Automatic, Fine-Grained Algorithmic Choice for Differential Privacy

Jacob Imola
School of Computer Science, Carnegie Mellon University

Jean Yang
School of Computer Science, Carnegie Mellon University

February 6, 2018

1 Introduction

Why is algorithmic choice in differential privacy important?

The rapid technological increase in data collection, speed, and storage has brought about revolutionary insights and ideas and will continue to do so. However, with huge amounts of private data comes the concern of preventing data from ending up in the wrong hands. In order to prevent data leakage, we must lay a strong privacy foundation and give data analysts the tools they need without forcing them to become privacy experts.

Consider a healthcare database with records like patient weight, age, some genetic information, or whether they are HIV positive. Granting access rights to just the patients and their doctors protects privacy perfectly and developing tools for verification is an interesting question in its own right. However, sometimes it's okay to release a few general statistics about a database so that an analyst can find risk factors for people who have HIV. On the other side of the spectrum, publicly releasing all the information doesn't protect an individual's information. It is therefore necessary to use a middle ground tool where the amount of information released can be reliably controlled. Perhaps the most promising tool is differential privacy.

Differential privacy has been the interest of many researchers and programmers since its conception in 2006. Its goal is to provide strong bounds on the amount of information being released from a dataset. Previous attempts were not mathematically rigorous and did not reliably prevent attacks. Most notably, before differential privacy, researchers were able to reidentify users in a Netflix dataset given an auxiliary dataset from IMDB and form a generalized attack against the state-of-the-art privacy algorithms of the time [3]. The strong privacy guarantee of differential privacy, on the other hand, has a rigorous mathematical foundation that makes it impervious to the post-processing attacks that exploited the Netflix dataset, and more recently, AirBnB and Instagram. As our world becomes more and more data-driven, it becomes ever more important to protect privacy in a sturdy way.

However, just building a suite of differential privacy algorithms is not satisfactory. We are motivated by the need to build tools that make the analyst's life easier, and an analyst who is not well-versed in privacy would be daunted by the mountains of differential privacy algortihms out there. Privacy adds a new layer complexity to the data analyst's job because the performance of algorithms may vary wildly depending on the database or other input parameters. This leads to the central problem of this work:
Problem 1. *How can a data analyst make the correct algorithmic choice given that a many tasks may be solved by more than privacy algorithm?*

This problem is noted by the authors of DPComp [1], who comment that currently, "the practitioner is lost and incapable of deploying the right algorithm". A data analyst should not responsible for making this choice. Little research has been conducted on how to help make the proper algorithmic choice; the most related work so far has been on visualization tools such as DPComp. However, we take a programming-language based approach for the following reasons:

- **Abstraction** Abstracting privacy in a programming language allows for the suppression of complicated privacy machinery, allowing programmers to view privacy as a black box. Programmers write code faster with fewer bugs when they can think about complicated ideas as black boxes.
- **Generalization** It is infeasible to conduct research on all use cases that a programmer may encounter. Writing a sophisticated programming language that automatically makes this choice would perform on par with the state-of-the-art methods. Effectively, we would allow anyone to stand on the shoulders of the giants who invented these algorithms.

```
def f(D):
    NoisyIf(g D):
        do A
    else:
        do B
```

Figure 1: Example `NoisyIf` statement.

The programming language could be termed *privacy agnostic* because it allows programmers and data practitioners to be agnostic to the advanced studies that have gone into developing differentially-private algorithms yet still attain the level of performance of the highest-grade algorithms in the field. Because the programming language will automatically make decisions during execution in an attempt to maximize accuracy, we call it `Jostle`.

The most central idea in `Jostle` is the `NoisyIf` statement, illustrated in Figure 1. A programmer will use `NoisyIf` when they are unsure of which choice to make at a certain point in their code. If they have certain beliefs that may aid `Jostle` in making the choice, then they may specify an *advice* function (denoted by `g` in Figure 1). Jostle will combine this advice along with previous executions of this particular `NoisyIf` to make the best decision.

We will begin with a background section on differential privacy and related work. Then we will introduce decision trees and highlight the existence of Problem 1 with them. Thirdly, we will solve the decision tree problem with different `NoisyIf` statements, and finally, we will illustrate how `NoisyIf` statements can be generalized to make `Jostle`.

