CS839: Differential Privacy and Learning January 29, 2026

Lecture 4: DP Gradient Descent

Instructor: Gavin Brown *Scribe:* Porter Provan

1 Adaptive Composition

Adaptive composition means that privacy guaranties still hold true even when each new query is calculated based on the results of earlier queries. It describes how the total privacy loss adds up as multiple differentially private mechanisms are run on the same data. This is important because real systems often make decisions step-by-step instead of all at once. Without adaptive composition, privacy could be broken by carefully choosing later queries. In simple terms, adaptive composition determines whether differential privacy is sustained when the privacy of $F(X)$ depends on the output of $G(X)$.

1.1 Theorem - Adaptive Composition

Let $A_1, \ldots, A_k$ be randomized algorithms such that each A_i is ε_i-differentially private, even when its input may depend on the outputs of $A_1, \ldots, A_{i-1}$.

Then their adaptive composition

$$C(x) = (A_1(x), A_2(x, y_1), \ldots, A_k(x, y_1, \ldots, y_{k-1}))$$

is $\left(\sum_{i=1}^{k} \varepsilon_i\right)$-differentially private.

Let us now go through a proposed claim and its proof.

1.2 Claim - Adaptive Composition

Claim 1.1. *Suppose $A : \mathcal{X}^n \to \mathcal{Y}$ is an ε-differentially private algorithm. Suppose that for every $y \in \mathcal{Y}$, the algorithm $B(\cdot, y) : \mathcal{X}^n \to \mathcal{Z}$ is also ε-differentially private. Define the composed algorithm $C : \mathcal{X}^n \to \mathcal{Y} \times \mathcal{Z}$ by*

$$C(x) = (A(x), B(x, A(x))) .$$

Then C is (2ε)-differentially private.

1.3 Proof - Adaptive Composition

Proof. Fix two neighboring datasets $x \sim x'$ and fix an output (y, z).

$$L_C^{x \to x'}(y, z) = \log\left(\frac{\Pr[C(x) = (y, z)]}{\Pr[C(x') = (y, z)]}\right)$$

Since $C(x) = (A(x), B(x, A(x)))$, we write:

$$= \log\left(\frac{\Pr[A(x) = y \ \cap \ B(x, A(x)) = z]}{\Pr[A(x') = y \ \cap \ B(x', A(x')) = z]}\right)$$

Rewrite using conditional probability:

$$= \log\left(\frac{\Pr[A(x)=y]\cdot\Pr[B(x,A(x))=z \mid A(x)=y]}{\Pr[A(x')=y]\cdot\Pr[B(x',A(x'))=z \mid A(x')=y]}\right)$$

Split the log:

$$= \log\left(\frac{\Pr[A(x)=y]}{\Pr[A(x')=y]}\right) + \log\left(\frac{\Pr[B(x,A(x))=z \mid A(x)=y]}{\Pr[B(x',A(x'))=z \mid A(x')=y]}\right)$$

Since A is ε-differentially private,

$$\log\left(\frac{\Pr[A(x)=y]}{\Pr[A(x')=y]}\right) \leq \varepsilon$$

For every fixed y, $B(\cdot, y)$ is ε-differentially private. Since in both the numerator and denominator we condition on $A(\cdot) = y$, we have

$$\log\left(\frac{\Pr[B(x,A(x))=z \mid A(x)=y]}{\Pr[B(x',A(x'))=z \mid A(x')=y]}\right) = \log\left(\frac{\Pr[B(x,y)=z]}{\Pr[B(x',y)=z]}\right) \leq \varepsilon$$

Therefore,

$$L_C^{x\to x'}(y,z) \leq \varepsilon + \varepsilon = 2\varepsilon.$$

Hence, C is (2ε)-differentially private. □

2 Differentially Private Gradient Descent (DP-GD)

Differentially Private Gradient Descent (DP-GD) is an optimization algorithm designed to train machine learning models while preserving the privacy of the underlying dataset. The algorithm ensures that the contribution of any single data point to the model update is limited and obfuscated with noise, providing differential privacy guarantees. This is important in applications where sensitive data is involved, such as medical or financial datasets, preventing leakage of private information. DP-GD balances model utility and privacy by combining gradient clipping and carefully calibrated noise addition during training.

2.1 Setup

- **Dataset:** $X = (x_1, \ldots, x_n) \in \mathcal{X}^n$
- **Parameters/Weights:** $W \in \mathcal{W} \subseteq \mathbb{R}^d$
- **Loss function:**

$$L(W, X) = \frac{1}{n}\sum_{i=1}^{n} \ell(w_i, x_i)$$

2.2 Algorithm: DP-Gradient Descent

Input: $X \in \mathcal{X}^n$, noise scale $\lambda \geq 0$, step size η, number of iterations T, clipping threshold C, loss function ℓ, initial parameters W_0.

