Privacy in Statistics and Machine Learning Spring 2023
In-class Exercises for Lecture 2 (Reconstruction Part 1)
January 24, 2023

Adam Smith

Problems with marked with an asterisk (*) are more challenging or open-ended.

1. Consider the setting of Section 2.3 of the notes (Reconstruction from Many Queries) and the attack of Figure 4. Suppose the data set has size $n = 2,000$ and the attacker receives as input approximate answers to all possible linear queries on a secret vector s, each with error (at most) ± 100. What upper bound does Theorem 2.4 give on the attack's reconstruction error?

2. **(Baselines)** Suppose an attacker does not get access to the released statistics. They know only that $s \in \{0,1\}^n$ is uniformly random.

   (a) What is the *expected* error $\|\tilde{s} - s\|_1$ of every reconstruction attack that guesses a vector $\tilde{s} \in \{0,1\}^n$? Here $\|\tilde{s} - s\|_1$ is the number of bits in which $\tilde{s}$ and s differ.[1] This expected error gives us a *baseline* to evaluate when an attack that does use the released statistics is "interesting".

   (b) Suppose, as a different baseline, we want to understand the probability that an attacker without access to released statistic can guess $\tilde{s}$ such that $\|\tilde{s} - s\|_1 \leq n/4$. How could you argue that this probability is small? (Relevant tools include the Chebyshev inequality and Chernoff bounds.)

   In fact, the probability is exponentially small in n (that is, at most 2^{-cn} for a constant $c > 0$)? Prove using a Chernoff bound.

   (In general, finding a good "baseline" for evaluating reconstruction attacks is tricky.)

3. (Optimality of the attack in Section 2.3.)

   (a) Show that the guarantee of Theorem 2.4 is essentially tight. Specifically: Give an algorithm that takes as input a data set with a secret vector s of bits and an error rate α, and produces answers to all possible linear queries so that with (i) each approximate answer is within αn of the correct one, and (ii) the attack of Theorem 2.4 returns a vector $\tilde{s}$ with reconstruction error at least αn.

   (b) **(*)** Can you modify your procedure so it guarantees that *no attack algorithm* always returns a vector $\tilde{s}$ with reconstruction error αn? What if the algorithm just has to work with high probability (say 0.95)? What additional assumptions does your result require? (For example, you might have to make assumptions about the distribution of s, and what the attack algorithm knows about it.)

(See next page.)

[1] More generally, $\|\cdot\|_1$ denotes the sum of the absolute values of the entries of a vector.

4. Now let's consider a slightly different setting, in which the attacker gets approximate answers to a much smaller number of queries.

   Instead of all 2^n queries on $s \in \{0,1\}^n$, the attacker receives approximate answers only to the n prefix sums of the form $\sum_{j=1}^{i} s_j$ (for i from 1 to n). These correspond to query vectors

$$F_i = (\underbrace{1, 1, \ldots, 1}_{i \text{ ones}}, \underbrace{0, 0, \ldots, 0}_{n-i \text{ zeros}})$$

   (a) Suppose that n is even (for simplicity) and s consists of alternating 0's and 1's, that is $s = (0101 \cdots 01)$. Show how you could give a sequence of answers $a_1, \ldots, a_n$ such that (i) each prefix sum query is answered to within 1, that is,

$$|F_i \cdot s - a_i| \leq 1 \quad \text{for all } i = 1, \ldots, n\,,$$

   and (ii) the algorithm of Figure 4 would reconstruct a vector $\tilde{s}$ that is wrong in all n positions (that is, $\tilde{s}$ differs from s in every entry.

   (b) Try to generalize this as follows: suppose that s is uniformly random in $\{0,1\}^n$. Give a procedure that takes s as input and returns a sequence of answers $a_1, \ldots, a_n$ such that (i) each prefix sum query is answered to within 1, that is,

$$|F_i \cdot s - a_i| \leq 1 \quad \text{for all } i = 1, \ldots, n\,,$$

   and (ii) the algorithm of Figure 4 would reconstruct a vector $\tilde{s}$ whose expected distance from s is $\Omega(n)$. (Here the expectation is taken over the choice of s; the attack of Figure 4 is deterministic and your algorithm can also be.)

   (c) **(*)** Can you come up with a version of this result that works against every attack algorithm (with high probability over the choice of s and any random choices made by your algorithm and the attack)?