HW 4B: Stochastic Gradient Descent and Lipschitz Extensions

CS 208 Applied Privacy for Data Science, Spring 2019

Version 1.0: Due Tuesday, April 30, 11:59pm.

Instructions: Submit a single PDF file containing your solutions, plots, and analyses. Make sure to thoroughly explain your process and results for each problem. Also include your documented code and a link to a public repository with your code (such as GitHub/GitLab). Make sure to list all collaborators and references.

1. For each of the following sets $\mathcal{G}$ of datasets and neighbor relations $\sim$, hypotheses $\mathcal{H} \subseteq \mathcal{G}$, and functions $f : \mathcal{G} \to \mathbb{R}$, calculate (i) the global sensitivity of f (denoted GS_f or ∂f), (ii) the minimum local sensitivity of f, i.e. $\min_{x \in \mathcal{G}} \mathrm{LS}_f(x)$, and (iii) the restricted sensitivity of f (denoted $\partial_{\mathcal{H}} f$ or $\mathrm{RS}_f^{\mathcal{H}}$). For Part 1a, also describe an explicit Lipschitz extension of f from $\mathcal{H}$ to all of $\mathcal{G}$.

   (a) $\mathcal{G} = \mathbb{R}^n$ where $x \sim x'$ if x and x' differ on one row, $\mathcal{H} = [a,b]^n$ for real numbers $a \leq b$, and $f(x) = (1/n)\sum_{i=1}^n x_i$.

   (b) $\mathcal{G} = \mathbb{R}^n$ where $x \sim x'$ if x and x' differ on one row, $\mathcal{H} = [a,b]^n$ for real numbers $a \leq b$, and $f(x) = \mathrm{median}(x_1, \ldots, x_n)$.

   (c) $\mathcal{G} =$ the set of undirected graphs (without self-loops) on vertex set $\{1, \ldots, n\}$ where $x \sim x'$ if x and x' are identical except for the neighborhood of a single vertex (i.e. node privacy), $\mathcal{H}$ the set of graphs in $\mathcal{G}$ in which every vertex has degree at most d for a parmeter $2 \leq d \neq n-1$, and $f(x) =$ the number of isolated (i.e. degree 0) vertices in x.

2. Recall for data universe $\mathcal{X} = \{x \in \mathbb{R}^k : \|x\| \leq R\}$ and $\varepsilon < 1$, the following is a (ε, δ)-DP local randomizer:

$$Q_{\text{gauss}}(x) = x + \mathcal{N}\left(0, \left(\frac{R}{2\varepsilon^2}\right)^2 \cdot \ln\left(\frac{1.25}{\delta}\right) \cdot I_k\right).$$

   We mentioned in class that there is also a pure DP randomizer that can be used instead. It works as follows.

   $Q_{\text{pure}}(x)$ for $x \in \mathcal{X}$:

   i. Choose a uniformly random unit vector $u \in \{v \in \mathbb{R}^k : \|v\| = 1\}$.

   ii. Do randomized response on $\langle u, x \rangle / R \in [-1, 1]$ to obtain a binary value $b \in \{\pm 1\}$.

   iii. Output bu

   Analyze Q_{pure} as follows.

   (a) Prove that Q_{pure} is ε-DP.

   (b) Argue that for every $x \in \mathcal{X}$, we have

$$\mathrm{E}[Q_{\text{pure}}(x)] = \frac{e^\varepsilon - 1}{e^\varepsilon + 1} \cdot \frac{c_k}{R} \cdot x,$$

for a constant c_k that depends only on the dimension k. (Hint: decompose $u = u^{\|} + u^{\perp}$ where $u^{\|}$ is a scalar multiple of x and $u^{\perp}$ is orthogonal to x.) You don't need to calculate the constant c_k (but it happens to equal $2s_{k-1}/((k-1)\cdot s_k)$, where s_d denotes the $(d-1)$-dimensional volume of the unit sphere $\{v \in \mathbb{R}^d : \|v\| = 1\}$ in d dimensions).

(c) Thus, like $Q_{\text{gauss}}(x)$, the post-processed output

$$Q'_{\text{pure}}(x) = \frac{e^{\varepsilon}+1}{e^{\varepsilon}-1} \cdot \frac{R}{c_k} \cdot Q_{\text{pure}}(x)$$

is an unbiased estimator of x. Ignoring constant factors, assuming $\varepsilon \leq 1$, and using the fact that $c_k = \Theta(1/\sqrt{k})$, compare the maximum expected squared error $\max_{x\in\mathcal{X}} \mathrm{E}[\|Q(x) - x\|^2]$ for $Q = Q'_{\text{pure}}$ vs. $Q = Q_{\text{gauss}}$.

3. In our code example,[1] we saw how to release an estimated Logistic regression using differentially private stochastic gradient descent (DP-SGD) to optimize a regression under the centralized model. Convert this code to once again release the probability of marriage given education level, but using DP-SGD under the *local* model. Recall that local DP does not satisfy privacy amplification by subsampling, but you can achieve a similar effect by rotating through disjoint batches, so that each individual partipates in at most $\lceil T \cdot B/n \rceil$ batches, where T is the number of iterations and B is the batch size.[2] Using Gaussian noise as in the current code will give an (ε, δ) local DP implementation. Also construct a pure $(\varepsilon, 0)$ local DP implementation using Problem 2, and compare the performance of the two implementations.

[1] See `https://github.com/privacytoolsproject/cs208/blob/master/examples/wk7_localmodel/privateSGD.r` and `privateSGD.ipynb`.

[2] Note, in the code example, $T = 1$, $B = \sqrt{n}$.