CS 860: Algorithms for Private Data Analysis Fall 2020

Lecture 10 — Beyond Global Sensitivity

Prof. Gautam Kamath *Scribe: Gautam Kamath*

Up to this point, we have focused on adding noise to a statistic calibrated to its global sensitivity. That is, we consider the maximum sensitivity over all possible neighbouring datasets: $\Delta = \max_{X,X'} \|f(X) - f(X')\|$, where X and X' are neighbouring databases. However, it is tempting to think this is overkill: after all, we're looking at a specific database X, can't we just add noise calibrated to the sensitivity around X? To investigate this idea, we define the local sensitivity.

Definition 1. *The local sensitivity of a function $f : \mathcal{X}^n \to \mathbb{R}$ on a database $X \in \mathcal{X}^n$ is*

$$\Delta_{LS}^{(f)}(X) = \max_{X'} |f(X) - f(X')|,$$

where X' is a neighbouring database of X.

When f is clear from context, we will omit it. For the sake of this lecture, we will only consider univariate functions.

With this definition in mind, the natural approach would be to add Laplace noise calibrated to the *local* sensitivity, rather than the global sensitivity as we have been doing. However, the issue is that the magnitude of the noise may reveal information about the dataset. Let's see this illustrated through an example. Consider the function f which computes the distance between the closest two points in a dataset, where the domain is the reals. Let the dataset $X = \{0, 0, 0\}$. It is clear that the local sensitivity of f on X is 0: moving any single point will leave two points at 0, and thus the value of the function will not change. Thus adding noise calibrated to the local sensitivity would just release 0 deterministically.

On the other hand, examine the neighbouring dataset $Y = \{0, 0, 10000\}$. The local sensitivity of f on Y is now 10000. If we add Laplace noise of order 10000 to $f(Y)$, it would be significantly different from the distribution obtained by adding Laplace noise of order 0 to $f(X)$, thus allowing us to distinguish between X and Y.

While this naïve approach failed, the idea of exploiting the local sensitivity still has merit. Today, we explore this through a few different appproaches.

Propose-Test-Release

The issue with the previous example was that, while the local sensitivity on the dataset X was low, the local sensitivity on a nearby dataset was high. Instead, we can (privately) check the distance to the nearest dataset with high sensitivity. If it is distant, then it is will be private to add a small amount of noise. If it is close to a dataset with high sensitivity, then the algorithm can "give up" – it may return $\perp$, or it may noise according to the global sensitivity, if desired.

The name of this approach is Propose-Test-Release [DL09]. It consists of the following three steps:

1. *Propose* an upper bound for the local sensitivity,

2. *Test* whether this is a valid upper bound (privately),
3. *Release* the value (privately) if this is the case.

In slightly more detail, it can be described as follows. Let f be the function of interest, which we wish to privately evaluate on dataset X.

1. Propose a bound β on the local sensitivity.
2. Compute the distance from X to the nearest dataset X' such that $\Delta_{LS}(X') \geq \beta$, name it γ. Distance is measured in terms of how many points must be changed to get from X to X'.
3. Compute $\hat{\gamma} = \gamma + \text{Laplace}(1/\varepsilon)$.
4. If $\hat{\gamma} \leq \ln(1/\delta)/\varepsilon$, return $\perp$.
5. If $\hat{\gamma} > \ln(1/\delta)/\varepsilon$, return $f(X) + \text{Laplace}(\beta/\varepsilon)$.

Note that the second step may not be computationally efficient. Indeed, we would naïvely have to iterate over all possible datasets in the domain. Nonetheless, you should try convincing yourself that it isn't hard to compute in certain nice cases, such as the median. Consider pausing right now and showing this as an exercise.

Another thing to note about the second step: when the value of our function is the number of points that much be changed, this automatically makes it have sensitivity 1. This is what allows us to privatize the value of γ with Laplace$(1/\varepsilon)$ noise in line 3. This trick is useful for making statistics have bounded sensitivity, and appears in other applications as well.

We will argue that this algorithm is $(2\varepsilon, \delta)$-differentially private. This can intuitively be seen by breaking the algorithm into two parts. The first part can be seen as privately answering the following question about X: is the local sensitivity low? While we did this before, we cheated by peeking at the true local sensitivity. To answer this privately, we have to in fact say that the local sensitivity is low for all nearby datasets. The step is done using ε of the privacy budget. If the answer is yes, the second part uses this fact to add a low amount of noise to the statistic – this takes the other ε of the privacy budget, but the other δ is reserved for if our bound in the first part was incorrect. We proceed with the proof in more formality.

Theorem 2. *Propose-Test-Release is $(2\varepsilon, \delta)$-differentially private.*

Proof. First, we consider the probability of outputting $\perp$ under neighbouring databases. This is done deterministically depending on whether or not $\hat{\gamma}$ passes some threshold. Since neighbouring databases will result in a value of γ which differs by at most 1, the Laplace mechanism implies that $\Pr[M(X) = \perp] \in [e^{-\varepsilon}, e^{\varepsilon}] \cdot \Pr[M(X') = \perp]$.

