Section 7: Machine Learning with DP and Ethics

CS 208 Applied Privacy for Data Science, Spring 2022

March 24, 2022

1 Agenda

- Discuss any questions about problem sets.
- Machine Learning with DP.
- Ethics discussion.

2 General Machine Learning

Machine Learning (ML) is a sub-field of computer science—and statistics and artificial intelligence—dedicated to using data to make predictions or decisions based on learned models. At the moment, the field seems to be dominated by advances in neural networks and deep learning! But it certainly is not the same as deep learning. ML has found applications in many disparate fields including computer vision, medicine, and astrophysics.

ML relies on using functions of computed statistics (e.g., gradients of loss functions that depend on data) to optimize a certain objective for predictions or decision-making. As a result, the naive release of ML models could lead to privacy violations. DP could help stem these potential privacy harms. You have already read the work on DP deep learning (Abadi et al., 2016). Using similar ideas, you can extend any ML procedure to satisfy DP with an inherent—and provable—loss in accuracy.

In the ML literature, it seems that there are 3 major subfields: supervised learning, unsupervised learning, and reinforcement learning. In supervised learning, the goal is to learn models or rules using input data that has labels (of which classes the examples/datapoints belong to). For unsupervised learning, the ML algorithm has to find structure from the input data without access to these labels. e.g., via clustering. As for reinforcement learning, the ML algorithm has to interact with a dynamic environment via a reward-feedback framework—which might produce actionable data. It is beyond the scope of this section to discuss these subfields in detail. We will focus on supervised learning methods for DP.

3 Supervised ML

3.1 Inputs

In general, supervised ML requires three inputs (some of which might be implicitly defined):

1. The first is data: $(x_1, y_1), \ldots, (x_n, y_n) \sim \mathcal{P}$, where $x_i \in \mathcal{X}$ is a d-dimensional vector of features taking discrete or continuous values, $y_i \in \mathcal{Y}$ is a value indicating classification or regression values for each feature vector, and where the distribution $\mathcal{P}$ is typically unknown. Note that

supervised learning refers to the data being labeled, as opposed to unsupervised learning where the algorithms must learn the inherent structure of unlabeled data.

2. The second is a family $\mathcal{M}$ of models $m_\theta : \mathcal{X} \to \mathcal{Y}$. The parameters $\theta \in \Theta$ are k-dimensional discrete or continuous values. For example, the model might be the following for linear regression:
$$m_\beta(x) = \langle \beta, x \rangle = \beta^T x,$$
where $x \in \mathbb{R}^k$, $\beta \in \mathbb{R}^k, k = d$. On the other hand, for deep neural networks, θ might be the vector of weights at all nodes.

3. The third input is a loss function $l : \Theta \times \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, mapping parameter, feature, and classification or regression triples to real values. Some common loss functions are:

   (a) Classification error:
   $$\ell(\theta|x, y) = I(m_\theta(x) \neq y).$$

   (b) Squared loss:
   $$\ell(\theta|x, y) = (m_\theta(x) - y)^2.$$

   (c) Neegative log likelihood:
   $$\ell(\theta|x, y) = -\log \Pr[m_\theta(x) = y] = \log\left(\frac{1}{\Pr[m_\theta(x) = y]}\right).$$

   You might wonder why it is common to use negative log likelihood for the loss function, rather than simply likelihood. First, optimizers typically minimize a function, and so to maximize the likelihood we have to minimize the negative likelihood. Second, applying the log makes the likelihoods easier to work with, as multiplications become sums and small likelihoods become larger negative values. Since the logarithm is a monotone function, optimizing the function is the same as optimizing the logarithm of it.

