Differential Privacy: A survey

Jiaxuan Gao
gaojx19@mails.tsinghua.edu.cn

Ruoyu Yan
ry-yan19@mails.tsinghua.edu.cn

Wenda Chu
chuwd19@mails.tsinghua.edu.cn

June 3, 2021

1 Introduction

Over the past several decades, immense amount of data were collected, which enables a variety of new applications and services. Some of these applications investigate user behaviors and gain economic profits from it (such as recommendation algorithms); while some of them get access to crucial information such as health condition or medical data. As a result, it has been a growing concern to guard the privacy of users and protect sensitive data from exposure.

Among various approaches, Differential Privacy is considered as one of the most promising privacy preservation techniques. An elegant definition of privacy is proposed by [6] and several basic mechanisms are introduced as building blocks toward privacy ([6, 10]). In this report, we make a concise introduction to these techniques and briefly discuss several applications of differential privacy.

1.1 Basic Definitions

Model setup

A complete model of differential privacy consists of:

- A database $\mathcal{X} \in X^n$.
- An adversary may ask queries: $f : X^n \to Y$.
- The database is maintained by a trusted curator $M : X^n \to Y$. It is supposed to respond the queries with data protected.

Definition 1.1 (ϵ, δ-Differential Private)**.** An algorithm M is ϵ, δ-Differential Private if: $\forall \mathcal{X}, \mathcal{X}' \in X^n$ such that $\|\mathcal{X} - \mathcal{X}'\|_0 \leq 1$, and $\forall T \subseteq Y$

$$\Pr[M(\mathcal{X}) \in T] \leq e^{\epsilon} \Pr[M(\mathcal{X}') \in T] + \delta \tag{1}$$

M is called ϵ-pure differential private if $\delta = 0$. If $\delta \neq 0$, M is approximate differential private, in the sense that it is guaranteed to be differential private with probability $1 - \delta$.

This definition captures the concept of privacy by examining two *neighbouring database* $\mathcal{X}, \mathcal{X}'$ with some distance metric $d(\mathcal{X}, \mathcal{X}') \leq 1$. Intuitively, differential privacy press demands on algorithm M to be *" continuous"* in the whole X^n space. We shall also point out that the distance of two database are measured by Hamming distance (ℓ_0 norm) in the above definition, so two neighbouring database are only different on one entry. However, it is shown $d(\mathcal{X}, \mathcal{X}')$ could be measured using any other distance metric according to [3].

Definition 1.2 (Sensitivity)**.** Consider a function $f : X^n \to \mathbb{R}^k$. Its ℓ_p-sensitivity is defined as:

$$\Delta_p^{(f)} = \max_{\mathcal{X},\mathcal{X}':\|\mathcal{X}-\mathcal{X}'\|_0\leq 1} \|f(\mathcal{X}) - f(\mathcal{X}')\|_p. \tag{2}$$

Definition 2 takes a maximum for $\mathcal{X}, \mathcal{X}'$ spanned on the whole X^n space. It may facilitate analytic calculation for finding an upper bound, but in practice, the curator is only required to protect a single database $\mathcal{X}$. This provides incentive to develop definition of local sensitivity, which will be discussed in later sections.

1.2 Properties of Differential Privacy

Theorem 1.3 (Immunity to post-processing)**.** If $M : X^n \to Y$ is (ϵ, δ)-differential private, then for any function $h : Y \to Y'$, $h \circ M : X^n \to Y'$ is (ϵ, δ)-differential private as well.

Proof. It suffices to prove for deterministic h. If not, h can be decomposed into convex combination of deterministic functions. Moreover, by definition, a convex combination of DP mechanisms is still DP.