1. $w_0 \leftarrow 0$
2. **for** $t = 1, \ldots, T$:
3. $\quad \hat{g}_t \leftarrow \frac{1}{n} \sum_{i=1}^{n} \text{CLIP}_\gamma \left(\nabla_w \ell(w_{t-1}; x_i) \right)$
4. $\quad \tilde{g}_t \leftarrow \hat{g}_t + \mathcal{N}(0, \lambda^2 \mathbb{I}_d)$
5. $\quad w_t \leftarrow w_{t-1} - \eta \tilde{g}_t$
6. **Return** $(w_1, \ldots, w_T)$

Definition: $\text{CLIP}_\gamma(x)$ is

$$\min \left\{ \frac{\gamma}{\|x\|_2}, 1 \right\} \cdot x$$

2.3 Claim - Basic Composition

For $0 \leq \varepsilon \leq 1$, $0 < \delta \leq 1/2$, if

$$\lambda \geq \frac{2CT\sqrt{2 \log \frac{2T}{\delta}}}{\varepsilon n}, \text{ then DP-GD is } (\varepsilon, \delta)\text{-DP.}$$

2.4 Proof - Basic Composition

T steps, prove each is $(\varepsilon' = \frac{\varepsilon}{T}, \delta' = \frac{\delta}{T})$-DP. Then use basic composition.

$$\lambda = \frac{2C}{n} \cdot \frac{T}{\varepsilon} \sqrt{2 \log(2\frac{T}{\delta})} = \underbrace{\frac{2C}{n}}_{\Delta} \frac{\sqrt{2 \log(2\frac{T}{\delta'})}}{\varepsilon'}$$

2.5 Proof - Sensitivity Analysis

Our $f(x)$ is $\hat{g}_t$. What's global sensitivity of $\hat{g}_t$?

Fix $x \sim x'$, w.

$$\begin{aligned}
\left\| \hat{g}_t(w, x) - \hat{g}_t(w, x') \right\|_z &= \left\| \frac{1}{n} \sum_{i=1}^{n} \bar{g}_{x_i}(w) - \frac{1}{n} \sum_{i=1}^{n} \bar{g}_{x'_i}(w) \right\|_z \\
&= \left\| \frac{1}{n} (\bar{g}_{x_i}(w) - \bar{g}_{x'_i}(w)) \right\|_z \\
&\leq \frac{1}{n} \left(\|\bar{g}_{x_i}(w)\|_z + \|\bar{g}_{x'_i}(w)\|_z \right) \\
&= \frac{2C}{n}. \quad \boxtimes
\end{aligned}$$

3 General Notes & Discussion

- **Theoretical Foundation**
  - Most DP optimization relies on **DP-SGD**.
  - The core idea is to make the gradient descent process "algorithmically stable" so that the output model doesn't overfit to any specific user's data.
- **The Accuracy Trade-off**
  - **Bigger Clipping Threshold (C):** Captures more of the true gradient (lower bias), but increases the **Sensitivity**. High sensitivity requires adding more noise, which can drown out the learning signal.
  - **Small Clipping Threshold:** Reduces sensitivity (allowing for less noise), but introduces **Gradient Bias** because you are essentially lying about the true magnitude of the updates.
  - **Clipping Adaptivity:** A technique where C is tuned dynamically to find the "sweet spot" between noise and bias during training.

3.1 Clipping: Before or After Averaging?

Someone asked about what happens if we clip after taking the average, instead of before. This type of clipping is commonly used outside privacy applications to control large gradients. However, it will not really help our goals.

For simplicity, consider estimating the mean of data $x_1, \ldots, x_n \in \mathbb{R}$. If we believe each example will live between 0 and 1, then we can clip to $[0, 1]$ and add Laplace noise with scale $\mathrm{Lap}(1/\varepsilon n)$, or Gaussian noise with similar parameters. As n grows, this will result in a very small amount of noise.

If we took the average and then clipped, it would be most natural to force the average to also live in $[0, 1]$. But then a single data point could change the gradient estimate from 0 to 1, which means we would have to add noise $\mathrm{Lap}(1/\varepsilon)$. In this setting, getting more data doesn't seem to help!

If we wanted to add noise with scale $1/\varepsilon n$, we could try clipping the average to live in $[0, 1/n]$. This would force the global sensitivity to be low, but if the average gradient is actually close to 1 then it would introduce a lot of error.