How to train private models?

There are 3 broad approaches for solving ERM with privacy guarantee:

1- **Output Perturbation:** Suppose we have an algorithm that approximates θ^*.

To make θ^* private, we can add noise to it

$$\theta^* + (N^1, \ldots, N^d)$$

Potential drawback : Sensitivity of θ^* might be large !

* In SVM, θ^* is close to a datapoint ("support vector")

so changing that datapoint might drastically change θ^* $\Rightarrow$ high sensitivity.

Potential solution : Appropriate regularization $\sum \ell(\theta, (x_i, y_i)) + \lambda \underbrace{F(\theta)}_{\text{regularizer}}$

Resulting θ^* might have reasonable sensitivity (under some restrictive assumption)

2- Objective perturbation: Instead of minimizing $\sum l(\theta, (x_i, y_i))$

we minimize

$$\sum l(\theta, (x_i, y_i)) + \langle \vec{N}, \theta \rangle \quad (*)$$

Gaussian random vector.

It was shown that the optimizer for this objective can be thought of as the private version of the optimizer of original ERM.

Drawback: All early works on objective perturbation required the optimization problem to find exact minimum for the objective function (*). this turns out to be very unrealistic as most popular optimizers are iterative;

that is they only approximate the optimum value.

* there are <u>several</u> recent works that relax this requirement, showing objective perturbation can be quite practical.

3. Gradient Perturbation: this is by far the most popular approach which is partially inspired by the fact that gradient-based iterative algorithms are ubiquitous in AI. this approach is best described in Gradient Descent Alg. (GD)

Non-Private GD :

$$:= \mathcal{L}(\theta, D) \quad \text{(dataset)}$$

$$\arg\min_{\theta \in \mathcal{C}} \sum_{i=1}^{n} \ell(\theta, (x_i, y_i))$$

where $\mathcal{C} \subset \mathbb{R}^d$ is a set where θ should live in.

Projected GD

Input : Dataset $D = \{(x_i, y_i)\}_{i=1}^{n}$, loss function ℓ, parameter set $\mathcal{C}$ & η_t

1- Pick $\theta_0 \in \mathcal{C}$ arbitrarily

2- For $t=1$ to T

- set $\theta_t = \Pi_{\mathcal{C}}\left(\theta_{t-1} - \eta_t \nabla \mathcal{L}(\theta_{t-1}, D)\right)$ ← Projection operator

Output $\theta_1, \theta_2, \dots \theta_T$

It is widely known that moving along the negative of gradient gets us closer to the minimum. Why?

You can formalize this observation using Taylor expansion.

To assess utility of this algorithm, we use this folklore theorem

Theorem. (Shamir & Zhang 2013, "SGD for non-smooth optimization")

Let F be a convex function & let $\theta^* = \arg\min_{\theta \in \mathcal{C}} F(\theta)$.

Let θ_0 be an arbitrary point in $\mathcal{C}$ &

$\theta_{t+1} = \Pi_{\mathcal{C}}(\theta_t - \eta_t G_t(\theta_t))$, where $E[G_t(\theta_t)] = \nabla F(\theta_t)$ &

$\mathbb{E}[\|G_t(\theta)\|_2^2] \le G^2$ & the learning rate $\eta_t = \frac{\|C\|_2}{G\sqrt{t}}$. ($C$: diameter of $\mathcal{C}$)

then for any $T > 0$, we have

$$\mathbb{E}[F(\theta_T)] - F(\theta^*) = O\left(\frac{\|C\|_2 \, G \log T}{\sqrt{T}}\right)$$

we can immediately apply this theorem, with $G_t(\theta_t) = \nabla \ell(\theta_t, D)$.

"which happens to be deterministic"

To ensure that $\mathbb{E}[\|G_t(\theta_t)\|_2^2] \le G^2$, we need to make a highly restrictive assumption:

Assumption: loss function $\ell(\theta, (x,y))$ is L-Lipschitz for all (x,y): that is

$$|\ell(\theta,(x,y)) - \ell(\theta',(x,y))| \le L \cdot \|\theta - \theta'\|_2$$

Note: f is L-Lip $\Rightarrow \|\nabla f\|_2 \le L$

thus, $\mathbb{E}\left[\|G_t(\theta_t)\|_2^2\right] = \mathbb{E}\left[\|\nabla \mathcal{L}(\theta_t, D)\|_2^2\right] \le (nL)^2$

Because $\nabla \mathcal{L}$ is nL-Lip.