The proof for deterministic h is straightforward. Fix $\mathcal{X}, \mathcal{X}'$ with $\|\mathcal{X} - \mathcal{X}'\|_0 \leq 1$. Then for any $T \subseteq Y'$, define $S = \{y \in Y : h(y) \in T\}$.

$$\Pr[h(M(\mathcal{X})) \in T] = \Pr[M(\mathcal{X}) \in S] \leq e^\epsilon \cdot \Pr[M(\mathcal{X}') \in S] + \delta = e^\epsilon \cdot \Pr[h(M(\mathcal{X}')) \in S] + \delta. \tag{3}$$

This property is **crucial** for the idea of differential privacy to make sense; because an adversary is capable for doing any post-processing after querying $f(\mathcal{X})$.

Theorem 1.4 (Basic Composition)**.** Suppose algorithm M_i is (ϵ_i, δ_i)-DP, then a sequence of algorithms $M = (M_1, M_2, \ldots, M_k)$ is (ϵ, δ)-DP with $\epsilon = \sum_{i=1}^k \epsilon_i$ and $\delta = \sum_{i=1}^k \delta_i$.

This is in fact not a tight bound for DP composition. An advanced version of composition bound claims (ϵ, δ)-DP for composed algorithms with:

$$\tilde{\epsilon} = \epsilon\sqrt{2k\log(1/\delta')} + k\epsilon\frac{e^\epsilon - 1}{e^\epsilon + 1}, \tilde{\delta} = k\delta + \delta'. \tag{4}$$

The composition property of differential privacy captures the behavior of an adversary who asks for data in a sequential manner. DP properties are guaranteed to be preserved under polynomial number of sequential queries.

2 Basic Mechanisms

In this section, we briefly review the basic mechanisms that serve as building blocks of differential privacy. The main idea of these mechanisms is to apply minimal noise of the output of $f(\mathcal{X})$, meeting the demand of DP with relatively small accuracy loss.

Laplacian Mechanism

First define Laplacian density distribution $Lap(b) = \frac{1}{2b}e^{-\frac{|x|}{b}}$. Given any function $f : X^n \to \mathbb{R}^k$ with ℓ_1-sensitivity is $\Delta_1^{(f)}$, the Laplacian Mechanism gives:

$$M_\epsilon(x) = f(x) + (Y_1, Y_2, \ldots, Y_k) \tag{5}$$

where Y_i is i.i.d. variable drawn from Laplacian distribution: $Y_i \sim Lap(\frac{\Delta_1^{(f)}}{\epsilon})$.

Theorem 2.1. Laplacian Mechanism achieves $(\epsilon, 0)$-differential privacy with error bounded by $O(\frac{\Delta_1^{(f)}}{\epsilon})$.

Gaussian Mechanism

Gaussian mechanism is applied to any function $f : X^n \to \mathbb{R}^k$ with bounded ℓ_2-sensitivity $\Delta_2^{(f)}$. The same noise perturbation function as equation 5 is used for Gaussian mechanism. The only difference is that Y_i is i.i.d. sampled from a Gaussian distribution:

$$Y_i \sim \mathcal{N}(0, 2\ln(1.25/\delta)\Delta_2^2/\epsilon) \tag{6}$$

Gaussian mechanism guarantees (ϵ, δ)-DP with requirement on ℓ_2 sensitivity of the query function. In high dimensional queries($k \gg 1$), ℓ_2 norm is $\sqrt{k}$ times smaller than ℓ_1 norm, in which case the superiority of Gaussian mechanism emerges.

Theorem 2.2. Gaussian mechanism is (ϵ, δ)-DP.

Exponential Mechanism

While Laplace mechanism and Gaussian mechanism are applied in the case of real-valued functions, exponential mechanism is devised for the case of discrete-valued functions.

The model setting is as following. We have a dataset of n individuals $\mathcal{X} \in X^n$, a finite set of objects H and a score function $s : X^n \times H \to \mathbb{R}$ for any dataset $\mathcal{X}$ and an object $h \in H$. The goal is to output an object $h \in H$ such that $s(\mathcal{X}, h)$ is as high as possible while preserving privacy.