3.2 Goals

The primary goal of supervised ML is to find parameters that minimize the risk. Specifically, we want to find $\theta \in \Theta$ minimizing
$$L(\theta) = \mathbb{E}_{(x,y)\in\mathcal{P}}[\ell(\theta|x, y)].$$
However, we are unable to compute this quantity, much less minimize it, because P is unknown. Thus, we must resort to empirical risk minimization (ERM). Here, we use our dataset as a proxy for the underlying distribution and aim to find $\theta \in \Theta$ minimizing the average loss over the n data points:
$$L(\theta|\vec{x}, \vec{y}) = \frac{1}{n}\sum_{i=1}^{n} \ell(\theta|x_i, y_i).$$
Intuitively, empirical risk minimization turns learning into optimization. However, we still have a problem of overfitting to the data. In other words, we might minimize empirical risk for the given dataset, but this might not generalize to the probability distribution. In fact, DP prevents

this problem; it has been proven that anything learned in a differentially private way might, under certain conditions, generalize to the population from which the data was drawn.

But more generally, we can prevent overfitting by adding another source of noise: a regularization term. Now, we want to find $\theta \in \Theta$ minimizing

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

where $\vec{x} = (x_1, x_2, \ldots, x_n)^T$ and $\vec{y} = (y_1, y_2, \ldots, y_n)^T$.

The regularization term typically prevents large θs (which would imply too much complexity that is specific to the given dataset) and can capture a Bayesian prior.

Exercise 3.1. Suppose we wish to minimize the loss for the ordinary least squares (OLS) objective. What can we set the regularization to?

Solution. Let $X \in \mathbb{R}^{n\times k}$ be the feature matrix. i.e., rows of X are $x_i \in \mathcal{X} \subseteq \mathbb{R}^k$ for all $i \in [n]$. Let Y be the regression responses i.e., $Y = (y_1 \ldots y_n)^T$.

Then the OLS loss is

$$L(\theta \mid X, Y) = \frac{1}{n}\|Y - X\theta\|^2,$$

for any $\theta \in \mathbb{R}^k$. With regularization, the regularized OLS loss will be

$$L(\theta \mid X, Y) = \frac{1}{n}\|Y - X\theta\|_2^2 + \lambda\|\theta\|_2^2.$$

(This is called LASSO.)

Exercise 3.2. What is the gradient of the following convex functions?

1. $L(\theta \mid X, Y) = \theta^2$.
2. $L(\theta \mid X, Y) = \|Y - X\theta\|_2^2$.

It is known that Lipschitzness and smoothness of some convex functions help the optimization problem to converge faster to the optimum value. It is beyond the scope of this section to discuss this topic.

4 Approaches to ML with DP

4.1 Output Perturbation

As usual, the naive method is to simply add noise to the non-private mechanism.

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

However, this approach comes with several challenges. First, it is difficult to know how much noise to add. If the function is complex, we may not know how much one datapoint affects the parameter. Second, the global sensitivity of the optimal parameter might be very high. In the case of linear regression, we saw that the global sensitivity of the function that computes the slope/intercept is infinite!

There are cases in which we can reason about the global sensitivity; in particular, when the loss function l is strictly convex, has bounded gradient as a function of θ, and R is strongly convex. But in general, if there is no unique optimum, it is unlikely that we will have bounded sensitivity.

4.2 Objective Perturbation

Instead of adding noise after minimization, another approach is to add noise before minimization. This is done by adding a new randomized regularization term within the objective function.

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

Here, we are doing a random perturbation of the landscape, and we want the solution to the problem to be similar on neighboring databases. Proving privacy for objective perturbation requires similar conditions to those of output conditions, plus some extra second derivative assumptions about the loss function. However, objective perturbation might have better statistical performance than output perturbation, especially in high-dimensional cases.

4.3 Exponential Mechanism for ML

We could also try using the exponential mechanism, since it is a general template for producing a differentially private output. The utility function we choose is the negative loss function, since we want to minimize loss and maximize utility.

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

Thus, for every possible parameter θ, the mechanism outputs θ with probability proportional to

$$\Pr[M(x) = \theta] \propto \exp\left(-\frac{\epsilon n}{2\Delta} \big(\sum_{i=1}^{n} \ell(\theta | x_i, y_i) - R(\theta) \big) \right),$$

where Δ is the sensitivity of the loss function ℓ. This procedure, as we have seen in class, satisfies ϵ-DP.