Thus, for the GD algorithm, we have:

$$\underbrace{\mathcal{L}(\theta_T, D) - \mathcal{L}(\theta^*, D)}_{\text{Excess empirical risk}} \le \|\theta\|_2 \, n \cdot L \, \frac{\log T}{\sqrt{T}}$$

If we run this algorithm sufficiently large ($T \to \infty$), then $\mathcal{L}(\theta_T, D) \to \mathcal{L}(\theta^*, D)$.

sufficiently large: something like $T = \Omega(n^2)$

Drawback:

At each iteration, we need to take n gradient, so in total, nT gradients ($\approx n^3$ gradients).

* Is there anyway to reduce computation at each iteration?

perhaps only <u>one</u> gradient at each iteration? Yes!

Stochastic GD (SGD)

Projected SGD

Input: Dataset $D = \{(x_i, y_i)\}_{i=1}^{n}$, loss function ℓ, parameter set $\mathcal{C}$ & η_t

1- Pick $\theta_0 \in \mathcal{C}$ arbitrarily

2- For $t=1$ to T

- Select $I \in \{1, 2, \dots, n\}$ uniformly at random
- set $\theta_t = \prod_{\mathcal{C}} \left(\theta_{t-1} - \eta_t n \nabla \ell(\theta_{t-1}, (x_I, y_I)) \right)$

Output $\theta_1, \theta_2, \dots \theta_T$

claim: $\nabla \ell(\theta, (x_I, y_I))$ is a good estimate of $\nabla L(\theta, D)$.

note that

$$E\left[\nabla \ell(\theta, (x_I, y_I))\right] = \frac{1}{n} \sum_{i=1}^{n} \nabla \ell(\theta, (x_i, y_i)) = \frac{1}{n} \nabla L(\theta, D)$$

$$\Rightarrow \quad n\, E\left[\nabla \ell(\theta, (x_I, y_I))\right] = \nabla L(\theta, D)$$

what this means is that $\nabla \ell(\theta, (x_I, y_I))$ is an unbiased estimate of $\nabla L(\theta, D)$.

the computation is way less in SGD than in GD:

at each iteration, we only need <u>one</u> gradient, so in total we need only n^2 gradient, down from n^3 in GD.

What is the catch? Convergence is probabilistic!

Using the previous general purpose result (Shamir & Zhang 2013):

with $G_t(\theta) := n \nabla \ell(\theta_t, (x_I, y_I)) \longrightarrow E[G_t(\theta)] = \nabla R(\theta, D)$

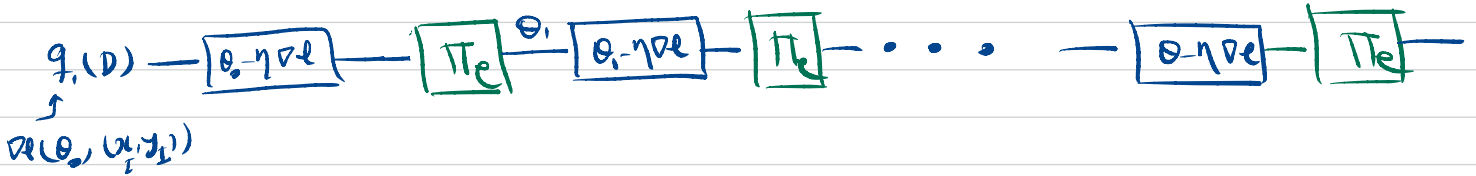
& $E[\|G_t(\theta)\|^2] \leq n^2 L^2$ again we assume loss is L-lip

so: $$E[R(\theta_T, D)] - R(\theta^*, D) \leq \|C\|_2 \, nL \frac{\log T}{\sqrt{T}}$$