Definition 2.3 (Sensitivity of a score function)**.** For a score function $s(\mathcal{X}, h)$, its sensitivity is defined as

$$\Delta_s = \max_{h \in H} \max_{\mathcal{X}, \mathcal{X}'} |s(\mathcal{X}, h) - s(\mathcal{X}', h)| \tag{7}$$

The exponential mechanism M_E on inputs $\mathcal{X}, H, s$ selects and outputs some object $h \in H$, where the probability of a particular h is selected is proportional to $\exp\left(\frac{\varepsilon s(\mathcal{X}, h)}{2\Delta}\right)$. The probability is similar to that in Laplace mechanism and it can actually be shown that exponential mechanism is a discrete case of Laplace mechanism under certain circumstance.

Theorem 2.4. Exponential mechanism is ϵ-DP.

Theorem 2.5. Let $OPT(\mathcal{X}) = \max_{h \in H} s(\mathcal{X}, h)$ be the highest score with respect to dataset $\mathcal{X}$. For a dataset $\mathcal{X}$, let $H^* = \{h \in H : s(\mathcal{X}, h) = OPT(\mathcal{X})\}$ be the set of best objects. Then

$$Pr\left[s(M_E(\mathcal{X}) \leq OPT(\mathcal{X}) - \frac{2\Delta}{\varepsilon}\left(\ln\left(\frac{|H|}{|H^*|}\right) + t\right)\right] \leq \exp(-t) \tag{8}$$

This theorem implies that exponential mechanism would output an object with error of roughly $O\left(\frac{\Delta \ln |H|}{\varepsilon}\right)$. Exponential mechanism could be applied in the scenario of selling digital goods, as shown in Section 10.1 of [7], and also in private PAC learning.

3 Common Techniques on Databases

In this part, we briefly introduce two common techniques in differential privacy, private multiplicative weights algorithm and sparse vector technique. While basic mechanisms could serve as building blocks in DP applications, they are not optimal in the sense that many intrinsic properties of problems are omitted, for example correlation between queries, leading to suboptimal privacy or accuracy guarantee.

3.1 Private Multiplicative Weights

Private multiplicative weights algorithm [8] aims to improve performance in the setting of k linear queries. Linear queries are of the form $q(\mathcal{X}) = \frac{1}{n}\sum_{x_i \in \mathcal{X}} q(x_i)$, which are common in nowadays databases. If we directly apply Laplace mechanism or Gaussian mechanism on the k queries, the sample complexity would be $n = O\left(\frac{|Q|\log|Q|}{\alpha\varepsilon}\right)$ and $n = O\left(\frac{\sqrt{|Q|\log|Q|\log(1/\delta)}}{\alpha\varepsilon}\right)$ respectively, which are both polynomial in $|Q|$.

Private multiplicative weights works in an offline manner, that is, all k linear queries are given in advance as a set Q. Build on success of multiplicative weights algorithm [1] in field of no-regret learning, private multiplicative weights could achieve sample complexity of $n = \tilde{O}\left(\frac{\log|Q|\sqrt{\log|\mathcal{X}|\log(1/\delta)}}{\alpha^2\varepsilon}\right)$, which is only logarithm of $|Q|$. By regarding regret as error and distributions over experts as "fake" datasets, we could generate a "fake" dataset $\hat{\mathcal{X}}$ that answers all queries in Q with error bounded by α simultaneously. To make it differential private, we apply exponential mechanism and Laplace mechanism at every step querying the dataset.

Notice that instead of adding perturbation on the answers trivially, in private multiplicative weights we generate a *synthetic* dataset $\hat{\mathcal{X}}$ and then answer all queries in Q according to $\hat{\mathcal{X}}$. It's a typical and nice application of differential privacy.

