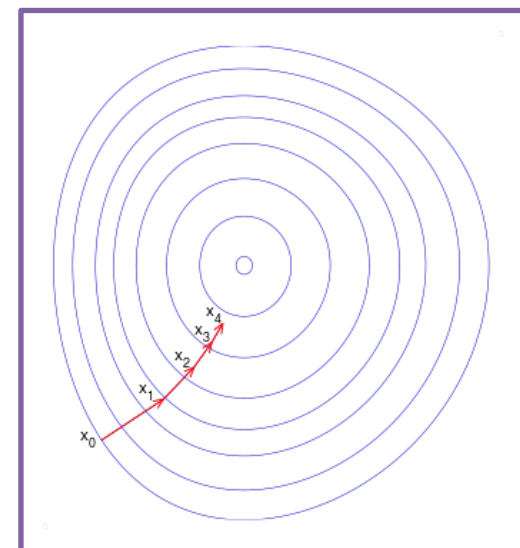
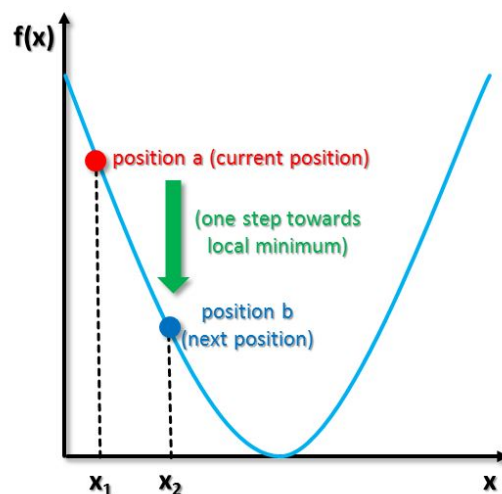
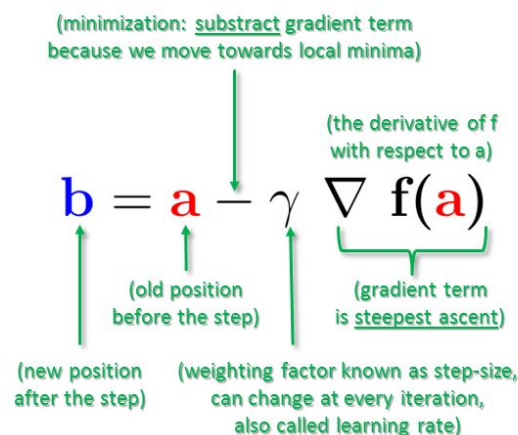
- Convex functions have **no local minima**



- Loss function for logistic regression is convex
 - No closed form solution for minimum, but it is easy to find

Gradient Descent

- Proceed in steps
- Start from (carefully chosen) initial parameters $\hat{\theta}_0$
- At each step, move in direction opposite to the gradient of the loss $\nabla L(\hat{\theta} \mid \vec{x}, \vec{y})$.
- Gradient is the vector of partial derivatives



Gradient Descent

- Specify
 - Number of steps T
 - Learning rate η
- Pick initial point $\hat{\theta}_0 \in \Theta$
- For $t = 1$ to T
 - Compute gradient

$$g_t = \nabla L(\hat{\theta}_{t-1}) = \frac{1}{n} \sum_i \nabla \ell(\hat{\theta}_{t-1} | x_i, y_i)$$

