The Algorithmic Foundations of Adaptive Data Analysis November 6, 2017

Lecture 12: The Sparse Vector Technique

Lecturer: Adam Smith *Scribe: Adam Smith*

1 The Sparse Vector Technique

Recall that in Lecture 5, we saw the "AboveThreshold" algorithm, which is $\log_2(k+1)$-compressible when run for k rounds:

Algorithm 1: AboveThreshold($\mathbf{s}, T, q_1, q_2, \ldots$):

AllDone $\leftarrow$ **FALSE**;
while *not* **AllDone do**
    Accept the next query q_j;
    $a_j \leftarrow q_j(\mathbf{s})$;
    if $a_j < T$ **then**
        return $b_j = \perp$;
    else
        return $b_j = \top$;
        AllDone $\leftarrow$ **TRUE** ;

We will see a differentially private version of the algorithm which will allow us to get differentially-private versions of the Ladder, Median and Re-usable Holdout Mechanisms. The changes, highlighted in red, are that we use a noisy threshold $\tilde{T}$ instead of T.

Algorithm 2: SparseVector($\mathbf{s}, T, \Delta, \epsilon, q_1, q_2, \ldots$):

Input: $q_1, q_2 \ldots$ is a stream of Δ-sensitive queries
AllDone $\leftarrow$ **FALSE**;
$\tilde{T} = T + Z_0$ where $Z_0 \sim \text{Lap}(2\Delta/\epsilon))$;
while *not* **AllDone do**
    Accept the next query q_i;
    $a_i \leftarrow q_i(\mathbf{s})$;
    $\tilde{a}_i \leftarrow a_i + Z_i$ where $Z_i \sim \text{Lap}(4\Delta/\epsilon)$;
    if $\tilde{a}_i < \tilde{T}$ **then**
        return $b_j = \perp$;
    else
        return $b_j = \top$;
        AllDone $\leftarrow$ **TRUE** ;

Theorem 1 *The Sparse Vector mechanism is $(\epsilon, 0)$-differentially private.*

Before reading the proof, it may be helpful to work through the following exercise:

Exercise 1 *Show that Sparse Vector is* not *$(\epsilon, 0)$-differentially private (for any $\epsilon < \infty$) if we use the unperturbed threshold T instead of $\tilde{T}$.*

Proof Fix an output of the form $(\bot)^{k-1}\top$ for some $k \in \mathbb{N}$ (we leave the proof for the output $(\bot)^\infty$ as an exercise). As in other proofs, we may condition on the analyst's random coins and consider only a deterministic analyst. Thus, when considering a single output sequence $(\bot)^{k-1}\top$, we need only consider a single query sequence $q_1, ..., q_k$ of Δ-sensitive queries. For the remainder of the proof, let $\Delta = 1$ (since we can always rescale query answers and T so that queries are 1-sensitive without changing the output).

We will condition on the values $Z_1 = z_1, ..., Z_{k-1} = z_{k-1}$. With these values fixed, consider the function

$$g(\mathbf{s}) \stackrel{\text{def}}{=} \max_{j=1}^{k-1} q_j(\mathbf{s}) + z_j \,.$$

Observe that g is the maximum of 1-sensitive queries, it is itself 1-sensitive. Also, the output $(\bot)^{k-1}\top$ occurs if and only if

$$g(\mathbf{s}) < \tilde{T} \leq q_k(\mathbf{s}) + Z_k \qquad (\text{"Event } E_{\mathbf{s}}\text{"}) \tag{1}$$

Now fix two adjacent data sets $\mathbf{s}, \mathbf{s}'$. We want to compare the probability of events $E_{\mathbf{s}}$ and $E_{\mathbf{s}'}$. Notice that if we were to first fix Z_k, then the events' probabilities might be very different (for example, one might be zero and the other nonzero).

