Privacy in Statistics and Machine Learning — Spring 2023
Homework 1: Due Sunday, February 26, 2023

Adam Smith (based on materials developed with Jonathan Ullman)

Collaboration and Honesty Policy Reminder: Collaboration in the form of discussion is allowed. However, all forms of cheating (copying parts of a classmate's assignment, plagiarism from papers or old posted solutions) are NOT allowed. A rough rule of thumb: you should be able to walk away from a discussion of a homework problem with no notes at all and write your solution on your own. Finding answers to problems on the Web or from other outside sources (these include anyone not enrolled in the class) is forbidden.

- *You must write up each problem solution by yourself without assistance, even if you collaborate with others to solve the problem.*
- You must identify your collaborators. If you did not work with anyone, you should write "Collaborators: none."
- Asking and answering questions in every forum the class provides (on Piazza, in class, and in office hours) is encouraged!
- Even though look up answers is forbidden, using the web to find alternative explanations of concepts you need for the homework *is* allowed, and encouraged. For example, you can look up background on probability and linear algebra, documentation for particular programming languages, etc.

Problems to be handed in

1. (**Node sensitivity**) Imagine we have a data set that comes from a simple social network with n people. Each node in the graph is a person. For each person, we have the following data: a unique identifier id, their income $a_{id} \in [0, 1]$ and their friend list F_{id}, which is a list of identifiers of other nodes. F_{id} can have any size—it can be empty, or consist of *all* othe nodes, or anything in between. We assume that friendship is symmetric: if Alice is on Bob's friend list, then Bob is on Alice's.

   We'll say two graphs are neighbors if they differ by changing the data for one node, including a_{id} and the list of edges connected to that node. Notice that changing one person's data can potentially affect everyone's else list of friends. Because of that, natural things we would like to compute can have high sensitivity. For example, the number of connected components in the graph can go from n (the current number of nodes) to 1 by changing the friend list of a single node.

   (a) As a function of n, what is the global sensitivity of each of the following functions? Give the best upper bounds and lower bounds that you can. (It should be possible to given an exact answer, like $n^2 - 1$), for each of these.)

   How does the sensitivity compare to the range of the function (that is, the difference between the largest and smallest possible values it can return)?

   i. the *number of edges* in the graph
   ii. the *number of triangles* in the graph
   iii. the *diameter of the graph* (this is the largest possible length of the shortest path betweent two nodes; we define the diameter of a disconnected graph to be the largest diamter of any of its connected components).

iv. *Distance from bipartiteness*: this is the smallest number of nodes that must be changed for the graph to be bipartite.

(b) *Income correlation*: How correlated are friends' incomes? Let μ be the average income in the graph $\mu = \frac{1}{n}\sum_{id} a_{id}$. The income correlation is

$$g(x) = \frac{1}{2\#(edges)} \sum_{id} \sum_{id' \in F_{id}} (a_{id} - \mu)(a_{id'} - \mu)\,.$$

(We multiply the number of edges by 2 since each edge gets counted twice in the formula.)

This quantity ranges between 0 and 1. Design a differentially private algorithm which, when $\varepsilon = 1$, with probability at least 0.9 approximates g with additive error $\pm O(1/n)$ on all graphs for which $\#(edges) \geq n^2/20$. (The constants 1 and 20 are arbitrary. In fact, one can get vanishing relative error under much weaker conditions.)

2. (In-class Exercise 3 from Lecture 3) **More accurate reconstruction with more random queries**

In this question we'll explore how to interpolate between the two reconstruction theorems we've seen. Specifically, we will prove a version of Theorem 2.5 that gives a more accurate reconstruction when we have $k \gg n$ queries. Suppose we have the following version of Claim 2.6 from the lecture notes:

Claim 0.1. *Let $t \in \{-1, 0, +1\}^n$ be a vector with at least m non-zero entries and let $u \in \{0,1\}^n$ be a uniformly random vector. Then for every parameter $2 \leq w \ll 2^m$*

$$\mathbb{P}\left(|u \cdot t| \geq \frac{\sqrt{m \log w}}{10}\right) \geq \frac{1}{w} \tag{1}$$

Using this claim, prove the following theorem

Theorem 0.2. *If we ask $n^2 \ll k \ll 2^n$ queries, and all queries have error at most αn, then with extremely high probability, the reconstruction error is at most $O(\frac{\alpha^2 n^2}{\log(k/n)})$.*

How does this theorem compare to the reconstruction attacks we've seen for $k \approx n^2$? What about $k \approx 2^{\sqrt{n}}$? What about $k \approx 2^n$?