2 Background

2.1 Differential Privacy

Differential Privacy is based on the intuition of a user being asked to participate in a study involving an algorithm $\mathcal{A}$ being run on a database D. If $\mathcal{A}(D)$ changes as a result of the user participating, then this poses a privacy concern. An attacker may be able to infer something about the participant's input. This means that non-trivial deterministic algorithms are already unacceptable; their output may change depending on just a single addition to

D. The strength ϵ of a differential privacy guarantee is the maximum factor that the randomized output of $\mathcal{A}$ changes for two databases D and D' which differ in one row. Mathematically, we are saying:

Definition 1. *$\mathcal{A}$ satisfies ϵ-differential privacy if for all D and D' such that $|D - D'|_1 = 1$ and for all o in the range of $\mathcal{A}$,*

$$\Pr\left(\mathcal{A}(D) = o\right) \leq e^{\epsilon} \Pr\left(\mathcal{A}(D') = o\right)$$

There is also a weaker definition:

Definition 2. *$\mathcal{A}$ satisfies (ϵ, δ)-differential privacy if for all D and D' such that $|D - D'|_1 = 1$ and for all o in the range of $\mathcal{A}$,*

$$\Pr\left(\mathcal{A}(D) = o\right) \leq e^{\epsilon} \Pr\left(\mathcal{A}(D') = o\right) + \delta$$

For much of this paper, we will focus on ϵ-differential privacy, but it is worth knowing the more general case so we can import the well-known privacy theorems in their most general form. Below is perhaps the most important result, and its proof comes cleanly from the definition of differential privacy.

Theorem 1. *(Post-Processing) If $\mathcal{A}$ satisfies (ϵ, δ)-differential privacy, and F is any function that takes the output of $\mathcal{A}$ as input, then $F(\mathcal{A})$ satisfies (ϵ, δ)-differential privacy.*

This theorem is the reason why differential privacy is such a useful guarantee. Data analysts can be sure that once they run their algorithm $\mathcal{A}$ and release its output, then the differential privacy guarantee gets no weaker *no matter what an adversary does with the data.* This prevents the headaches where an analyst realizes retroactively that the data he released can be combined in some way to reveal much more information than was intended. Another useful, intuitive result is:

Theorem 2. *(Composition) Given k algorithms M_1 and M_2 satisfying ϵ_1 and ϵ_2 differential privacy, respectively, along with a database D, the algorithm $M = (M_1(D), M_2(D))$ has $(\epsilon_1 + \epsilon_2, \delta_1 + \delta_2)$ differential privacy.*

So, what's a simple example of a differential privacy algorithm? Suppose each row of our database D is 0 or 1, so $D \in \{0, 1\}^n$, and that we are trying to release the sum of the elements of D. If this sum is S, then all neighboring databases D' have sum S or $S + 1$. We can add noise to S so that it looks very similar in distribution to $S + 1$. The distribution we are looking for is the Laplace distribution:

Definition 3. *The Laplace(λ) distribution has probability mass function $f(x) = \frac{1}{2\lambda} e^{-|x|/\lambda}$.*

This distribution fits perfectly with the definition of differential privacy because of the exponentials. If X, Y are i.i.d. from Laplace $\left(\frac{1}{\epsilon}\right)$, then it is straightforward to show that the distributions of $S + X$ and $S + 1 + X$ satisfy $(\epsilon, 0)$ differential privacy. To generalize this statement, we will use the following definition:

Definition 4. *(Sensitivity) A function f is Δ-sensitive if for all x, y such that $|x - y|_1 = 1$, we have*

$$|f(x) - f(y)| \leq \Delta$$

This can equivalently be rephrased as

$$\max_{|x-y|_1=1} |f(x) - f(y)| = \Delta$$

This gives us the following mechanism:

```
Algorithm 1: Laplace Mechanism
Input  : $D$, a database; $f$, a function; $\Delta$, sensitivity of $f$; and $\epsilon$
Output: An estimate for $f(D)$ satisfying $\epsilon$ differential privacy.
1 $X \sim$ Laplace $\left(\frac{\Delta}{\epsilon}\right)$;
2 return $X+f(D)$
```

Theorem 3. *The Laplace Mechanism 1 satisfies $(\epsilon, 0)$ differential privacy.*

For the counting or histogram queries such as our example above, we have $\Delta = 1$ so we add Laplace $\left(\frac{1}{\epsilon}\right)$ noise to our function.