To do the comparison, we set up a 1-1 correspondence between the randomness of the two variants. For a given pair $\tilde{T} = \tau, Z_k = z$ that might occur when the data is $\mathbf{s}$, we will consider a different pair (τ', z') for data $\mathbf{s}'$, where

$$\begin{aligned} \tau &\mapsto \tau' \stackrel{\text{def}}{=} \tau + g(\mathbf{s}) - g(\mathbf{s}') \\ z &\mapsto z' \stackrel{\text{def}}{=} z + g(\mathbf{s}) - g(\mathbf{s}') + q_k(\mathbf{s}') - q_k(\mathbf{s}') \end{aligned}$$

We have chosen z' so that the length of the interval in which $\tilde{T}$ must land is the same if we condition on $Z_k = z$ when the data is $\mathbf{s}$ or on $Z_k = z'$ when the data is $\mathbf{s}'$. That is, $q_k(\mathbf{s}) + z - g(\mathbf{s}) = q_k(\mathbf{s}') + z' - g(\mathbf{s}')$. So instead of conditioning on the same value for Z_k under both $\mathbf{s}$ and $\mathbf{s}'$, we will condition on different values. These values are not too far apart, though: $|\tau' - \tau| \leq 1$, and $|z' - z| \leq 2$. Now,

$$\frac{\Pr(E_{\mathbf{s}})}{\Pr(E_{\mathbf{s}'})} = \frac{\int_z \Pr(E_{\mathbf{s}} | Z_k = z) f_{Z_k}(z) dz}{\int_z \Pr(E_{\mathbf{s}'} | Z_k = z') f_{Z_k}(z') dz'} \leq \sup_{\substack{z \in \mathbb{R} \\ z' = z + g(\mathbf{s}) - g(\mathbf{s}') + q_k(\mathbf{s}') - q_k(\mathbf{s}')}} \frac{\Pr(E_{\mathbf{s}} | Z_k = z)}{\Pr(E_{\mathbf{s}'} | Z_k = z')} \cdot \frac{f_{Z_k}(z)}{f_{Z_k}(z')} \,.$$

We can bound each of the two ratios in the right-hand expression separately. For the first term, we are comparing the probability that $\tilde{T}$ lands in two different intervals of the same length, which are *shifted relative to each other by at most 1*. Thus, the first ratio is bounded by $\exp(d_\diamond(\text{Lap}(\frac{2}{\epsilon}), 1 + \text{Lap}(\frac{2}{\epsilon}))) = e^{\epsilon/2}$.

In the second ratio, we are comparing the density of $\text{Lap}(4/\epsilon)$ at two points *within distance 2 of each other*. The ratio is thus bounded by $\exp(d_\diamond(\text{Lap}(\frac{4}{\epsilon}), 2 + \text{Lap}(\frac{4}{\epsilon}))) = e^{\epsilon/2}$.

Combining these, we get that $\frac{\Pr(E_{\mathbf{s}})}{\Pr(E_{\mathbf{s}'})} \leq e^\epsilon$, as desired. ■

What should accuracy mean for this thresholding algorithm? One simple measure is the following: given a run of the algorithm with queries $q_1, q_2, ...$ and $b_1, b_2, ...$, the algorithm's *empirical error* at a given round is $\max(0, q_j(\mathbf{s}) - T)$ if $b_j = \bot$ and $\max(0, T - q_j(\mathbf{s}))$ if $b_j = \top$. (That is, it is the gap between $q_j(\mathbf{s})$ and T when the wrong decision was made, and 0 otherwise.) When the data are drawn i.i.d from distribution $\mathcal{D}$, the *population* error is defined the same way, with $q_j(\mathcal{D})$ replacing $q_j(\mathbf{s})$.

Theorem 2 *For all data sets* $\mathbf{s}$*, all analysts* A*, and all* $\beta > 0$*, when run on a sequence of* k *queries, with probability* $1 - \beta$ *over the coins of the algorithm and* A*, the sparse vector algorithm has empirical error at most* $\alpha = \frac{6\Delta \ln((k+1)/\beta)}{\epsilon}$ *at all rounds up to termination.*

A direct corollary is that Sparse Vector has expected empirical error at most $O(\Delta \frac{\ln(k)}{\epsilon})$.