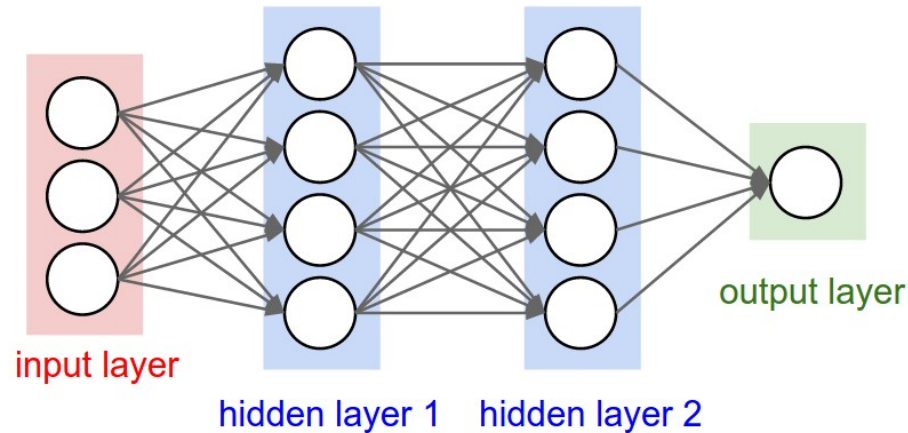
- $\hat{\theta}_t = \hat{\theta}_{t-1} - \eta \cdot g_t$

- Output $\hat{\theta} = \sum_{t=1}^T \hat{\theta}_t$ or $\hat{\theta}_T$

Average iterate

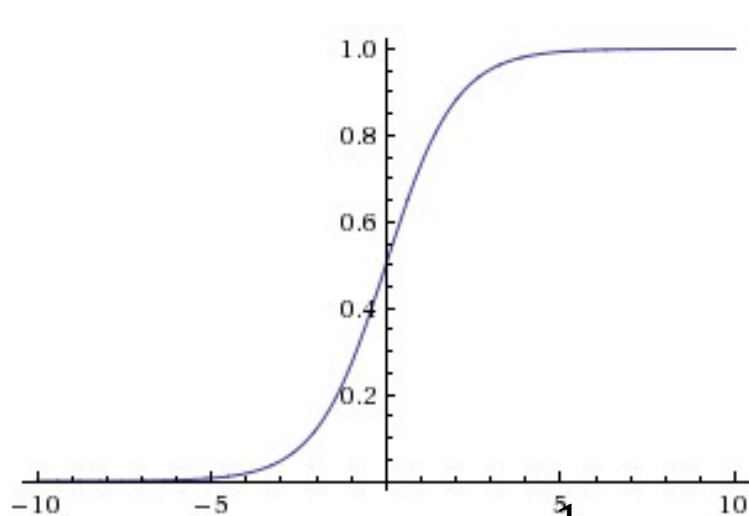
Last iterate

Gradient Descent for Neural Networks

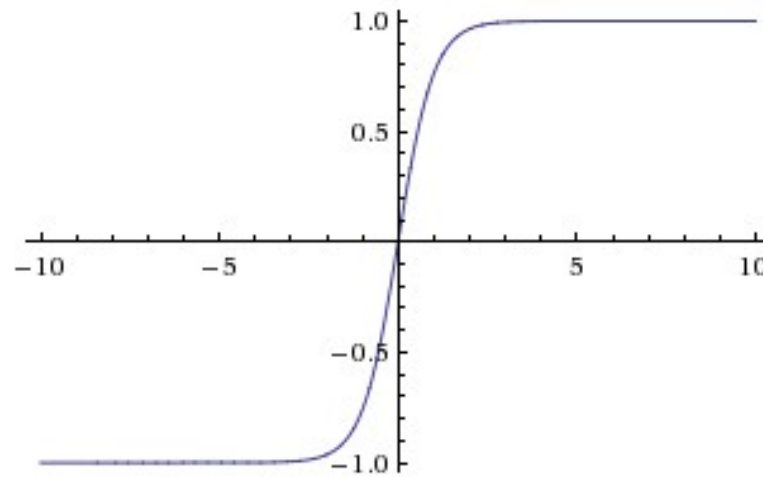


- Each node is a linear function of inputs (specified by θ) composed with a nonlinear “activation” function
- Gradient of Loss function can be computed quickly
 - Using chain rule (technique called “backpropagation”)
- But no longer convex, has many local minima
 - Can get stuck in a bad place
 - But works well in practice!