Now we break the analysis into two cases – in particular, we separately consider databases X depending on the local sensitivity. First, we consider the case when $\Delta_{LS}(X) > \beta$. In this case, $\gamma = 0$, and using the PDF of the Laplace distribution, the probability that $\hat{\gamma}$ is greater than $\log(1/\delta)/\varepsilon$ is at most δ. With this, it is not hard to prove the desired bound, for any $T \subseteq \mathbb{R} \cup \perp$:

$$\begin{aligned}
\Pr[M(X) \in T] &= \Pr[M(X) \in T \cap \{\perp\}] + \Pr[M(X) \in T \cap \mathbb{R}] \\
&\leq e^{\varepsilon} \Pr[M(X') \in T \cap \{\perp\}] + \Pr[M(X) \neq \perp] \\
&\leq e^{\varepsilon} \Pr[M(X') \in T] + \delta.
\end{aligned}$$

To see the first inequality, note that $T \cap \{\perp\}$ is either equal to $\{\perp\}$ or the empty set – in the former case we can use the bound above, and in the latter, the inequality vacuously holds. This implies the desired privacy guarantee.

Next, let's consider when $\Delta_{LS}(X) \leq \beta$. We view this as the composition of two differentially private algorithms. The first one is the release of $\hat{\gamma}$, which is $(\varepsilon, 0)$-differentially private. The second one simply applies the Laplace mechanism with a parameter which is a valid upper bound on the sensitivity, satisfying the desired guarantess of $(2\varepsilon, 0)$-DP. $\square$

While this framework is much more general, let's see a simple example of it in action, applied to histograms. Before, we were trying to estimate the count of entries in every bin – for now, we're just going to settle with an easier task, of just finding the most frequent element. When we were looking at histograms before, using the Laplace Mechanism, we focused on the case where the domain $\mathcal{X}$ was discrete. Indeed, as we incur an acccuracy error which decays logarithmically in $|\mathcal{X}|$, a finite domain was necessary. However, since we are relaxing our requirement from pure differential privacy to approximate differential privacy, we will see this is no longer required. One might speculate that the weaker goal of only finding the most frequent element is also responsible for these savings. This turns out to not be the case, and we will mention that a similar "stability-based" approach can also be used to estimate the counts of all elements simultaneously.

We have a dataset $X \in \mathcal{X}^n$, and we wish to compute the most frequent element (the mode). Suppose the entire dataset is the same value v: we can see that this is an incredibly stable function, with a local sensitivity of 0 for all datasets at distance $< n/2$. In particular, if we move fewer than $n/2$ datapoints, we will always have strictly greater than $n/2$ datapoints on v, resulting in the same mode. Along these lines, the distance to a dataset with a non-zero local sensitivity is very easy to compute: it is simply half the difference between the count of the most frequent and the scond most frequent element. Interestingly, note that since we are trying to apply propose-test-release with a value of $\beta = 0$, we can actually release the most frequent item *exactly*, if it occurs frequently enough.

We can now instantiate the framework described above as follows:

1. Propose a bound 0 on the local sensitivity.
2. Let γ be half the difference between the count on the most frequent and the second most frequent element in X.
3. Compute $\hat{\gamma} = \gamma + \text{Laplace}(1/\varepsilon)$.
4. If $\hat{\gamma} \leq \ln(1/\delta)/\varepsilon$, return $\perp$.
5. If $\hat{\gamma} > \ln(1/\delta)/\varepsilon$, return the most frequent element in X.

Note that this algorithm is (ε, δ)-differentially private – we save an ε, since we don't have to noise the function again at the end. What type of accuracy does it guarantee? If $\hat{\gamma} \geq \ln(1/\delta)/\varepsilon$, then we return the most frequent element exactly. We know, by Laplace tail bounds, that the Laplace noise added to γ will be of magnitude at most $\ln(1/\delta)/\varepsilon$ with probability $1 - \delta$, so this implies we need γ to be at least $2\ln(1/\delta)/\varepsilon$ for this to occur. Since this is half the difference between the "gap" for the most and second most frequent elements, we need this gap to be at least $4\ln(1/\delta)/\varepsilon$. Combining this all, we get the following theorem:

Theorem 3. *There exists an* (ε, δ)*-differentially private algorithm which identifies the most frequent element from an arbitrary dataset with probability at least* $1 - \delta$*, as long as the gap between the count of the most frequent and the second most frequent element is at least* $4\ln(1/\delta)/\varepsilon$*.*

To compare this with the Laplace histogram which provided pure differential privacy: that would have required a gap of $\Theta(\log|\mathcal{X}|/\varepsilon)$ to attain similar guarantees. In fact, we can go further: we can use a "stability-based histogram" approach to achieve similarly accurate counts for all elements simultaneously. We do not prove this here, but refer the interested reader to Theorem 3.5 in [Vad17].

