Detecting Violations of Differential Privacy

Zeyu Ding

Pennsylvania State University

Collaborators:

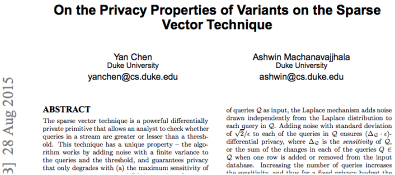
- Yuxin Wang (ykw5136@psu.edu)
- Danfeng Zhang (zhang@cse.psu.edu)
- Guanhong Wang (gpw5092@psu.edu)
- Daniel Kifer (dkifer@cse.psu.edu)

Motivation

- A mathematically rigorous definition of privacy is needed, with a computationally rich class of algorithms that satisfy this definition.
- **Differential Privacy** is such a definition that is widely accepted.

Motivation

- However, many published algorithms are incorrect.
- How can we identify incorrect ones?
- How can we fix them?



On the Privacy Properties of Variants on the Sparse Vector Technique

Yan Chen
Duke University
yanchen@cs.duke.edu

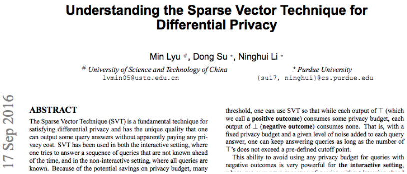
Ashwin Machanavajjhala
Duke University
ashwin@cs.duke.edu

3] 28 Aug 2015

ABSTRACT

The sparse vector technique is a powerful differentially private primitive that allows an analyst to check whether queries in a stream are greater or lesser than a threshold. This technique has a unique property – the algorithm works by adding noise with a finite variance to the queries and the threshold, and guarantees privacy that only degrades with (a) the maximum sensitivity of

of queries Q as input, the Laplace mechanism adds noise drawn independently from the Laplace distribution to each query in Q. Adding noise with standard deviation of $\sqrt{2}/\epsilon$ to each of the queries in Q ensures $(\Delta_Q \cdot \epsilon)$-differential privacy, where Δ_Q is the *sensitivity* of Q, or the sum of the changes in each of the queries $Q \in Q$ when one row is added or removed from the input database. Increasing the number of queries increases



Understanding the Sparse Vector Technique for Differential Privacy

Min Lyu [#], Dong Su [*], Ninghui Li [*]
[#] *University of Science and Technology of China*
lvmin05@ustc.edu.cn
[*] *Purdue University*
{su17, ninghui}@cs.purdue.edu

17 Sep 2016

ABSTRACT

The Sparse Vector Technique (SVT) is a fundamental technique for satisfying differential privacy and has the unique quality that one can output some query answers without apparently paying any privacy cost. SVT has been used in both the interactive setting, where one tries to answer a sequence of queries that are not known ahead of the time, and in the non-interactive setting, where all queries are known. Because of the potential savings on privacy budget, many

threshold, one can use SVT so that while each output of $\top$ (which we call a **positive outcome**) consumes some privacy budget, each output of $\perp$ (**negative outcome**) consumes none. That is, with a fixed privacy budget and a given level of noise added to each query answer, one can keep answering queries as long as the number of $\top$'s does not exceed a pre-defined cutoff point.

This ability to avoid using any privacy budget for queries with negative outcomes is very powerful for **the interactive setting**,

Our goal

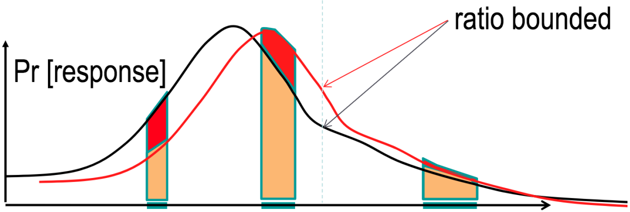
- Help algorithm designers test their designs in a **semi-blackbox** manner.
  - No restrictions on noise mechanisms used.
  - No restrictions on programming languages used.
- If the algorithm fails the test, provide hints to the designer as how to fix it by generating counterexamples.

Definition of differential privacy

Definition (Dwork '06)

Let $\epsilon \geq 0$. An algorithm M is said to be ϵ-differentially private if for every pair of adjacent databases D_1 and D_2, and every $E \subseteq \mathrm{Range}(M)$, we have

$$P(M(D_1) \in E) \leq e^{\epsilon} \cdot P(M(D_2) \in E).$$



Credit: *C. Dwork*

What is a counterexample?