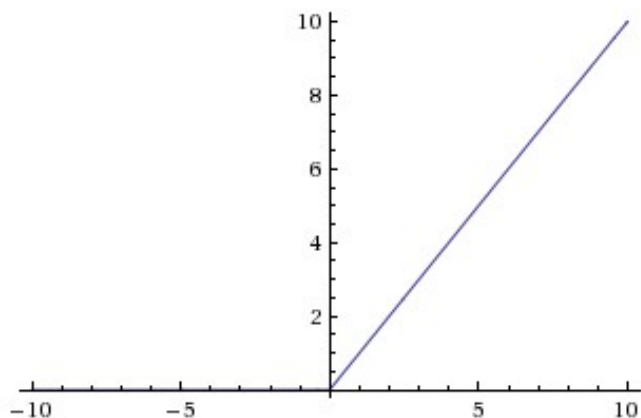
Common Activation Functions



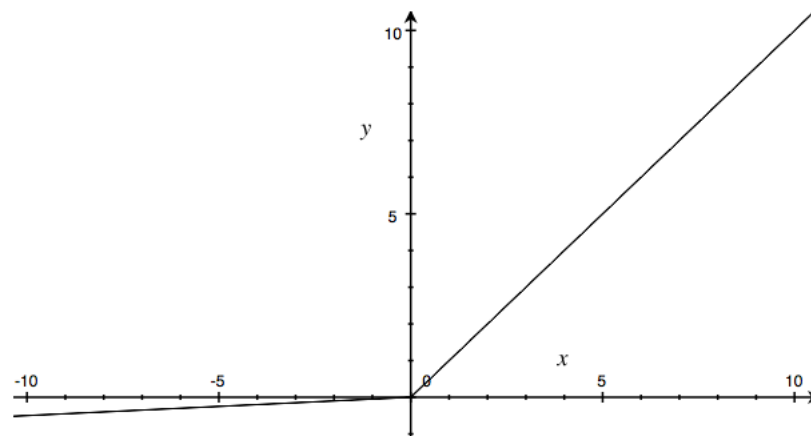
$$\text{Sigmoid } \sigma(x) = \frac{1}{1+e^{-x}}$$



$$\tanh(x) = 2\sigma(2x) - 1$$



$$\text{ReLU}(x) = \max(0, x)$$



$$\text{Leaky ReLU}(x) = \max(0.05x, x)$$

DP for Vector-Valued Functions

- Let $f : \mathcal{X}^n \rightarrow \mathbb{R}^k$, and $M(x) = f(x) + Z$ for noise $Z \in \mathbb{R}^k$.
- global ℓ_2 -sensitivity of f is

$$\text{GS}_{f, \ell_2} \stackrel{\text{def}}{=} \max_{x \sim x'} \|f(x) - f(x')\|_2.$$

$$\|z\|_2 = \left(\sum_j |z_j|^2 \right)^{1/2}$$

- **Gaussian Mechanism:** $Z \sim \mathcal{N}\left(\vec{0}, 2 \left(\frac{\text{GS}_{f, \ell_2}}{\varepsilon}\right)^2 \cdot \ln \frac{1.25}{\delta} \cdot I_k\right)$
 - independent Gaussian noise per coordinate.

Robustness to Noise in Gradient Estimation

- For efficiency:

Sample a minibatch $B \in \{1, 2, \dots, n\}$

Gradient estimate $\tilde{g}_t = \frac{1}{|B|} \sum_{i \in B} \nabla \ell(\hat{\theta}_{t-1}, x_i, y_i)$

Stochastic Gradient Descent (SGD)!

- For privacy:

Add Gaussian Noise $\tilde{g}_t = g_t + \mathcal{N}(0, \sigma^2 I)$

In both cases, \tilde{g}_t is an unbiased estimate of g_t : $E[\tilde{g}_t] = g_t$

DP Gradient Descent

[Williams-McSherry'10, ...]