The caveat to this approach is that there might not exist efficient implementation of the exponential-mechanism-based approach. In particular, if the parameter space is enormous, even when discretizing, the runtime of this mechanism can be exponential in the dimensionality of the parameter space. For example, if the parameter space is $\{0, 1\}^k$, and we discretize into 10 bins, we **might** have to calculate the loss for 2^k points.

Theorem 4.1 (KLNRS 11). *Anything learnable non-privately on a finite data universe is also learnable with DP (with larger n, which is at most logarithmic in the size of the data universe).*

4.4 Modifying ML Algorithms

The approach we will focus on is modifying ML algorithms themselves by decomposing them into steps that can be made differentially private. These steps will often be statistical queries, such as estimating the mean of bounded values. For example, in the case of linear regression, we will add Laplace noise to the sufficient statistics.

More relevant to ML, we will look at how to decompose and add privacy to one the most common ML algorithms, gradient descent.

5 Gradient Descent with DP

5.1 Non-private Gradient Descent

Gradient descent proceeds in steps. We start from carefully chosen initial parameters $\hat{\theta}_0$. $\hat{\theta}_0$ could be chosen randomly. Sometimes, the initial choice of $\hat{\theta}_0$ is very important. At each step, move in direction opposite to the gradient of the loss $\nabla L(\hat{\theta}|\vec{x}, \vec{y})$, where the gradient is the vector of partial derivatives. The next step θ_t is calculated as

$$\theta_t = \theta_{t-1} - \eta_t \cdot g_t,$$

where θ_{t-1} is the current position, η_t is the learning rate (or step size), and g_t is an estimate of the gradient of L at the current position.

We will consider gradient descent for convex loss functions.

Definition 5.1 (Convex functions)**.** L is convex if for all points $\vec{a}, \vec{b}$ and for all $t \in [0, 1]$, we have that

$$L(t\vec{a} + (1-t)\vec{b}) \leq tL(\vec{a}) + (1-t)L(\vec{b}).$$

For convex functions, there is only one local minimum which is equal to the global minimum.

Exercise 5.2. Why of the following functions are convex: $f(x) = x^2, f(x) = x^3, f(x) = \sqrt{x}, f(x) = \log x$?

Hint: In one dimension, a twice-differentiable function is convex if and only if its second derivative is nonnegative on the entire domain of the function.

The full algorithm for a generic template of gradient descent is below.

Algorithm 1 Gradient Descent

```
Specify
Number of steps $T$
Learning rate $\eta_t$
Pick initial point $\hat{\theta}_0$
for $t = 1$ to $T$ do
    Compute gradient
    $g_t = \frac{1}{n}\sum_i[\nabla l(\hat{\theta}_{t-1}|x_i, y_i)] + \nabla R(\hat{\theta}_{t-1})$
    $\hat{\theta}_t = \hat{\theta}_{t-1} - \eta_t \cdot g_t$
end for
```

Exercise 5.3. Write some python code for vanilla gradient descent on the OLS function. How many iterations of the gradient descent/ascent does it take to converge to a value that is close to the optimal?

5.2 DP for Vector-Valued Functions

Before diving into DP gradient descent, we will first look at two mechanisms for incorporating DP for vector-valued functions.

Let $f : \mathcal{X}^n \to \mathbb{R}^k$ and $M(x) = f(x) + Z$ for noise $Z \in \mathbb{R}^k$.

Definition 5.4 (Global $||\cdot||$ sensitivity)**.** For a norm $||\cdot||$ on $\mathbb{R}^k$, the global $||\cdot||$ sensitivity of f is

$$GS_{f,||\cdot||} \triangleq \max_{x\sim x'} ||f(x) - f(x')||.$$

The first mechanism is the $||\cdot||$-Norm Mechanism. This mechanism sets the density of the noise Z at z proportional to

$$e^{\epsilon||z||/(2GS_{f,||\cdot||})}.$$

This mechanism yields ϵ-DP for any norm $||\cdot||$. In addition, if we are using the ℓ_1 norm ($||z|| = \sum_j |z_j|$), this is equivalent to adding independent Laplace noise to each coordinate.

