Privacy II: Differential Privacy and Machine Learning

Lecturer: Kamalika Chaudhuri

May 4, 2020

A major application of differential privacy that we will next talk about is machine learning – how to design machine learning classifiers trained on sensitive data that still preserve differential privacy of the training data. We will look at two classes of solutions – one-shot and iterative, and will present algorithms for each class.

Observe that if we do not care about accuracy, or if we have enough data, then it is not too hard to build a private classifier. The main challenge in differentially privacy machine learning is to build one which has a good privacy-accuracy-sample-size tradeoff. A large body of work in this area has thus been devoted to finding algorithms which have increasingly better tradeoffs.

1 One Shot Solutions

One-shot solutions are those that add noise to preserve differential privacy only once during the training process, instead of multiple times. There are two major classes of private one-shot solutions that we will look at next – Output Perturbation and Objective Perturbation.

1.1 The Setup

We begin with describing the most common setup. We are given training data $(x_1, y_1), \ldots, (x_n, y_n)$, where $x_i \in \mathbb{R}^d$ and $y_i \in -1, 1$ and the goal is to fit a linear classifier $w \in \mathbb{R}^d$ through the data while still preserving differential privacy. Specifically, we would like to calculate a $\tilde{w}$ that is an ϵ-differentially private approximation to w^* which minimizes the following L_2-regularized loss:

$$w^* = \text{argmin}_w \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \ell(w, x_i, y_i), \tag{1}$$

where ℓ is of the form: $\ell(w, x_i, y_i) = l(y_i w^\top x_i)$ for a convex l. An example of such an l is the logistic loss $l(w) = \log(1 + e^{-y_i w^\top x_i})$ and the hinge loss $l(w) = \max(0, 1 - y_i w^\top x_i)$ – which gives us logistic regression and soft margin SVM respectively. We assume in addition that $\|x_i\|$ is bounded – in particular, it is ≤ 1. Without this assumption, a single very high norm data point can drastically change the solution to 1, which makes private and faithful solutions less likely.

1.2 Output Perturbation

The idea behind output perturbation is to apply the global sensitivity method directly to w^*, the minimizer of (1). For an arbitrary optimization problem, the global sensitivity can be quite large; however, certain properties of l (that hold for common machine learning problems) and

L_2-regularization allow for a good solution in this particular case. In fact, what we can show is something like this.

Theorem 1. *If $\|x_i\| \leq 1$, l is convex with $l' \leq 1$, then the L_2-Global Sensitivity of w^* is at most $\frac{2}{\lambda n}$.*

The conditions on l hold for logistic loss and hinge loss and hence we can use Lemma 1 the global sensitivity for logistic regression and the (soft-margin) support vector machine classifiers. This in turn allows us to use the multidimensional Laplace Mechanism to get an ϵ-differentially private (approximations to) logistic regression and support vector machines. Replacing the multidimensional Laplace with the the multidimensional Gaussian Mechanism gives a corresponding (ϵ, δ)-differentially private solution. The full algorithm for ϵ-differential privacy is provided in Algorithm 1.

Algorithm 1 Output Perturbation

Calculate w^* by solving the convex optimization problem (1).
Output $w^* + \frac{1}{\lambda n} Z$ where $p(Z = z) \propto e^{-\epsilon \|z\|}$.

Observe that we are essentially using the multidimensional Laplace mechanism with $S_2(f) = \frac{1}{\lambda n}$. This means that the magnitude of the noise vector Z is proportional to $\approx 1/\lambda n \epsilon$. So higher n (more data) means less noise, and so does higher ϵ (low privacy) – and the privacy-accuracy-sample-size tradeoff remains. Additionally higher λ (high regularization) also leads less noise; so less complex models are easier to make private. Another observation is that the magnitude of Z grows with d – so there is a dependence on dimension, which applies even for large margin data.

The advantage of output perturbation is that it is very simple and easy to implement – use a standard machine learning algorithm to get w^* and add noise to get privacy. The disadvantage is that its statistical efficiency can be suboptimal.

1.3 Objective Perturbation

The noise distribution induced by the output perturbation solution is spherically symmetric; noise is added uniformly in all directions. However, the high dimensional loss function in (1) is steeper in some directions than others, and perturbing w^* in those directions leads to higher loss and hence higher classification error.

Suppose we could perturb w^* privately in such a way that it gets perturbed less along the directions where the loss function is steep, and more along the flatter directions. This way, we would get lower loss overall – since the change along the steeper directions is lower. This is the main idea behind objective perturbation.

How do we accomplish this? In objective perturbation, this is done by directly adding a random linear term to the objective function in (1), and then solving the resulting optimization problem. Specifically, we now minimize the objective:

$$w^* = \text{argmin}_w \frac{\lambda}{2}\|w\|^2 + \frac{1}{n}\sum_{i=1}^{n} \ell(w, x_i, y_i) + \frac{1}{n} b^\top w, \tag{2}$$

where b is a random vector, and $\ell(w, x, y)$ is of the form $\ell(w, x, y) = l(yw^\top x)$ for convex l. To achieve ϵ-differential privacy, Z is drawn from the multidimensional Laplace mechanism with $S_2(f) =$

1; replacing the multidimensional Laplace by a multidimensional Gaussian in the usual manner gives an (ϵ, δ)-differenially private version. The following theorem establishes privacy properties of objective perturbation.

