CS 860: Algorithms for Private Data Analysis Fall 2020

Lecture 9 — Sparse Vector

Prof. Gautam Kamath *Scribe: Gautam Kamath*

Suppose we receive a sequence of k sensitivity-1 queries in an online setting: at timestep t, we receive a query $f_t : \mathcal{X}^n \to \mathbb{R}$, and we have to answer $f_t(D)$ as best we can under the constraint of differential privacy. In this setting, all we can really do is run the Laplace mechanism on the queries as they come. If there are a total of k queries, then using basic composition, we would be able to answer all queries with accuracy roughly k/ε. If we use advanced composition, we can do a bit better: $\sqrt{k}/\varepsilon$. But the point is, both of these are polynomial in k, and this is unavoidable.

Instead, we will consider a slightly easier question: we didn't necessarily want to *answer* all the queries, but only identify which ones were *large*. We can see there's a bit of hope here. Suppose we were given all the queries f_1 through f_k in advance: we could simply run the exponential mechanism. In particular, the set of objects would be the k functions $f_1, \ldots, f_k$, and the score function of f_i would be $f_i(D)$. This would pick the query with (approximately) the largest value, up to an error of roughly $\frac{\log k}{\varepsilon}$ – only logarithmic in the number of queries, rather than polynomial. Note that this algorithm can also be run iteratively to (approximately) pick out the c largest queries, increasing the error to $\frac{c \log k}{\varepsilon}$ (or $\frac{\sqrt{c} \log k}{\varepsilon}$ with approximate DP). So this shows that there's some hope – but the exponential mechanism only works in the *offline* setting, where all the queries are given to us in advance. In order to do this in the online setting, we will introduce the *sparse vector technique*.

First, since the queries will be coming in an online manner, we can't hope to say whether the first query is going to be the largest one until we see the later queries. Instead, we try to answer which queries are greater than some pre-specified (and publicly known) threshold T. With this in mind, the actual algorithm we present will be relatively straightforward, though it is notoriously challenging to get the details precisely right to guarantee privacy. We will just run the Laplace mechanism for each query: however, rather than outputting the value of this query, we only return the bit corresponding to whether it is greater than or less than some threshold. The tricky part is that we must not only add Laplace noise to the query result but also to the threshold T, and each (noisy) query result is compared to the resulting (noisy) threshold $\hat{T}$. As we will see in the proof, this has the miraculous effect of privatizing the result of many queries simultaneously, thus saving significant amounts in our privacy budget.

To state our goals again, succinctly: we have a sequence of k sensitivity-1 queries, and the goal is to identify the first c queries (where $c \ll k$) which have value greater than some (public) threshold T.

Above Threshold

We start with a simple case, when $c = 1$. Specifically, we first present AboveThreshold (Figure 1), an algorithm which keeps answering queries until it observes the first "large" one, and then it halts. For a large query, it will output $\top$, and for a small query it outputs $\bot$. The case for general c (which we discuss afterwards) will follow by composition of differential privacy.

A convenient thing to observe about AboveThreshold is that nothing in the algorithm depends

Algorithm 1 Input is a private database D, an adaptively chosen stream of sensitivity 1 queries $f_1, \ldots$, and a threshold T. Output is a stream of responses $a_1, \ldots$

AboveThreshold$(D, \{f_i\}, T, \epsilon)$

Let $\hat{T} = T + \text{Lap}\left(\frac{2}{\epsilon}\right)$.
for Each query i **do**
  Let $\nu_i = \text{Lap}(\frac{4}{\epsilon})$
  if $f_i(D) + \nu_i \geq \hat{T}$ **then**
    Output $a_i = \top$.
    Halt.
  else
    Output $a_i = \bot$.
  end if
end for

Figure 1: Algorithm 1 in [DR14], the AboveThreshold method for finding the first large query in a stream.

on the number of queries k – it could potentially be an infinitely long stream, and we would still have differential privacy. What *would* decay is the accuracy guarantee – indeed, as we take more and more Laplace random variables, the maximum deviation of any of them would become larger, leading to weaker accuracy bounds. We will quantify this more precisely later, but for now we focus on the privacy guarantee.

Theorem 1. *AboveThreshold is ε-differentially private.*

A very interesting part of this proof is how the two randomizations serve to preserve privacy in different parts of the algorithm. Roughly speaking, randomizing the threshold privatizes all but the last result, whereas the last one is privatized by the noise addition to the query.

Proof. We fix neighbouring databases D and D', and let the outputs of AboveThreshold on these two databases (with the same set of queries, thresholds, and ε) be A and A', respectively. The algorithm runs by output a stream of $\bot$, and then a single $\top$. Let a be the output $\bot^{t-1}\top$ – that is, it returns $\top$ and halts on query t, for some arbitrary t. We will relate the probabilities that $A = a$ and $A' = a$ in the usual way for differential privacy.