Proof The $|Z_i|$'s are exponential with parameters $2\Delta/\epsilon$ for $i = 0$ and $4\Delta/\epsilon$ for $i = 1, ...k$. By Lemma 5 from last lecture, with probability at least $1 - \beta$, none of them exceeds its parameter by a factor of more than $\ln((k+1)/\beta)$. ■

2 Using Sparse Vector

Suppose we want an algorithm that reports several above-threshold queries (for example, suppose we want to shut down the algorithm only after m occurrences of outputting $\top$). We can simply run m copies of Sparse Vector in sequence. If there are k queries overall, the resulting algorithm will have expected empirical error $O(\frac{\log(m+k)}{n\epsilon}) = O(\frac{\log(k)}{n\epsilon})$ (by a union bound over the $m+k$ Laplace random variables generated during the runs). Of course, the resulting algoithm's privacy/stability parameters will degrade with m: the algorithm will be $m\epsilon$-differentially private (by composition) and τ-KL stable for $\tau \leq m\epsilon(e^\epsilon - 1)$.

Where does this leave us with population error? The expected population error of the algorithm will be at most the sum of its expected empirical and generalization errors, that is,

$$O\Big(\underbrace{\frac{\log(k)}{n\epsilon}}_{\substack{\text{empirical}\\ \text{error}}} + \underbrace{\epsilon\sqrt{m}}_{\substack{\text{gen. error}\\ \text{using } \tau\text{-KL}\\ \text{stability}}}\Big), \text{ which is } O\left(\frac{m^{1/4}\log^{1/2} k}{n^{1/2}}\right) \text{ for } \epsilon = \sqrt{(\log k)/n\sqrt{m}}\,.$$

For some of our applications, we will also want high-probability bounds on the empirical error. We know that with probability $1-\beta$, the empirical error will be at most $O(\log(k/\beta)/\epsilon)$. With this bound, setting ϵ appropriately ($\epsilon = \sqrt{(\log k/\beta)/n\sqrt{m}}$), we will have both generalization error that is (in expectation) on the order of the empirical error, which is (with high probability) $O\Big(\frac{m^{1/4}\log^{1/2}(k/\beta)}{n^{1/2}}\Big)$

We can combine this algorithm with Laplace or Gaussian noise to get a distributionally stable version of Guess and Check from Lecture 6.

PrivGuessAndCheck$(T, \epsilon, m, \Delta, (q_1, g_1), (q_2, g_2), \ldots)$
  TimesWrong $\leftarrow 0$
  while TimesWrong $< m$ **do**
    Start an instance of **SparseVector** with threshold T, privacy parameter *epsilon*, and sensitivity Δ.
    while AboveThreshold has not halted **do**
      Accept the next query (q_i, g_i).
      Feed **AboveThreshold** the query $\hat{q}_i(S) = |q_i(S) - g_i|$.
      if AboveThreshold returns $\perp$ **then**
        Return the answer $a_i = g_i$
      end if
    end while
    Return the answer $a_i = q_i(\mathbf{s}) + Z_i$ where $Z_i \sim \text{Lap}(4\Delta/\epsilon)$.
    TimesWrong $\leftarrow$ TimesWrong $+ 1$
  end while

Given a sequence of k $\frac{1}{n}$-sensitive queries and conjectured values, this algorithm will provide answers with error $\eta = O\Big(\frac{m^{1/4}\log^{1/2}(k/\beta)}{n^{1/2}}\Big)$ until it halts (since using the Laplace mechanism to answer queries for which the conjectured answers are far off at most doubles the privacy/stability parameters, and increases the number of Laplace random variables by at most a factor of 2). In contrast, the compressibility version from Lecture 6 had an error bound of $O(\sqrt{\frac{m\log(kn/m)}{n}})$.