- Specify
 - Number of steps T
 - Learning rate η
 - Privacy parameters ϵ, δ
 - Clipping parameter C . Write $[\vec{z}]_{\Delta} = \vec{z} \cdot \max\left(1, \frac{C}{\|\vec{z}\|_2}\right)$.
 - Noise variance $\sigma^2 = \text{TBD}(T, \epsilon, \delta, C)$.
- Pick initial point $\hat{\theta}_0$
- For $t = 1$ to T
 - Estimate gradient as **noisy** average of **clipped** gradients
$$\hat{g}_t = \frac{1}{n} \sum_i [\nabla \ell(\hat{\theta}_{t-1} | x_i, y_i)]_C + \mathcal{N}(0, \sigma^2 I)$$
 - $\hat{\theta}_t = \hat{\theta}_{t-1} - \eta \cdot \hat{g}_t$
- Output $\hat{\theta} = \sum_{t=1}^T \hat{\theta}_t$ or $\hat{\theta}_T$

Privacy Analysis

- Proof idea: Show releasing $(\hat{g}_1, \hat{g}_2, \dots, \hat{g}_T)$ satisfies DP
 - Each step (releasing \hat{g}_t) satisfies (ϵ, δ) -DP
 - Adaptive composition across T steps

Privacy Analysis

- By Gaussian Mechanism, each iteration is (ϵ_0, δ_0) -DP if

$$\sigma^2 = 2 \left(\frac{C}{\epsilon_0 n} \right)^2 \cdot \ln \frac{1.25}{\delta_0}$$

- By Advanced Composition for **adaptive** queries, overall algorithm is (ϵ, δ) -DP for:

$$\begin{aligned} \epsilon &= O \left(\epsilon_0 \cdot \sqrt{T \ln(2/\delta)} \right) \\ \delta &= 2T \cdot \delta_0 \end{aligned}$$

- Solving, suffices to use noise variance

$$\sigma^2 = O \left(\left(\frac{C}{\epsilon n} \right)^2 \cdot T \cdot \ln \frac{T}{\delta} \cdot \ln \frac{1}{\delta} \right)$$

Improved Analysis with “Concentrated DP”

[Dwork-Rothblum '16, Bun-Steinke '16]

- By Gaussian Mechanism, each iteration is ϵ_0^2 -zCDP if

$$\sigma^2 = \frac{1}{2} \left(\frac{C}{\epsilon_0 n} \right)^2 \cdot \ln \frac{1.25}{\delta_0}$$

- By composition of zCDP, overall algorithm is $T \cdot \epsilon_0^2$ -zCDP.
- By properties of zCDP, overall algorithm is (ϵ, δ) -DP for:

$$\epsilon = T \cdot \epsilon_0^2 + 2 \sqrt{T \cdot \epsilon_0^2 \cdot \ln(1/\delta)}$$

- Solving, suffices to use noise variance

$$\sigma^2 = O \left(\left(\frac{C}{\epsilon n} \right)^2 \cdot T \cdot \ln \frac{1}{\delta} \cdot \ln \frac{T}{\delta} \right)$$

DP **Stochastic** Gradient Descent (**S**GD)

[Jain-Kothari-Thakurta '12, Song-Chaudhuri-Sarwate '13, Bassily-Smith-Thakurta '14]