First, we fix the values of $\nu_1, \ldots, \nu_{t-1}$ to some arbitrary realizations – the proof will still hold, no matter what they come out to be. This relates to the prior comment about the randomization present in $\hat{T}$ privatizing all but the last query. In particular, we will compute the probabilities only with respect to ν_t and $\hat{T}$. Let $g(D)$ be the maximum noised value of the first $t-1$ queries:

$$g(D) = \max_{i \leq t-1} (f_i(D) + \nu_i).$$

Observe that, since we are fixing the realizations of the ν_i, this is a deterministic quantity.

We can write the probability that the algorithm on D outputs a as follows:

$$\begin{aligned}\Pr_{\hat{T},\nu_t}[A = a] &= \Pr_{\hat{T},\nu_t}[\hat{T} > g(D) \text{ and } f_t(D) + \nu_t \geq \hat{T}] \\ &= \Pr_{\hat{T},\nu_t}[g(D) \leq \hat{T} \leq f_t(D) + \nu_t] \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \Pr[\nu_t = v] \Pr[\hat{T} = \tau] \mathbb{1}_{\tau \in (g(D), f_t(D)+v]} dv d\tau\end{aligned}$$

At this point, we perform a change of variables. The goal is to transform from D to D', so we define the new variables to account for this shift. Let $\tau = \hat{\tau} - g(D') + g(D)$, and $v = \hat{v} - g(D') + g(D) - f_t(D) + f_t(D')$. We perform this substitution for the indicator random variable:

$$\begin{aligned}&\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \Pr[\nu_t = v] \Pr[\hat{T} = \tau] \mathbb{1}_{(\hat{\tau} - g(D') + g(D)) \in (g(D), f_t(D) + \hat{v} - g(D') + g(D) - f_t(D) + f_t(D')]} d\hat{v} d\hat{\tau} \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \Pr[\nu_t = v] \Pr[\hat{T} = \tau] \mathbb{1}_{\hat{\tau} \in (g(D'), \hat{v} + f_t(D')]} d\hat{v} d\hat{\tau}\end{aligned}$$

At this point, we convert the v and τ in the probabilities from v to $\hat{v}$, and similarly from τ to $\hat{\tau}$. As we can see from the definitions above, $|\hat{\tau} - \tau| \leq |g(D) - g(D')| \leq 1$, the latter inequality following because the queries are sensitivity-1. A similar reasoning gives that $|\hat{v} - v| \leq 2$. Putting these together with the definition of the Laplace PDF, we can upper bound the above by

$$\begin{aligned}&\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \exp(\varepsilon/2) \Pr[\nu_t = \hat{v}] \exp(\varepsilon/2) \Pr[\hat{T} = \hat{\tau}] \mathbb{1}_{\hat{\tau} \in (g(D'), \hat{v} + f_t(D')]} d\hat{v} d\hat{\tau} \\ &= \exp(\varepsilon) \Pr_{\hat{T},\nu_t}[g(D') \leq \hat{T} \leq f_t(D') + \nu_t] \\ &= \exp(\varepsilon) \Pr_{\hat{T},\nu_t}[A' = a].\end{aligned}$$

□

Next, we reason about accuracy. Since we are not outputting the numerical values of the queries, it is not immediately obvious how to define our accuracy notion. We will say that, with high probability, our algorithm makes no "mistakes". A mistake will be when the algorithm says a small query is larger than the threshold, or when a large query is smaller than the threshold. But since the answers to the queries are noisy, we must give the algorithm a bit of slack. Specifically: we say the algorithm is (α, β) accurate if, when applied to a sequence of k queries, then for all $i \in [k]$, if $a_i = \top$ then $f_i(X) \geq T - \alpha$, and if $a_i = \perp$ then $f_i(X) \leq T + \alpha$, with probability at least $1 - \beta$. This lets us get away with saying a query is large even if it is slightly below the threshold, and small even if it is slightly above. The accuracy of the algorithm will be measured in terms of this α.

Theorem 2. *Given a sequence of k queries where all but the last one are significantly smaller than the threshold (i.e., for all $i < k$, $f_i(D) \leq T - \alpha$), then AboveThreshold is (α, β) accurate for $\alpha = \frac{8(\log k + \log(2/\beta))}{\varepsilon}$.*

Observe that the error increases only logarithmically in the total number of queries k, as opposed to polynomial as we would have incurred with the naive Laplace mechanism, or if we were trying

to output the values of the queries. The accuracy guarantee is comparable to what we would get if we used the exponential mechanism, with the additional benefit of being applicable in the online setting.