Theorem 2. *If $\|x_i\| \leq 1$ for all i, l is convex and differentiable, with $|l'(z)| \leq 1$ for all z, and $|l''(z)| \leq c$ for all z, then, the minimizer of (2) provides $\epsilon + 2\log\left(1 + \frac{c}{n\lambda}\right)$-differential privacy.*

Observe that there is an extra condition that the double derivative of l has to be bounded. This applies to the logistic loss with $c = 1/4$; while it does not directly apply to hinge loss, it does apply to Huber loss – a partially quadratic approximation to the hinge loss. We can thus use objective perturbation to get ϵ-differentially private logistic regression and Huber SVM.

The advantage of objective perturbation over output perturbation is its improved privacy-accuracy tradeoff; for the same privacy, one can get higher accuracy since solving the perturbed objective ensures that the solution is perturbed less in the high error, steeper directions. The disadvantage is that it is more complex and harder to implement – one has to now modify the optimization process, and cannot simply use an off-the-shelf learning algorithm.

2 Iterative Solutions

The one-shot solutions presented above apply to machine learning problems that involve linear classifiers and convex optimization. We next look at how to get differentially private solutions for more complex classifiers, such as neural networks.

The convex solutions discussed above do not apply in these cases. Unlike linear classification, the empirical loss function is non-convex and hence cannot be exactly minimized. When privacy is not a concern, the empirical loss is approximately minimized through an iterative process that converges to a local optima. In each iteration, we pick one or a small batch of examples, and take a step against the gradient on these examples. More specifically, for a batch B of size b, the iterative process looks like this:

$$\theta_{t+1} = \theta_t - \frac{\eta}{b} \cdot \sum_{i \in B} \nabla \ell(\theta_t, x_i, y_i)$$

where ℓ is the loss function, η is a learning rate and θ_t is the model at the t-th iteration.

2.1 Direct Method

The easiest way to make this training process differentially private is to directly execute each step privately. If the norm of the gradient $\nabla \ell$ at each step is bounded, then we can use the global sensitivity method at each step. If the gradient is not bounded, then we can clip to make it fit within bounds. Another trick we can use is large(r) mini-batches; if the batch-size b is relatively large, then the global sensitivity, which is proportional to $1/b$, is lower and the relative noise added per step is less. Finally, observe that if the batches are disjoint and if we are making a single pass over the data, then we can use what is called *parallel composition* – each minibatch gradient uses privacy budget ϵ, and the total risk remains ϵ due to the disjointedness.

Combining these tricks will give us a solution, but with rather suboptimal privacy-accuracy-data size tradeoffs. In the next two sections we look at how to get better tradeoffs – both algorithmically, as well as through tighter accounting of privacy losses. A great deal of research in this area in the last few years has focussed on this question.

2.2 Better Privacy via Subsampling

A way to improve the privacy-accuracy tradeoff is by using the following observation: running an ϵ-differentially private algorithm on a random subsample of a dataset results in better privacy than ϵ. In the privacy literature, this is called *privacy amplification by subsampling*. More specifically, we have the following theorem.

Theorem 3. *Let A be an ϵ-differentially private algorithm for $\epsilon < 1$, and let D be a dataset of size n. If D' is a subset of size γn chosen uniformly at random from D without replacement, then $A(D')$ is $2\gamma\epsilon$ differentially private.*

Observe that for $\gamma < 1/2$, subsampling gives a strict improvement on the privacy risk. We will not go into the details here, but a version of the theorem can also be proven for $\epsilon \geq 1$.

Theorem 3 automatically suggests the following algorithm. Run for a fixed number of iterations; in each iteration, randomly sample a mini-batch of size $b = \gamma n$, and execute a mini-batch gradient descent step with privacy budget ϵ. If the process is run for T iterations, then from Theorem 3 and linear composition property of differential privacy, the total privacy loss is $\frac{2bT\epsilon}{n}$, while noise added to each batch gradient is $\approx \frac{1}{b\epsilon}$. Setting the parameters T and b suitably will give us a better privacy-accuracy tradeoff than the direct method alone.

2.3 Better Composition via the Moments Accountant

In the previous analysis, we used linear composition of differential privacy risks. This raises a natural question: can we get a tighter bound on how the privacy budget changes upon composition? If so, this would allow us to run mini-batch gradient updates for a longer number of iterations at the same privacy cost, ultimately leading to a better privacy-accuracy tradeoff.