A **counterexample** consists of

- a pair of adjacent databases D_1, D_2;
- an event E of possible outputs of M;
- "strong evidence" that differential privacy is violated, i.e.,

$$P(M(D_1) \in E) > e^{\epsilon} \cdot P(M(D_2) \in E).$$

How our tool detects violations of DP

Find candidate pairs (D_1, D_2)
Input Generation

- Confine search space to a small but highly representative set

For every (D_1, D_2) find an E
Event Selection

- Confine search space for E
- Use hypothesis test in exploratory mode to select the best E

For every (D_1, D_2, E) try to find strong evidence that $P(M(D_1) \in E) > e^{\epsilon} \cdot P(M(D_2) \in E)$

- Use hypothesis test again to generate evidence

Outline

1. **Motivation**
2. **Detecting Privacy Violations**
   - Finding Evidence
   - Event Selection
   - Input Generation
3. **Evaluations**

Evidence? Use statistical hypothesis test

- **Problem statement**: Given (D_1, D_2, E), how do we show there is violation of differential privacy under this setting?
- In other words, let $p_1 = P(M(D_1) \in E)$ and $p_2 = P(M(D_2) \in E)$. How to show strong evidence indicating

$$p_1 > e^{\epsilon} \cdot p_2?$$

- We do hypothesis testing.

What is a hypothesis test?

We ask the following question: suppose $p_1 \leq e^{\epsilon} \cdot p_2$, what is the probability of seeing the data we have?

- The assumption $p_1 \leq e^{\epsilon} \cdot p_2$ is called the **null hypothesis**.
- The probability of seeing a data sample under the null hypothesis is bounded by the so-called **p-value**.
- If the p-value is small (say, $< 1\%$), we consider the data sample as strong evidence that the null hypothesis is not true.

How to do a hypothesis test?

First we obtain sample data.

- Run M on D_1 many (n) times.
- Count how many outputs are inside E, denote this number by c_1.
- Note: c_1 is equivalent to a sample from Binomial(n, p_1).
- Repeat this process on D_2 to get another count c_2.

How to calculate a valid p-value

Then we compute a p-value.

- The difficulty is that we don't know p_1 and p_2.
- The good news is that we don't need to know p_1 and p_2. We just need to know if $p_1 \leq e^{\epsilon} \cdot p_2$.
- Therefore, we
  - downsample $\tilde{c}_1$ from Binomial$(c_1, 1/e^{\epsilon})$. The marginal distribution of $\tilde{c}_1$ is Binomial$(n, p_1/e^{\epsilon})$.
  - apply *Fisher's Exact Test* on $\tilde{c}_1, c_2$.
  - average to reduce variance.

Outline

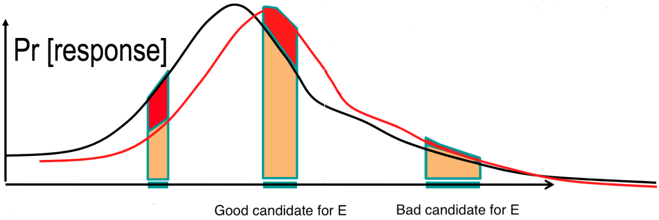
1. **Motivation**

2. **Detecting Privacy Violations**
   - Finding Evidence
   - Event Selection
   - Input Generation

3. **Evaluations**

Event selection: how to choose the best E

Problem statement: Given (D_1, D_2), how to find a good E where differential privacy is violated?



Event selection: how to choose the best E

- Confine the search space for E according to the return types of M.
  - If M returns a list of Booleans, we could say "let E be the set of outputs with 5 Trues".
  - If M returns a list of reals, we could say "let E be the set of outputs with minimal value in $[-1.0, 1.0]$", etc..
- Run hypothesis test in exploratory mode (with $n = 100,000$).
- The one with the smallest p-value is selected as the best candidate.

Outline

1. **Motivation**
2. **Detecting Privacy Violations**
   - Finding Evidence
   - Event Selection
   - Input Generation
3. **Evaluations**

Input generation: find adj. databases (D_1, D_2)

Problem statement: How do we find good candidate pairs (D_1, D_2) to begin with?

- Confine the search space to a small set on which exhaustive search is feasible
- Pairs in this set should be:
  - simple, so counterexamples are easier to understand for humans
  - maximum difference
    i.e., if a query can change by a max of 1, then it will change by 1.
  - highly representative.