Proof. Ideally, in the ith query, we would like to compare $f_i(D)$ with T. In actuality, we compare $f_i(D) + \nu_i$ with $T + (\hat{T} - T)$. Note that both ν_i and $\hat{T} - T$ are Laplace random variables. If we can bound $|\nu_i|$ and $|\hat{T} - T|$ by $\alpha/2$ for all i simultaneously, then we have the desired accuracy guarantee. For example, consider if $a_i = \top$, this implies that

$$f_i(D) + \nu_i \geq T + (\hat{T} - T).$$

Rearranging this gives that $f_i(D) \geq T + (\hat{T} - T) - \nu_i$. If the latter two terms are bounded in magnitude by $\alpha/2$, this gives $f_i(D) \geq T - \alpha$, which is the accuracy guarantee we desire. The same reasoning works for when $a_i = \perp$, try verifying the details yourself.

It remains to prove that $|\nu_i|$ and $|\hat{T} - T|$ are both bounded by $\alpha/2$. We use the tail bound $\Pr[|\operatorname{Laplace}(b)| \geq tb] = \exp(-t)$. Thus:

$$\Pr[|T - \hat{T}| \geq \alpha/2] = \exp(-\varepsilon\alpha/4),$$

$$\Pr[\max_{i \in [k]} |\nu_i| \geq \alpha/2] \leq k \exp(-\varepsilon\alpha/8).$$

The latter inequality uses a union bound. Setting both these probabilities to be at most $\beta/2$ imposes the constraint that we have the desired accuracy guarantees so long as $\alpha \geq \frac{8(\log(2/\beta) + \log k)}{\varepsilon}$. □

Sparse Vector

AboveThreshold is a convenient primitive for detecting a single large query, and it can be chained together multiple times to detect c large queries. This is a testament to the power of differential privacy – despite the fact that the privacy proof for AboveThreshold was rather technical, it is easy to use it as a building block for this more complicated scenario. The sparse vector algorithm is displayed in Figure 2.

Let's inspect the privacy guarantee in the pure DP case (when $\delta = 0$). The algorithm is exactly equivalent to running AboveThreshold c times in a row with parameter ε/c, starting again every time AboveThreshold halts. Importantly, the threshold is re-randomized after each $\top$ is output, otherwise it would not be equivalent to multiple subsequent runs of AboveThreshold. By basic composition, the overall algorithm is ε-DP. A similar analysis holds for $\delta > 0$ but using advanced composition, resulting in (ε, δ)-DP.

Theorem 3. *Sparse is (ε, δ)-DP, for any $\varepsilon > 0$, $\delta \geq 0$.*

Similarly, the accuracy can be reasoned from Theorem 2. β is rescaled by a factor of c and a union bound is applied. Additionally, ε is rescaled as done in Figure 2 to guarantee privacy (by composition). Putting these together allows us to conclude the following accuracy guarantee.

Theorem 4. *Suppose we are given a sequence of k queries where only c are large (i.e., the number of i such that $f_i(D) \geq T - \alpha$ is at most c). If $\delta = 0$, then Sparse is (α, β) accurate for $\alpha = \frac{8c(\log k + \log(2c/\beta))}{\varepsilon}$. If $\delta > 0$, then it is (α, β) accurate for $\alpha = \frac{\sqrt{512c \log(1/\delta)}(\log k + \log(2c/\beta))}{\varepsilon}$.*

Algorithm 2 Input is a private database D, an adaptively chosen stream of sensitivity 1 queries $f_1, \ldots$, a threshold T, and a cutoff point c. Output is a stream of answers $a_1, \ldots$

Sparse$(D, \{f_i\}, T, c, \epsilon, \delta)$

If $\delta = 0$ **Let** $\sigma = \frac{2c}{\epsilon}$. **Else Let** $\sigma = \frac{\sqrt{32c \ln \frac{1}{\delta}}}{\epsilon}$
Let $\hat{T}_0 = T + \text{Lap}(\sigma)$
Let count $= 0$
for Each query i **do**
  Let $\nu_i = \text{Lap}(2\sigma)$
  if $f_i(D) + \nu_i \geq \hat{T}_{\text{count}}$ **then**
    Output $a_i = \top$.
    Let count $=$ count $+1$.
    Let $\hat{T}_{\text{count}} = T + \text{Lap}(\sigma)$
  else
    Output $a_i = \perp$.
  end if
  if count $\geq c$ **then**
    Halt.
  end if
end for

Figure 2: Algorithm 2 in [DR14], the Sparse Vector algorithm.

We see that the dependence on the number of large values c is polynomial, and on total number of queries k is logarithmic, allowing us to detect large queries from a big set with good accuracy, as long as not too many of them are large.