3. (In-class exercise 6 from lecture 4) **Differential Privacy and Reconstruction Attacks**

Suppose A is an ε-differentially private algorithm that takes input $\mathbf{x} = (x_1, x_2, ..., x_n) \in \{0,1\}^n$ (so each person's secret information is just one bit). Consider an algorithm B that attempts to reconstruct the input from A's output: on input $A(\mathbf{x})$, it outputs a guess $\tilde{\mathbf{x}}$. Show that, for every algorithm B: if $\mathbf{x}$ is selected uniformly at random from $\{0,1\}^n$, and the algorithm B has access only to the output of A (nothing else), then

$$\mathop{\mathbb{E}}_{\substack{\mathbf{x} \in_r \{0,1\}^n \\ \tilde{\mathbf{x}} = B(A(\mathbf{x}))}} (\#\ \text{errors}(\tilde{\mathbf{x}}, \mathbf{x})) \geq \frac{n}{e^\varepsilon + 1}$$

Here, $\#\ \text{errors}(y, x)$ denotes the number of positions in which two vectors disagree (also called the Hamming distance). [1]

[1] In other words: when ε is small, differentially private algorithms do not allow for non-trivial reconstruction attacks. Even with no output at all, an attacker can always guess about $\frac{n}{2}$ of the bits of $\mathbf{x}$ in expectation (for example, by guessing the all-zeros string). The result above says that a attack based on differentially private output cannot do much better in expectation.

Hints: Use linearity of expectation. The number of errors can be written as a sum of random variables E_i (for $i = 1$ to n), where

$$E_i = \begin{cases} 1 & \text{if } \tilde{\mathbf{x}}_i = x_i\,, \\ 0 & \text{otherwise.} \end{cases}$$

What can you say about the conditional distribution of x_i given a particular output $A(\mathbf{x}) = a$? How big or small can $\Pr(x_i = 1 | A(\mathbf{x}) = a)$ be? Given that, what is the largest possible probability that $E_i = 1$? What does that tell you about E_i's expected value? It might be helpful to think about what happens when A is the randomized response mechanism, though your final proof should apply to any ε-DP algorithm.

4. **Implementing the Median Algorithm.** Implement the report-noisy-max version of the median algorithm from Lecture 6's In-class Exercises.

   Your code should take a set x of real values as input along with parameters R and ε. The first step should be to round all entries to the nearest integer in $\{1, ..., R\}$. (Values less than 1 should be rounded up to 1; values greater than R get rounded down to R.) The output should be a single integer.

   Test your code on data drawn from the following distributions, using $\varepsilon = 0.1$ and $n \in \{50, 100, 500, 2000, 10000\}$:

   - Gaussian: $\mathcal{N}(R/4, R^2/10)$ for $R \in \left\{2^7, 2^{10}, 2^{16}\right\}$.
   - Poisson: $Poi(50)$ for $R \in \left\{2^7, 2^{10}, 2^{16}\right\}$. (NB: The distribution does not change as the range increases.)
   - Bimodal: $R = 2^{10}$; each data point uniform over two values $\left\{\frac{R}{2} - k, \frac{R}{2} + k\right\}$ for $k = 10, 100, 200$.

   For each setting of parameters (data distribution, R and n), sample 50 data sets from the distribution, and run the algorithm 10 times on each. Collect the following statistics:

   - Average error in rank: how far from $n/2$ is the rank of the output (on average over all 500 runs)?
   - Standard deviation of error in rank (over all 500 runs): how much does this error vary from run to run?
   - Average over data sets of the standard deviation of error in rank *among runs on that data set.* (This tells you whether different data sets from the same distribution have different distributions on the error.)