- Specify
 - Number of steps T , learning rate η , privacy parameters ε, δ , clipping parameter C .
 - Batch size $B \ll n$ (for efficiency)
 - Noise variance $\sigma^2 = \text{TBD}(T, \varepsilon, \delta, C, B)$.
- Pick initial point $\hat{\theta}_0$
- For $t = 1$ to T
 - Select a random batch $S_t \subseteq \{1, \dots, n\}$ of size B .
 - Estimate gradient as noisy average of clipped gradients
$$\hat{g}_t = \frac{1}{B} \sum_{i \in S_t} [\nabla \ell(\hat{\theta}_{t-1} | x_i, y_i)]_C + \mathcal{N}(0, \sigma^2 I)$$
 - $\hat{\theta}_t = \hat{\theta}_{t-1} - \eta \cdot \hat{g}_t$
- Output $\hat{\theta} = \sum_{t=1}^T \hat{\theta}_t$ or $\hat{\theta}_T$

DP SGD: Improved Privacy Analysis

[Bassily-Smith-Thakurta '14, Abadi-Chu-Goodfellow-McMahan-Mironov-Talwar-Zhang '17]

- **Privacy amplification by subsampling:**

If $S : \mathcal{X}^n \rightarrow \mathcal{X}^B$ outputs a random subset of p out of n rows and $M : \mathcal{X}^B \rightarrow \mathcal{Y}$ is (ϵ, δ) -DP, then

$M'(x) = M(S(x))$ is $(\ln(1 + (e^\epsilon - 1)p), p\delta)$ -DP.

– Keep S_t secret; Use their randomness



$p\epsilon$

- Poisson sampling: choosing each point independently with probability $p=B/n$.
- Choosing B points without replacement
- Choosing B points with replacement

DP SGD: Improved Privacy Analysis

[Bassily-Smith-Thakurta '14, Abadi-Chu-Goodfellow-McMahan-Mironov-Talwar-Zhang '17]

- **Privacy amplification by subsampling:**

If $S : \mathcal{X}^n \rightarrow \mathcal{X}^B$ outputs a random subset of pn out of n rows and $M : \mathcal{X}^B \rightarrow \mathcal{Y}$ is (ϵ, δ) -DP, then

$M'(x) = M(S(x))$ is $(\ln(1 + (e^\epsilon - 1)p), p\delta)$ -DP.

- Keep S_t secret; Use their randomness



$p\epsilon$

- We can take $p = B/n$.

- Unfortunately privacy amplification by subsampling does not hold for zCDP.
- But similar analysis can be recovered using the “moments accountant” [Abadi et al. '17] or “truncated zCDP” [Bun et al. '18].

DP SGD: Privacy Analysis

- By Gaussian Mechanism, each iteration is (ϵ_0, δ_0) -DP if

$$\sigma^2 = 2 \left(\frac{C}{\epsilon_0 n} \right)^2 \cdot \ln \frac{1.25}{\delta_0}$$

- Subsampling + Gaussian is (ϵ'_0, δ'_0) -DP

$$\epsilon'_0 = \frac{B}{n} \epsilon_0 \quad \delta'_0 = \frac{B}{n} \delta_0$$

- Advanced Composition over T iterations

$$\begin{aligned} \epsilon &= O \left(\epsilon'_0 \cdot \sqrt{T \ln(2/\delta)} \right) \\ \delta &= 2T \cdot \delta'_0 \end{aligned}$$

Moments
accountant
 $O(\frac{B}{n} \epsilon_0 \sqrt{T}, T\delta)$;
[WBK19,MTZ19]
better analysis;
[JUO20] attack

Main Idea: Privacy amplification by subsampling + Composition

Neural Networks & Privacy

- Choice of the model architecture
 - The Gaussian noise proportional to the square root of number of parameters.
- Hyperparameter tuning
 - If we run analyses on the training data with various hyperparameter settings, and choose the best one (any problems?)
 - Doing this privately (with additional cost in privacy)
 - Use public dataset (CIFAR-100 dataset for CIFAR-10 training)
- Still can be improved...
 - MNIST(99.8% baseline) 98.1% $(2.93, 10^{-5})$ – *DP*
 - CIFAR-10(99.7% baseline) 66.2% $(7.53, 10^{-5})$ – *DP*

Results from [PTS+ 20]

Differentially Private Empirical Risk Minimization

Supervised ML Output

Primary Goal (risk minimization):

- Find $\theta \in \Theta$ minimizing $L(\theta) = \mathbb{E}_{(x,y) \sim \mathcal{P}}[\ell(\theta|x, y)]$.
- Difficulty: \mathcal{P} unknown.