2.2 Related Work

Perhaps the best-known example of a language that attempts to help practitioners use differential privacy is `PINQ` [2]. `PINQ` builds off Microsoft's C-sharp LINQ database system and provides an interface between the database and the programmer that can accomplish simple differential-privacy mechanisms. `PINQ` makes extensive use of the Laplace Mechamism 1 to noise histogram queries such as `Count`, `Average`, and `Median`. It also uses Composition 2 extensively when many of these queries are executed, and it attempts to abstract the composition into a `PINQAgent` class which keeps track of privacy budget. For example, the `NoisyCount` function is implemented in Figure 2 The functionality of PINQ is limited, although a lot of code

```
double NoisyCount(double epsilon){
    if(myagent.apply(epsilon)){
        return mysource.Count() + Laplace(1.0/epsilon);
    }else{
        throw new Exception("Access Denied")
    }
}
```

Figure 2: NoisyCount Implemented in PINQ.

can still be written, and McSherry provides k-means and Social Networking examples to prove its power. Also, the interface is a step in the right direction of thinking about privacy as a black box. However, the drawback of `PINQ` is that an analyst must still think about noise at every step of the computation. If an `Access is denied` error is thrown or the results are unacceptably noisy, the analyst will have no idea how to fix their code without diving into privacy.

Several languages have been built off PINQ to try to increase its functionality. wPINQ [4] uses

2.3 Decision Trees

3 Problem Setup

Theorem 4. *The entropy function on disjoint histogram counts $a_1, a_2, \dots, a_n$ has sensitivity*

Proof. Let $A = \sum_{i=1}^n a_i$. Then, the entropy is

$$\sum_{i=1}^n \frac{a_i}{A} \log\left(\frac{A}{a_i}\right) = \frac{1}{A}\sum_{i=1}^n a_i \log A - \frac{1}{A}\sum_{i=1}^n a_i \log(a_i) = \log(A) - \frac{1}{A}\sum_{i=1}^n a_i \log(a_i)$$

Suppose bucket a_j is reduced by 1, and the entropy change is

$$\log(A) - \log(A-1) - \frac{1}{A}a_j\log(a_j) + \frac{1}{A-1}(a_j-1)\log(a_j-1)$$
$$\leq \frac{1}{\ln(2)(A-1)} - \frac{1}{A}(a_j-1)\log(a_j-1) + \frac{1}{A-1}(a_j-1)\log(a_j-1)$$
$$= \frac{1}{\ln(2)(A-1)} + \frac{1}{A(A-1)}(a_j-1)\log(a_j-1) \leq \frac{1}{\ln(2)(A-1)} + \frac{1}{A}\log(A)$$

$\square$

Upon analyzing the Decision Tree paper, several questions are left unanswered:

- How would one generate the advice function in general?
- Is their decision to stop early if the database is small optimal? Can privacy be saved? Can accuracy be improved?

References

[1] Michael Hay, Ashwin Machanavajjhala, Gerome Miklau, Yan Chen, Dan Zhang, and George Bissias. Exploring privacy-accuracy tradeoffs using dpcomp. In *Proceedings of the 2016 International Conference on Management of Data*, SIGMOD '16, pages 2101–2104, New York, NY, USA, 2016. ACM.

[2] Frank McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. *Commun. ACM*, 53(9):89–97, September 2010.

[3] Arvind Narayanan and Vitaly Shmatikov. How to break anonymity of the netflix prize dataset. *CoRR*, abs/cs/0610105, 2006.

[4] Davide Proserpio, Sharon Goldberg, and Frank McSherry. Calibrating data to sensitivity in private data analysis: A platform for differentially-private analysis of weighted datasets. *Proc. VLDB Endow.*, 7(8):637–648, April 2014.