Input generation: find adj. databases (D_1, D_2)

- To represent the case where a few queries increase but most queries decrease, we use

$$\text{one-above-rest-below: } [1,1,1,1],[2,0,0,0]$$

- To represent the case where approximately half queries increase and half queries decrease, we use

$$\text{half-half: } [1,1,1,1],[2,2,0,0]$$

- We find that this simple and generic approach works surprisingly well.
- For the complete list, please refer to the paper.

Input generation: find additional arguments

Some algorithms require additional input arguments.

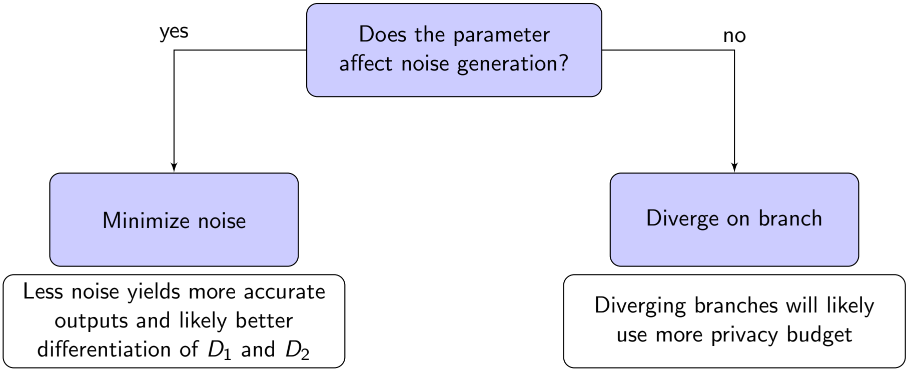
```
function SparseVector(Q, T, ε, N):
    out ← []
    η1 ← Lap(1/(ε/2))
    T̃ ← T + η1
    count ← 0
    foreach q in Q do
        η2 ← Lap(1/(ε/4N))
        if q + η2 ≥ T̃ then
            out ← ⊤ :: out
            count ← count + 1
            if count ≥ N then
                Break
            end
        else
            out ← ⊥ :: out
        end
    end
    return (out)
```

- Parameter T: a preset threshold.
- Parameter N: bound on number of above-the-threshold queries that can be answered.
- $\mathrm{Lap}(r)$ draws one sample from the Laplace$(0, r)$ distribution:

$$f(x \mid r) = \frac{1}{2r}\exp\left(-\frac{|x|}{r}\right)$$

Input generation: find additional arguments

- Additional parameters: symbolic execution using two heuristics.



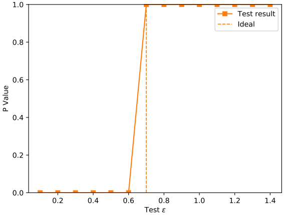
Outline

1. **Motivation**
2. **Detecting Privacy Violations**
   - Finding Evidence
   - Event Selection
   - Input Generation
3. **Evaluations**

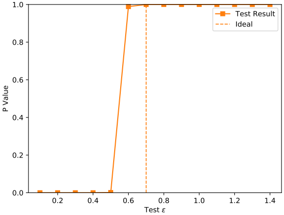
How do we evaluate our tool

- If M claims to be ϵ_0-differentially private, we run our counterexample detector for all ϵ's in a range containing ϵ_0.
- We plot the p-value of the counterexample found for every ϵ.
- When the p-value is close to 0 for some ϵ, it means we have strong evidence that M is not ϵ-differentially private.
- The largest ϵ with a p-value close to 0 can be regarded as a lower bound for the true ϵ_0.

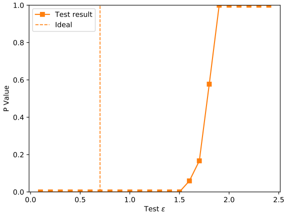
How to interpret the graphs



No violations found.
M correctly claims 0.7-differential privacy.



No violations found.
M correctly claims 0.7-differential privacy but could be **too conservative**.



Violations found.
The true ϵ_0 might be around 1.6.