The second mechanism is the Gaussian Mechanism. Here, we sample the noise Z as

$$Z \sim \mathcal{N}\left(\vec{0}, 2\left(\frac{GS_{f,||\cdot||}}{\epsilon}\right)^2 \cdot \ln\frac{1.25}{\delta} \cdot I_k\right).$$

Claim 5.5. *When we are using the ℓ_2 norm ($||z|| = \sqrt{\sum_j |z_j|^2}$), the Gaussian mechanism for vector-valued functions is (ϵ, δ)-DP and is equivalent to adding independent Gaussian noise to each coordinate.*

We show this by reducing the privacy analysis of the Gaussian mechanism for vector-valued functions to the one-dimensional Gaussian mechanism. When applying the Gaussian mechanism to privately release $f(x)$, we obtain a spherical normal distribution centered around $f(x)$.

We can represent the spherical distribution of Z using an orthogonal basis $q_1, \ldots, q_k$ for $\mathbb{R}^k$.

$$Z = W_1 q_1 + \ldots + W_k q_k$$

If we choose one component q_i of the basis to be $f(x) - f(x')$ for neighboring x, x', then the two distributions differ by only the W_i corresponding to this component. By the theorem below, W_i is a one-dimensional Gaussian.

If we look at $f(x), f(x')$ for neighboring datasets x, x', we can choose an orthogonal basis such that we can represent the densities as a one-dimensional Gaussian.

Theorem 5.6. *If Z is a multivariate Gaussian, then $W_1, \ldots, W_k$ are independent zero-centered normally distributed random variables.*

Given that the one-dimensional Gaussian mechanism is (ϵ, δ)-DP, we see that the mechanism for vector-valued functions is (ϵ, δ)-DP.

5.3 DP Gradient Descent

Below, we describe the algorithm for DP gradient descent. The items in red are what are added for privacy.

Algorithm 2 DP Gradient Descent

```
**Specify**
Number of steps $T$
Learning rate $\eta_t$
Privacy parameters $\epsilon, \delta$
Clipping parameter $\Delta$
Noise variance $\sigma^2$
Pick initial point $\hat{\theta}_0$
**for** $t = 1$ to $T$ **do**
    Estimate gradient as noisy average of clipped gradients
    $\hat{g}_t = \frac{1}{n}\sum_i [\nabla l(\hat{\theta}_{t-1}|x_i, y_i)]_\nabla + \nabla R(\hat{\theta}_{t-1}) + \mathcal{N}(0, \sigma^2 I)$
    $\hat{\theta}_t = \hat{\theta}_{t-1} - \eta_t \cdot \hat{g}_t$
**end for**
```

Note that the variance σ^2 of the noise we need to add to the gradient at every iteration can be set using advanced composition for adaptive queries, and will depend on T, ϵ, δ, and Δ.

5.4 DP Stochastic Gradient Descent

Algorithm 3 DP Stochastic Gradient Descent

```
**Specify**
Number of steps $T$
Learning rate $\eta_t$
Privacy parameters $\epsilon, \delta$
Clipping parameter $\Delta$
Batch size $B << n$
Noise variance $\sigma^2$
Pick initial point $\hat{\theta}_0$
**for** $t = 1$ to $T$ **do**
    Select a random batch $S_t \subseteq \{1, \ldots, n\}$ of size $B$
    Estimate gradient as noisy average of clipped gradients
    $\hat{g}_t = \frac{1}{B}\sum_{i \in S_t} [\nabla l(\hat{\theta}_{t-1}|x_i, y_i)]_\nabla + \nabla R(\hat{\theta}_{t-1}) + \mathcal{N}(0, \sigma^2 I)$
    $\hat{\theta}_t = \hat{\theta}_{t-1} - \eta_t \cdot \hat{g}_t$
**end for**
```

Now, the variance σ^2 of the noise we need to add to the gradient at every iteration will additionally depend on B.

6 Continuing the Embedded EthiCS discussion

See slides here: `https://docs.google.com/presentation/d/1oy0k6pnjeJAJJn0PSbFPw5akwe-6jSjHnQMgfE3Rl`
`edit?usp=sharing`