Subgoal 1 (empirical risk minimization (ERM)):

- Find $\theta \in \Theta$ minimizing $L(\theta|\vec{x}, \vec{y}) = \frac{1}{n} \sum_{i=1}^n \ell(\theta|x_i, y_i)$.
- Turns learning into optimization.
- Difficulty: overfitting*

Subgoal 2 (regularized ERM):

- Find $\theta \in \Theta$ minimizing $L(\theta|\vec{x}, \vec{y}) = \frac{1}{n} \sum_{i=1}^n \ell(\theta|x_i, y_i) + R(\theta)$.
- $R(\theta)$ typically penalizes “large” θ , can capture Bayesian prior.

*Fact: DP automatically helps prevent overfitting! [Dwork et al. '15]

Output Perturbation

[Chaudhuri-Monteleoni-Sarwate '11]

$$M(\vec{x}, \vec{y}) = \operatorname{argmin}_{\theta} \left(\frac{1}{n} \sum_{i=1}^n \ell(\theta | x_i, y_i) + R(\theta) \right) + \text{Noise}$$

Objective Perturbation

[Chaudhuri-Monteleoni-Sarwate '11]

$$M(\vec{x}, \vec{y}) = \operatorname{argmin}_{\theta} \left(\frac{1}{n} \sum_{i=1}^n \ell(\theta | x_i, y_i) + R(\theta) + R_{\text{priv}}(\theta, \text{noise}) \right)$$

Exponential Mechanism for ML

[Kasiwiswanathan-Lee-Nissim-Raskhodnikova-Smith '11]

Use utility function

$$u((\vec{x}, \vec{y}), \theta) = -L(\theta | \vec{x}, \vec{y}) = -\frac{1}{n} \sum_{i=1}^n \ell(\theta | x_i, y_i) - R(\theta).$$