Theorem 4. *There exists an* (ε, δ)*-differentially private algorithm which can, with high probability, output the count of every item in a dataset up to additive* $O(\log(1/\delta)/\varepsilon)$*.*

This is quite miraculous: the dataset may be arbitrarily large, and have an infinite domain, and we can still simultaneously estimate every count with quite low error. Once again, the equivalent theorem for pure histograms using the Laplace mechanism would pay logarithmically in the domain size $|\mathcal{X}|$.

Privately Bounding Local Sensitivity

The idea behind Propose-Test-Release was saying that the local sensitivity is small for databases in a small neighbourhood around the true database. This approach allows the local sensitivity to grow, but not too fast – we can then privately estimate the local sensitivity of the database of interest. I don't have an example of it being applied that doesn't require me to introduce new concepts (see one in Section 3.4 of [Vad17] involving differential privacy on graphs), but it's simple enough that it will hopefully be intuitive.

The previous approach "guessed" an upper bound on the sensitivity, and then checked if it was accurate. In contrast, this approach will try to estimate the value of local sensitivity, and then use the Laplace mechanism with this parameter to privatize the statistic. The analysis in the previous section will show that this approach is (ε, δ)-differentially private, so we will not repeat it.

The idea is to get a bound on the (global) sensitivity of the local sensitivity of the function f on the database X. If we can do this, we can privately compute the local sensitivity on the data in the usual way: by adding Laplace noise. That's it: the algorithm is easy to state.

1. Compute $\hat{\gamma} = \Delta_{LS}^{(f)}(X) + \text{Laplace}\left(\Delta^{\left(\Delta_{LS}^{(f)}\right)}/\varepsilon\right) + \ln(1/\delta)/\varepsilon$.
2. Output $f(X) + \text{Laplace}(\hat{\gamma}/\varepsilon)$.

Unlike Propose-Test-Release and smooth sensitivity (which we will see next), this method (when applicable) may be easier to compute. We must obtain an upper bound on the global sensitivity of the local sensitivity – this is generally done analytically. Then, the only computation required is determining the local sensitivity, which takes at most $n|\mathcal{X}|$ evaluations of the function f, and can sometimes be done even when $\mathcal{X}$ is infinite.

Smooth Sensitivity

This approach is kind of similar to the previous one: we will again allow the local sensitivity to grow bigger, but not too quickly as we get further away from the database of interest.

Another powerful technique is smooth sensitivity, introduced by Nissim, Raskhodnikova, and Smith [NRS07]. We unfortunately don't have time to get into the details in this lecture notes, but we will at least try to get across they key intuitions on how it works. Think back to the original example we gave in this lecture, considering the minimum distance between any two points. While the local sensitivity around the dataset X we considered was quite small, it was significantly larger for datasets at distance 2. Intuitively, whatever "local" notion of sensitivity we actually end up employing must account for more sensitive datasets which are nearby, but at a discounted factor depending on how far they are from X. The correct way to formalize this is not obvious. Smooth sensitivity of a function f at a dataset X is defined as the following quantity:

$$\Delta_{SS}^{(f)}(X) = \max_{X' \in \mathcal{X}^n} \{\Delta_{LS}^{(f)}(X') \exp(-\varepsilon d(X, X'))\},$$

where $d(X, X')$ is number of points which differ between X and X'. Note that we compute the maximum over *all* databases X', not just those which neighbour X. This quantity is not necessarily straightforward to compute, since it involves considering all X' in the domain, but for some simple functions (such as the median), it is tractable.

Even once we have computed the smooth sensitivity, the algorithm still has one last surprise. If one adds Laplace or Gaussian noise of magnitude calibrated to the smooth sensitivity, this would only give approximate differential privacy. To achieve pure differential privacy, one must instead add noise distributed according to a *Cauchy* random variable. The Cauchy distribution is rather unusual: to give just an example, it has no expected value, as the tails of the distribution are very heavy, and decay only polynomially. This is in contrast to the Laplace and Gaussian random variables, which have exponential decay. As a result, privatizing a statistic using smooth sensitivity often gives accuracy leaving something to be desired.

Notes

These notes are based mostly off of Section 3 of [Vad17].

References

[DL09] Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In *Proceedings of the 41st Annual ACM Symposium on the Theory of Computing*, STOC '09, pages 371–380, New York, NY, USA, 2009. ACM.

[NRS07] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the 39th Annual ACM Symposium on the Theory of Computing*, STOC '07, pages 75–84, New York, NY, USA, 2007. ACM.

[Vad17] Salil Vadhan. The complexity of differential privacy. In Yehuda Lindell, editor, *Tutorials on the Foundations of Cryptography: Dedicated to Oded Goldreich*, chapter 7, pages 347–450. Springer International Publishing AG, Cham, Switzerland, 2017.