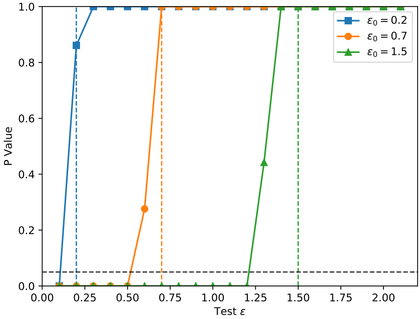
Evaluation: correct SparseVector

Algorithm 3: Correct SVT

function `SparseVector`(Q, T, ϵ, N):

- $out \leftarrow []$
- $\eta_1 \leftarrow \mathrm{Lap}(2/\epsilon)$
- $\tilde{T} \leftarrow T + \eta_1$
- $count \leftarrow 0$
- **foreach** q *in* Q **do**
  - $\eta_2 \leftarrow \mathrm{Lap}(2N/\epsilon)$
  - **if** $q + \eta_2 \geq \tilde{T}$ **then**
    - $out \leftarrow \top :: out$
    - $count \leftarrow count + 1$
    - **if** $count \geq N$ **then**
      - **Break**
    - **end**
  - **else**
    - $out \leftarrow \bot :: out$
  - **end**
- **end**
- **return** (out)

[Lyu et al., 2017]



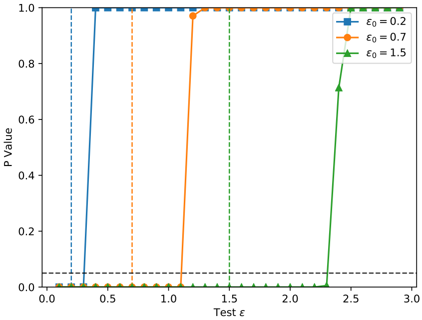
No violations found. The result is as expected.

Evaluation: incorrect SparseVector

Algorithm 4: iSVT3, $\frac{1+6N}{4}\epsilon$-DP.

function SparseVector(Q, T, ϵ, N):
- $out \leftarrow [\,]$
- $\eta_1 \leftarrow \mathrm{Lap}(4/\epsilon)$
- $\tilde{T} \leftarrow T + \eta_1$
- $count \leftarrow 0$
- **foreach** q *in* Q **do**
  - // noise added doesn't scale with N
  - $\eta_2 \leftarrow \mathrm{Lap}(4/3\epsilon)$
  - **if** $q + \eta_2 \geq \tilde{T}$ **then**
    - $out \leftarrow \top :: out$
    - $count \leftarrow count + 1$
    - **if** $count \geq N$ **then**
      - **Break**
    - **end**
  - **else**
    - $out \leftarrow \perp :: out$
  - **end**
- **end**
- **return** (out)

[Lee and Clifton, 2014]



- Violations detected.
- For claimed $\epsilon_0 = 0.2, 0.7, 1.5$, the **true** $\epsilon_0 = 0.35, 1.225, 2.625$.
- Our tool provides good estimates of the true ϵ_0.

Summary

- We evaluated our tool on (variations of) SparseVector, NoisyMax, Histogram, Laplace Mechanism, etc..
- For all correct variations, our tool finds no violations, confirming that their implementation is correct.
- For all incorrect variations, our tool is able to detect violations.
- The time spent on generating one single point in the figures is $1 \sim 23$ seconds on a double Intel® Xeon® E5-2620 v4 @ 2.10GHz CPU machine with 64 GB memory.
- Code available at: https://github.com/cmla-psu/statdp

Thank you!

Question?

References

Chen, R., Xiao, Q., Zhang, Y., and Xu, J. (2015).
Differentially private high-dimensional data publication via sampling-based inference.
In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 129–138. ACM.

Dwork, C., Roth, A., et al. (2014).
The algorithmic foundations of differential privacy.
Foundations and Trends® in Theoretical Computer Science, 9(3–4):211–407.

Lee, J. and Clifton, C. W. (2014).
Top-k frequent itemsets via differentially private fp-
In *Proceedings of the 20th ACM SIGKDD international on Knowledge discovery and data mining*, pages 931–940. ACM.

Lyu, M., Su, D., and Li, N. (2017).
Understanding the sparse vector technique for differential privacy.
Proceedings of the VLDB Endowment, 10(6):637–648.

Roth, A. (2011).
Sparse vector technique, lecture notes for "the algorithmic foundations of data privacy".

Stoddard, B., Chen, Y., and Machanavajjhala, A. (2014).
Differentially private algorithms for empirical machine learning.
arXiv preprint arXiv:1411.5428.