3.2 Sparse Vector Technique

Sparse vector technique [7] works in the scenario of *identifying which queries are large* in a online manner. This is a easier question than directly outputting the answers. Again if we directly apply Laplace or Gaussian mechanism, the sample complexity are roughly $O(k/\varepsilon)$ or $O(\sqrt{k}/\varepsilon)$. With sparse vector technique, we could solve it with roughly $n = O(\log k/\varepsilon)$ or $n = O(c\log k/\varepsilon)$ if the goal is to identify the first c large ones.

Idea of sparse vector is rather simple. We first set a threshold T and add some Laplace noise. With stream of queries incoming, first compute the answer with Laplace noise and then compare the answer to the threshold T. If T is larger, continue to the next query. Otherwise, identify it as a large one. If the goal is to identify the first c large ones, just repeat the process for c times.

4 Private Machine learning

Recent years, machine learning algorithms have achieved great success in many applications. These algorithms aim to extract information and learn the distribution from data, so they inherently depends on the massive use of data. Designing machine learning algorithm with dataset protected therefore becomes an intriguing topic.

4.1 Differential Private ERM

A typical setup for machine learning is to minimize a loss function $\mathrm{L}(f, \mathcal{D})$, where $\mathcal{D}$ stands for the real world data distribution of the problem we want to solve. Empirical risk minimization (ERM) is a well known and extensively explored method for optimizing the loss function. ERM trains the learner to minimize the loss function on a sampled dataset by gradient descent or any other techniques. It is believed that ERM would qualifies the learner for real world tasks as long as the learner generalizes well and the training dataset is somewhat representative.

Several approaches have been explored for privatizing ERM algorithm, including output perturbation[4] and gradient perturbation[2]. Yet another interesting idea is to perturb on loss functions. However, the

works on this perturbing method either requires exact minimization on training data[4] or convexity of loss functions[9].

Output Perturbation

We denote the loss function on training dataset as $\mathcal{L}(f(\theta), \mathcal{D}) = \sum_{i=1}^{n} \ell(f_\theta(x_i), y_i)$, where f_θ is the hypothesis function with parameter θ.

The brief idea of output perturbation is to add noise directly to the learnt parameters θ. Hence, a natural requirement to bound the sensitivity of $\mathcal{L}(f_\theta, \mathcal{D})$. A prevalent method is regularization that penalizes parameters for growing too big. Chaudhuri et al. [4] proved the following statement (we use a slightly different definition of loss function from the original paper):

Theorem 4.1. For a loss function with a differential and 1-strongly convex regularization function $N(\theta)$,

$$\mathcal{J}(f_\theta, \mathcal{D}) = \sum_{i=1}^{n} \ell(f_\theta(x_i), y_i) + \lambda N(\theta), \tag{9}$$

and if ℓ is convex, 1-Lipschitz, then the ℓ_2 sensitivity of $\mathcal{J}$ is at most $\frac{2n}{\lambda}$.

A Gaussian noise $\mathcal{N}(0, O(\frac{n^2 \log(1/\delta)}{\lambda^2 \epsilon^2}))$ is thus sufficient for (ϵ, δ) differential privacy.

Gradient Perturbation

Gradient perturbation is a current prevailing approach, in which a noisy form of gradient descent is applied. The results of this kind of privatization are based on previous analysis on projected SGD convergence. According to [12], for convex objective function f and projected SGD algorithm $\theta_{t+1} = \mathcal{P}_\mathcal{C}(\theta_t - \eta(t) G_t(\theta))$, where $\mathbb{E}(G_t(\theta_t)) = \nabla f(\theta_t)$ and $\mathbb{E}(\|G_t(\theta_t)\|_2^2) \leq G^2$, we can achieve

$$\mathbb{E}[f(\theta_T) - f(\theta^*)] = O\left(\frac{\|\mathcal{C}\|_2 G \log T}{\sqrt{T}}\right), \forall \theta_0 \in \mathcal{C} \tag{10}$$