The Moments Accountant is such a method, that uses mechanism-specific properties to get a tighter composition analysis. Before we can get into the method, we need to define a key concept – the privacy loss random variable. Given two datasets D and D' and a randomized algorithm A, the privacy loss random variable $Z_{D,D'}$ is defined as follows:

$$Z_{A,D,D'} = \log \frac{p(A(D) = t)}{p(A(D') = t)}, \quad \text{with probability } p(A(D) = t) \tag{3}$$

Observe that key privacy properties of A are captured by the privacy loss random variable. For example, if A is ϵ-differentially private, then $Z_{A,D,D'} \in [-\epsilon, \epsilon]$ with probability 1. If A satisfies (ϵ, δ)-differential privacy, then $Z_{A,D,D'} \in [-\epsilon, \epsilon]$ with probability $\geq 1 - \delta$. The Moments Accountant method establishes the privacy risk incurred by running a sequence of algorithms $A_1, A_2, \ldots$ by reasoning about the moment generating functions of the corresponding privacy loss random variables.

More specifically, the problem setup is as follows. A sequence of T algorithms $A_1, A_2, \ldots, A_T$ is run on the same dataset D, and the goal is to analyze the privacy properties – namely ϵ and δ parameters – of releasing the resulting union $A(D) = (A_1(D), A_2(D), \ldots, A_T(D))$.

The Moments Accountant Method does this in three stages. First, it computes the step-wise *moments* of each algorithm A_T; next, it combines them together into a moment function for A, and finally, it uses this moment function to recover the (ϵ, δ)-privacy guarantees. Let us now look at each stage in order.

Step 1: Find Step-wise Moment Functions. For a scalar $s > 0$, the step-wise moment function of an algorithm A_t is defined as:

$$\alpha_{A_t}(s) = \sup_{D,D':\Delta(D,D')=1} \log \mathbb{E}[e^{sZ_{A_t,D,D'}}],$$

where $Z_{A_t,D,D'}$ is the privacy loss random variable for A_t as defined in (3). Observe that this is a function of s, and is also a property of the algorithm A_t. For example, it can be written down either in closed form or numerically calculated for a variety of common differentially private algorithms.

Step 2: Combine into Joint Moment Function. If independent randomness is used in the algorithms $A_1, \ldots, A_T$, then we can combine the step-wise moment functions into a moment-function for A by using the following inequality. For any s,

$$\alpha_A(s) \leq \sum_{t=1}^{T} \alpha_{A_t}(s) \tag{4}$$

Again, observe that this can be calculated either in closed form or numerically for a variety of existing algorithms.

Step 3: Privacy Parameters from Moment Functions. Finally, we can obtain privacy properties of A from the moment function α_A; we do this by finding, for a specific ϵ, the best δ value that can be obtained by solving the following equation:

$$\delta = \min_s \exp(\alpha_A(s) - s\epsilon) \tag{5}$$

Since s is a scalar, the minimization can be implemented by a grid search over s.

Example. As a simple example, suppose that for each t, A_t is a Gaussian mechanism that answers a query q_t of global sensitivity 1 by adding $\mathcal{N}(0,1)$ noise; that is: $A_t(D) = q_t(D) + \mathcal{N}(0,1)$. Simple calculation shows that the step-wise moment function of A_t is:

$$\alpha_{A_t}(s) = s(s+1)/2$$

Combining this across T time steps gives us the moment function for A, which is now:

$$\alpha_A(s) = Ts(s+1)/2$$

Given a specific ϵ, the lowest δ can now be found by solving:

$$\delta = \min_s \exp(Ts(s+1)/2 - s\epsilon)$$

In this case fortunately the minimizing s and the corresponding δ can be found in closed form.

3 Concluding Remarks

Overall, we have talked about a few methods for training machine learning classifiers with differential privacy. The differentially private machine learning literature is quite extensive by now, but several open directions remain.

The first major challenge is designing mechanisms that improve the privacy-accuracy-sample size tradeoff. For linear classifiers, the tradeoff is manageable especially for large datasets. But for more complicated neural network classifiers, there is still a fairly sizeable gap, closing which remains a challenge. An even bigger challenge is that training classifiers is one part of the entire machine learning pipeline – which includes other processes such as data cleaning, feature selection and parameter tuning. It is currently unclear how to combine all these components to do differentially private end-to-end machine learning with a good privacy-accuracy-sample size tradeoff. Solving this is the biggest challenge in private machine learning.

4 Bibliographic Notes

Output perturbation is due to [CMS11, RBHT09] and objective perturbation due to [CMS11]. The first use of SGD for private learning is due to [SCS13]. The analysis of SGD with subsampling is due to [BST14], and the Moments Accountant Method is due to [ACG+16].

References

[ACG+16] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 308–318, 2016.

[BST14] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 464–473. IEEE, 2014.

[CMS11] Kamalika Chaudhuri, Claire Monteleoni, and Anand D Sarwate. Differentially private empirical risk minimization. *Journal of Machine Learning Research*, 12(Mar):1069–1109, 2011.

[RBHT09] Benjamin IP Rubinstein, Peter L Bartlett, Ling Huang, and Nina Taft. Learning in a large function space: Privacy-preserving mechanisms for svm learning. *arXiv preprint arXiv:0911.5708*, 2009.

[SCS13] Shuang Song, Kamalika Chaudhuri, and Anand D Sarwate. Stochastic gradient descent with differentially private updates. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 245–248. IEEE, 2013.