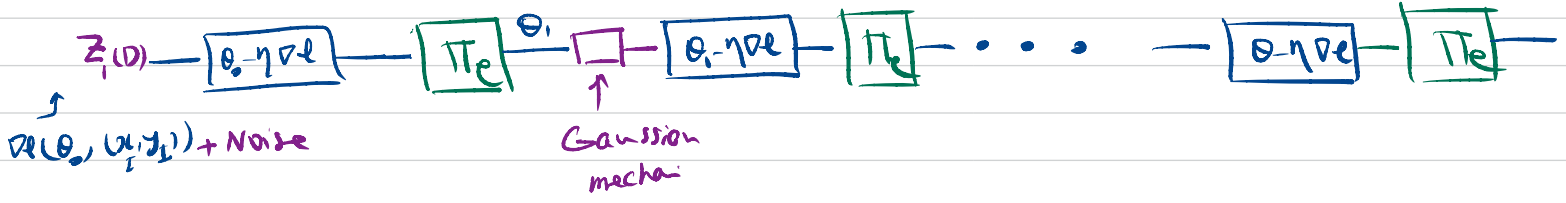
↔ same as GD but probabilistic convergence

Private Projected SGD (PN-SGD)

noisy



To make projected SGD, differentially private, we need to make each iteration private. To do so, we need to pass the query response through a DP mechanism, say Gaussian mechanism.



Thus, each iteration proceeds as follows:

Private Projected GD

Input: Dataset $D = \{(x_i, y_i)\}_{i=1}^{n}$, loss function ℓ, parameter set $\mathcal{C}$ & η_t

1- Pick $\theta_0 \in \mathcal{C}$ arbitrarily

2- For $t=1$ to T

- Select $I \in \{1, 2, \dots, n\}$ uniformly at random
- set $\theta_t = \prod_{\mathcal{C}} \left(\theta_{t-1} - \eta_t \left[n \nabla \ell(\theta_{t-1}, (x_I, y_I)) + (N_t^1, \dots, N_t^d) \right] \right)$

$N_t^i \overset{iid}{\sim} \mathcal{N}(0, \sigma^2)$

Output $\theta_1, \theta_2, \dots \theta_T$

At each iteration, query is $\nabla \ell(\theta, (x_I, y_I))$ which is an adaptive query as it depends on the previous iteration's output.

Each iteration becomes $(\varepsilon_0, \delta_0)$-DP with $\varepsilon_0 = \frac{2nL}{\sigma}\sqrt{2\log\frac{1}{\delta_0}}$

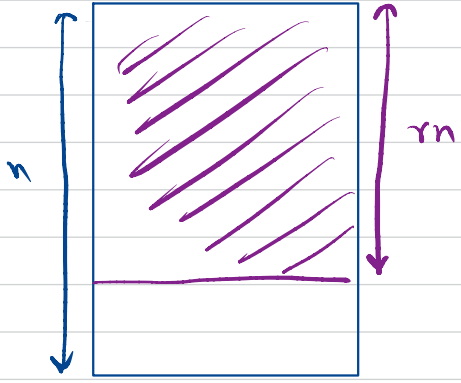
Note that: $\Delta_2^q \leq 2nL$ for the query $\nabla \ell(\theta, (x_I, y_I))$.

Therefore, advanced composition can be used to obtain the privacy guarantee. But, we can do better! At each iteration, we don't use the whole dataset! We only use <u>One</u> data record & <u>not</u> all the records. Thus, the privacy must be significantly better!

How to quantify this improvement:

* Privacy amplification by Sub_Sampling:

Let M be an (ε, δ)-DP mechanism. If it is run on a uniformly selected subset of dataset of size γn $(\gamma \leq 1)$, then it will provide $\left(\log(1+\gamma(e^{\varepsilon}-1)), \gamma\delta\right)$-DP.



$q(D)$ depends only the first part dataset — M → Z_D

Z_D provides more privacy compared to the case when $q(D)$ depends on the whole dataset.

In the SGD : $r = n$ (as the size of dataset = 1)

Gaussian mechanism coupled with this sub-sampling is typically called: Sub-sampled Gaussian mechanism.