Based on this theorem, Bassily et al. proposed a private SGD algorithm. Assuming loss function ℓ to be L-Lipschitz, it basically iterates the following procedure for n^2 times:

$$\theta \leftarrow \mathcal{P}_\mathcal{C}(\theta - \eta(t) \cdot (n \cdot \nabla \ell(f_\theta(x_i), y_i)) + b_t), \text{ where } b_t \sim \mathcal{N}(0, \sigma^2 \cdot \mathbb{I}_d). \tag{11}$$

where the noise parameter is chosen as $\sigma^2 = \frac{32 L^2 n^2 \log(n/\delta) log(1/\delta)}{\epsilon^2}$. ($d$ is the dimension of parameter space.)

Since $\mathbb{E}(\|G_t\|_2^2)$ can be bounded by $n^2 L^2 + d\sigma^2$, the accuracy of this algorithm is still guaranteed by:

$$\mathbb{E}[\mathcal{L}(f_{\theta_T}, \mathcal{D}) - \mathcal{L}(f^*, \mathcal{D})] = \tilde{O}\left(\frac{\|\mathcal{C}\|_2 \sqrt{n^2 L^2 + d\sigma^2}}{\sqrt{T}}\right) = \tilde{O}\left(\frac{\|\mathcal{C}\|_2 L \sqrt{d \log(1/\delta)}}{\epsilon}\right) \tag{12}$$

Theorem 4.2 (DP for private SGD)**.** The above private SGD algorithm achieves (ϵ, δ)-DP with n^2 iterations.

The proof of this theorem is straightforward by combining the Gaussian mechanism and the advanced composition of DP.

5 Different Definition of Sensitivity

As we have stated in section 1.1, the original definition of DP considers the worst case of the protected database $\mathcal{X}$, which may be an overkill since our ultimate goal is only to protect a specific database. This encourages the several looser definition of sensitivity, such as local sensitivity and smoothed sensitivity.

Definition 5.1 (Local Sensitivity)**.** Consider a function $f : X^n \to \mathbb{R}^k$. Its ℓ_p-local sensitivity is defined as:

$$\Delta_{LS}^{(f)}(\mathcal{X}) = \max_{\mathcal{X}':\|\mathcal{X}-\mathcal{X}'\|_0\leq 1} \|f(\mathcal{X}) - f(\mathcal{X}')\|_p, \tag{13}$$

However, this definition is highly ill-conditioned. For example, let f be a query that asks for the distance of the nearest two points in a database; and $A = (0, 0, 0)$ is a database with three entries. It is easy to check that $\Delta_{LS}^{(f)}(A) = 0$. However, as a neighbouring database, $A' = (0, 0, N)$ yields a much higher local sensitivity $\Delta_{LS}^{(f)}(A') = N$. As such, bounding the local sensitivity on only one database instance may not be secure, since a small number of modifications may perturb its sensitivity a lot. Instead, it is necessary to bound the sensitivity of all databases close to $\mathcal{X}$. Proposal-test-release algorithm [5] is proposed to address this issue.

Proposal-Test-Release Algorithm

Let f be a query function from the adversary, Proposal-Test-Release algorithm attempts to find a proper upper bound for local sensitivity of f. Then it outputs $f(\mathcal{X})$ with noise, the magnitude of which is decided by the calculated upper bound.

Specifically, it first proposes such a bound β; and computes the Hamming distance γ from $\mathcal{X}$ to the nearest database $\mathcal{X}'$, such that $\Delta_{LS}(\mathcal{X}') \geq \beta$.

Compute $\tilde{\gamma} = \gamma + Lap(\frac{1}{\epsilon})$. The reason of taking noise on γ is that γ may reveal some information about the database $\mathcal{X}$ as well. A trick here is that the measurement γ on Hamming distance is a function with sensitivity 1. Finally, test where $\tilde{\gamma}$ is greater than $\ln(1/\delta)/\epsilon$. If no, β is too small to be a proper upper bound, so the algorithm returns $\perp$.