That is,

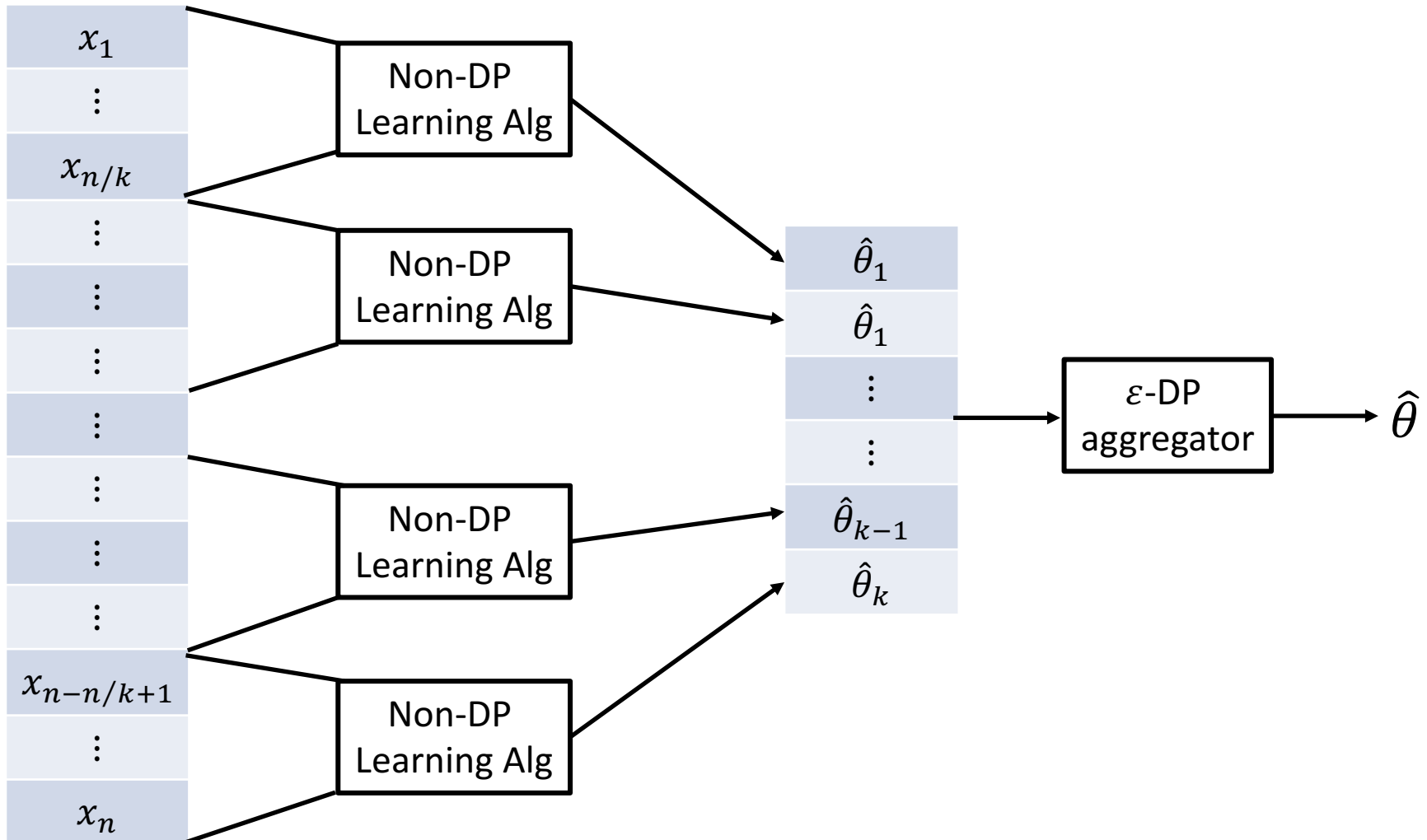
$$\Pr[M(\vec{x}, \vec{y}) = \theta] \propto e^{-\frac{\varepsilon}{2} \sum_{i=1}^n \ell(\theta | x_i, y_i) - \frac{\varepsilon n}{2} R(\theta)}.$$

Is ε -DP if the loss functions are clipped to $[0,1]$. (why?)

Thm [KLNRS '11, informally stated]: anything learnable non-privately on a finite data universe is also learnable with DP (with larger n).

Subsample & Aggregate

[Nissim-Rakhodnikova-Smith '07, Smith '11]



Q: Why is this ϵ -DP?

Subsample & Aggregate

[Nissim-Rakhodnikova-Smith '07, Smith '11]

- Typical aggregators: DP (clipped) mean, DP median
- **Benefits:**
 - Use any non-private estimator as a black box
 - Can give optimal asymptotic convergence rates: for many statistical estimators, variance is asymptotically $c_\theta / (\text{sample size})$, so variance of DP mean $\hat{\theta}$ is
$$(1/k) \cdot (c_\theta \cdot k/n) + O(1/\varepsilon k)^2 = (1 + o(1)) \cdot c_\theta / n$$
if $k = \omega(\sqrt{n})$.
- **Drawbacks:**
 - Dependence on dimension, model parameters, distribution can be bad.
 - Often takes very large sample size to kick in.
- **Develop:**
 - Private Aggregation of Teacher Ensembles (PATE) [PAE+17, PSM+18]

Modifying ML Algorithms

- **Another approach:** decompose existing ML/inference algorithms into steps that can be made DP, like Statistical Queries (estimating means of bounded functions)
- **Example:** linear regression
 - $S_{xx}/n, S_{xy}/n, \bar{x}, \bar{y}$ are all statistical queries