We note that there is also a variant of sparse vector, which we refer to as NumericSparse, which can output the (approximate) value of everything that is above the threshold at the cost of a small multiplicative factor in α. Specifically, if the algorithm is going to output $\top$ for a query, it instead adds fresh Laplace noise to the value of the query and outputs the result.[1] The output of the algorithm will now be a stream of values in $\mathbb{R} \cup \{\perp\}$, with the following guarantees:

- If $a_i = \perp$, then $f_i(D) \leq T + \alpha$;
- If $a_i \in \mathbb{R}$, then $f_i(D) \geq T - \alpha$ and $|f_i(D) - a_i| \leq \alpha$.

The value of α is the same as in Theorem 4, up to a constant factor. This constant factor arises since we have one more application of the Laplace mechanism when returning the numeric query values. To maintain the same privacy budget, we must rescale ε.

The intuition behind this modification: those whole idea behind the sparse vector technique is to pay logarithmically in the total number of queries, but polynomially in the number which are above the threshold. Since we already pay for these queries anyway, we can afford to do another private operation for each of them without changing the overall privacy cost by more than a constant factor.

[1] It is important that fresh noise is added once again, otherwise it will not be private.

Online Private Multiplicative Weights

With the powerful tool of sparse vector, we can revisit the private multiplicative weights algorithm, and port it to the online setting. First, we recall the (offline) version of private multiplicative weights. We try to maintain a distribution $\hat{p}$ over the domain elements, with the goal of matching the true dataset (in its histogram representation) p on all linear queries in some set Q.[2] We start with $\hat{p}$ being a uniform distribution over the domain $\mathcal{X}$, and repeatedly apply the following:

1. Use the exponential mechanism to privately choose a query $f \in Q$ where $\hat{p}$ and p would have significantly different answers.
2. Privately measure how large the error is. If it is small, then we return the current $\hat{p}$.
3. Otherwise, perform a multiplicative weights update on $\hat{p}$, based on the (private) result of f on p.

Guarantees of multiplicative weights allow us to bound the total number of iterations of this procedure as $O(\log |\mathcal{X}|/\alpha^2)$.

We move to the online setting by integrating this approach with the sparse vector technique. In this setting, we are given k queries $f_1, \ldots, f_k$ sequentially from a set Q. Our goal is slightly relaxed: instead of trying to output a "synthetic dataset" (a $\hat{p}$ which gives accurate answers on all queries in Q), we instead try to simply answer the queries correctly as they come.[3]

The algorithm will again initialize $\hat{p}$ to be a uniform distribution over the domain $\mathcal{X}$. However, this time, it will run the sparse vector technique on the queries $|\langle \hat{p}, f\rangle - \langle p, f\rangle|$, with threshold $T = \Theta(\alpha)$. In each iteration:

1. If Sparse returns $\perp$, this means that $|\langle \hat{p}, f\rangle - \langle p, f\rangle|$ is small, that $\langle \hat{p}, f\rangle \approx \langle p, f\rangle$. Therefore, we can just output $\langle \hat{p}, f\rangle$, and continue to the next iteration.
2. If Sparse returns $\top$, that means that $|\langle \hat{p}, f\rangle - \langle p, f\rangle|$ is large. We privately compute and output $\langle p, f\rangle$ (essentially using NumericSparse as mentioned above, which allows us to return the values of the large queries).
3. We perform a multiplicative weights update on $\hat{p}$ using the (private) result of f on p, and continue to the next iteration.

Note that, once again, the total number of iterations of this procedure is at most $O(\log |\mathcal{X}|/\alpha^2)$.

Given the tools we have developed, the analysis is straightforward. Privacy is simply by the guarantees of sparse vector, followed by privacy of the Laplace mechanism (or, one could use NumericSparse as a black box). As for accuracy, this is simply due to the accuracy guarantees of sparse vector in combination with the accuracy guarantees of the multiplicative weights algorithm for queries as discussed in the last lecture. With all this in hand, it is an exercise to derive the following guarantees for the online private multiplicative weights algorithm.

[2] Note that there's a small change in the setting here – as we are looking at normalized linear queries (i.e., ones which have values in $[0, 1]$) which makes the sensitivity $1/n$, in comparison to the sensitivity-1 queries discussed above.

[3] It is not hard to get the stronger synthetic dataset guarantee – you can just remember all the queries that were asked, and do the offline algorithm afterwards.

Theorem 5. *The Online Private Multiplicative Weights algorithm can, in an online manner, answer a set Q of linear queries on a database of size n to accuracy α under (ε, δ)-DP, given $n = \tilde{O}\left(\frac{\log|Q|\sqrt{\log|\mathcal{X}|\log(1/\delta)}}{\alpha^2\varepsilon}\right)$ datapoints.*

Notes

Content in this chapter is based heavily off Section 3.6 of [DR14].

References

[DR14] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3–4):211–407, 2014.