Otherwise, outputs $f(\mathcal{X}) + Lap(\beta/\epsilon)$. This is again $(\epsilon, 0)$-differential private by Laplacian mechanism. As a combination, it is proved that:

Theorem 5.2. The proposal-test-release algorithm is $(2\epsilon, \delta)$-DP.

Though the privacy guarantee of the above theorem is promising, its running time is not properly guaranteed. We shall point out that the step of finding γ is not computationally efficient. However, some common queries enjoy great properties of such efficiency. The task of calculating the histogram of a database $\mathcal{X} \in X^n$ shows the strength of this algorithm. For comparison, a pure ϵ-DP scheme attains $O(\log |X|/\epsilon)$ accuracy on histogram queries. However, by testing local sensitivity, an (ϵ, δ)-DP algorithm achieves $O(\log(1/\delta)/\epsilon)$ accuracy, which cancels the dependency on domain size $|X|$.

Yet another similar variation of sensitivity (Smoothed sensitivity) is defined by Nissim et al in [11]. Intuitively, this definition suggests putting more attention on databases closer to $\mathcal{X}$ with an exponential discount. In practise, however, it is often intractable for computation.

$$\Delta_{SS}^{(f)}(\mathcal{X}) = \max_{\mathcal{X}'\in X^n} \left(\Delta_{LS}^{(f)}(\mathcal{X}')e^{-\epsilon d(\mathcal{X},\mathcal{X}')}\right) \tag{14}$$

6 Summary

As we shown in previous sections, differential privacy has been proposed as a mathematical tool to protect individual's private information while preserving possibility to do analysis in large scale datasets. In this survey, we briefly talk about several basic techniques that serve as building blocks in differential privacy applications, for example, the Laplace mechanism. The privacy is guaranteed by the property of immunity to post-processing. With basic or advanced composition, we could analyze more complicated algorithms. In DP's world, the goal is to generate less statics error within certain privacy guarantee so designing different algorithms for special kinds of problems is important. In modern world, several Internet companies like google have already started to explore methods to prevent privacy. As far as we know, in practice, differential privacy is now deployed in some database applications. The queries in databases are usually simple. For example,

google has an open-source repository that contains libraries to generate ε- and (ε, δ)-differentially private statistics over datasets (see https://github.com/google/differential-privacy). However, for more complicated applications like machine learning or deep neural network, it's not practical to deploy differential privacy now. First, as we have shown in the part of machine learning, the model setting introduces too strong restrictions, which are also common in nowadays analysis. Second, to successfully deploy differential privacy in a real-world application, differential privacy experts are always needed, which further prevents the deployment of differential privacy.

References

[1] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(6):121–164, 2012.

[2] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 464–473, 2014.

[3] Konstantinos Chatzikokolakis, Miguel E. Andrés, Nicolás Emilio Bordenabe, and Catuscia Palamidessi. Broadening the scope of differential privacy using metrics. In Emiliano De Cristofaro and Matthew Wright, editors, *Privacy Enhancing Technologies*, pages 82–102, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[4] Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. Differentially private empirical risk minimization, 2011.

[5] Cynthia Dwork and Jing Lei. Differential privacy and robust statistics, 2009.

[6] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *heory of Cryptography*, pages 265–284, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[7] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407, August 2014.

[8] Moritz Hardt and Guy N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 61–70, 2010.

[9] Roger Iyengar, Joseph P. Near, Dawn Song, Om Thakkar, Abhradeep Thakurta, and Lun Wang. Towards practical differentially private convex optimization. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, pages 299–316. IEEE, 2019.

[10] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. IEEE, October 2007.

[11] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. A.: Smooth sensitivity and sampling in private data analysis. In *In: Proc. of STOC, ACM (2007) 75–84*.

[12] Ohad Shamir and Tong Zhang. Stochastic gradient descent for non-smooth optimization: Convergence results and optimal averaging schemes, 2012.