Recall the median mechanism from Lecture 6. We can write down a differentially private version:

We can run the same algorithm using the distributional stable version of Guess and Check. Recall that when the "model" database has size $n' \geq \log(4k)/(2\eta^2)$, the algorithm can make at most $m = n' \log|\mathcal{X}|$ guesses that are off by more than η. Moreover, so long as η is set so that no answer given has *empirical* answer greater than η, the median oracle will never end up with an empty version space, and so will be able to continue answering queries. If each iteration of above threshold is ϵ-differentially private, we get overall (expected) error

MedianOracle$(q_1, \ldots, q_k)$

Let $n' = \frac{\ln(4k)}{2\eta^2}$. Initialize an instance of **PrivGuessAndCheck**(η, m) with $m = n' \log|\mathcal{X}|$ and $\eta = O\left(\frac{m^{1/4}\log^{1/2}(k/\beta)}{n^{1/2}}\right)$.

Initialize a version space $\mathcal{S}_0 = \mathcal{X}^{n'}$.

for $i = 1$ to k **do**

  Given query q_i, construct a guess $g_i = \text{median}\left(\{q_i(S') : S' \in \mathcal{S}_{i-1}\}\right)$

  Feed the query (q_i, g_i) to **PrivGuessAndCheck** and receive answer a_i.

  if $\hat{a}_i = g_i$ **then**

    $\mathcal{S}_i \leftarrow \mathcal{S}_{i-1}$

  else

    $\mathcal{S}_i \leftarrow \mathcal{S}_{i-1} \setminus \{S' \in \mathcal{S}_{i-1} : |q_i(S') - a_i| > \eta\}$

  end if

  Return answer a_i.

end for

$$O\left(\frac{m^{1/4}\log^{1/2}(k/\beta)}{n^{1/2}}\right) = O\left(\frac{(\ln(k)\log|\mathcal{X}|)^{1/4}}{\eta^{1/2}} \cdot \frac{\log^{1/2}(k/\beta)}{n^{1/2}}\right)$$

Recall that the expected error has to be no more than η in order for the algorithm to succeed. Setting the expected error to be equal to η above, we obtain that with probability $1-\beta$, the algorithm answers all queries, and that the expected error is:

$$O(\eta) = O\left(\frac{\log^{1/2}(k/\beta)\log^{1/6}|\mathcal{X}|}{n^{1/3}}\right) \quad \text{or, solving for } n, \quad n = O\left(\frac{\log^{3/2}(k/\beta)\log^{1/2}|\mathcal{X}|}{\eta^3}\right).$$

This last bound should be interpreted as a sample error guarantee: it is a sufficient upper bound on n for the algorithm to give overall expected error η.

Exercise 2 *Use the differentially private version of Guess and Check to derive improved versions of the Ladder mechanism and Reusable Holdout (from Lecture 5 and 6, respectively). What bounds can you get on the expected error of each of these algorithms?*

3 Notes

The Sparse Vector algorithm is notoriously trickly to analyze, and several incorrect versions appear in the literature. Variants of the algorithm first appeared in [DNR+09, RR10]. The simple, general version here is adapted from [HR10]. A survey of some of the incorrect variants appears in Lyu, Su and Li (arXiv 1603.01699 [CR]). The presentation here is inspired by those of Dwork and Roth (2014) and Kifer and Zhang (POPL 2017). The differentially private version of the median mechanism is from [RR10].

References

[DNR+09] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil P. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In *STOC*, pages 381–390. ACM, May 31 - June 2 2009.

[HR10] Moritz Hardt and Guy Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *Proc. 51st Foundations of Computer Science (FOCS)*, pages 61–70. IEEE, 2010.

[RR10] Aaron Roth and Tim Roughgarden. Interactive privacy via the median mechanism. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 765–774. ACM, 2010.