Analyses of Differential Privacy's Precision and Privacy Relationship

Qiwei Zhao

ShangHai JiaoTong University

May 2018

Abstract. In the data time, as electronic data about individuals becomes increasingly detailed, and as technology enables ever more powerful collection and curation of these data. The protection of people's data is becoming a very hot topic, and the differential privacy is a very important technology to protect people's privacy. In the traditional circumstances, we assumed a fixed privacy requirement ε, then we try to maximize the accuracy of the computation subject to the privacy constraint. But this sequence has some problems under some certain circumstances. For example, when we use the medical data to train the model, the function of which is to predict Organ lesions, in this circumstance, the most important things are the accuracy, rather than the privacy. So, in this paper we estimate the relation among ε, privacy level and the accuracy, and attempt to achieve an algorithm that a fixed accuracy requirement is given and and the data analyst would like to maximize the privacy subject to the accuracy constraint. Moreover, we carry out those algorithms in some practical circumstances, and compare them.

Keywords: Differential Privacy, Data Analyze, Accuracy, privacy level, Algorithm

1 Introduction

We live in a big data time, and we use these data for many purposes, such as data analyze and machine learning, these now technologies present many opportunities for enhanced services and products in diverse areas such as healthcare, safety, online search, and so on. However, its use is hindered by concerns regarding the privacy of the individuals whose data are being used. People worried about that their data would be disclosed, they may be harassed because of this disclosure. So, we use the technology called "differential privacy" to protect people's sensitive data. There are much more recent set of large-scale practical deployments, including by Google and Apple. To protect data, differential privacy uses an output perturbation technique which adds random noise to the outputs. The magnitude of noise to add, which determines the degree of

privacy, depends on the type of computation and it must be large enough to conceal the largest contribution that can be made to the output by one single individual. To be specific, let X be a database to release statistics about and f be a query function. differentially private mechanism gives perturbed response f(X)+Y instead of the true answer f(X), where Y is the random noise. While this seems a perfect solution, the amount of noise needed to achieve indistinguishability between two datasets generally eliminates any useful information. But there are still many problems about this technology needed to be solved, first ε does not easily relate to practically relevant measures of privacy, we need to determine the relation between them. What's more After knowing the relation, we attempt to design the accuracy-first method's steps and give the background about them. In the ending part of this paper we do some experiments which includes the performance of the different algorithm under the Laplace mechanism of output perturbation then performs an optimization using the noisy data. Our experiments concentrate on a practical circumstance, and we will compare two method's performance.

2 Differential Privacy Background

2.1 Differential privacy

Differential privacy is a mathematical definition for the privacy loss that results to individuals when their private information is used in the creation of a data product. So, we have this following definition, that Let ε be a positive real number and A be a randomized algorithm that takes a dataset as input (representing the actions of the trusted party holding the data). Let imA denote the image of A. The algorithm A is ε -differentially private if for all datasets D1 and D2 that differ on a single element (i.e., the data of one person), and all subsets S of imA has

Following property:

$\Pr[A(D) \in S] \leq \exp(\varepsilon)\Pr[A(D') \in S]$

where the probability is taken over the randomness used by the algorithm.

2.2 Ex-Post Privacy Loss

Given a randomized algorithm A, which map X To O, the definition of the ex-post privacy loss is listed as following

$$\text{Loss}(o) = \max_{D,D':D\sim D'} log \frac{\Pr[A[D]=o]}{\Pr[A(D')=o]}$$

2.3 Global Sensitivity

In a model which is interactive, users' queries come to the database and receive a noisy response which may not be accuracy, because the magnitude of noise is added to the response, which is determined based on the query function f. Sensitivity of that function represents the largest change in the output to the query function which is made by a single data. From these, we can drive the notation, global sensitivity.

The given query function, which will may D to R, the global sensitivity is definite as:

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

For ∀x, y differing in at most one element.

3 The Relation Among ε, Privacy Level and The Accuracy

We now display the relation among ε, privacy level and the accuracy, and how to select the value of ε, given the goal of hiding any individual's presence (or absence) in the database. Assuming that the privacy requirement for our example dataset is to limit the any individual's probability of being identified as present in the database to be no greater than one certain number. We show two ways of selecting a good choice of that guarantees the probability of identifying any individual's presence is no greater than the maximum tolerable value δ. One is to use the upper bound; the other is to search for the right value (reducing noise). We first consider how the principle upper bound on the adversary's probability can be used to enforce the requirement. Let ρ be the probability of being identified as present in the database. After some mathematical derivation, we finally find that satisfies the following inequality:

$$\varepsilon \leq \frac{\Delta f}{\Delta v} \ln \frac{(n-1)p}{1-p}$$

Note that the greater n and ρ are, the greater the minimum ε needed. Therefore, as the size of database to publish and the probability to bound get larger, less noise need be added, which is a very useable conclusion when made the decision to choose that parameter.

And in another method, things become a little different: Let X = {x1, x2,..., xn−1, v} and X∗ = {x1, x2,..., xn−1}. Without loss of generality, assume that elements are sorted in ascending order (i.e., x1 < x2 <... < xn−1 < v). Two databases X and X∗ are identical except that only one element, v, is missing in X. If we divide numerator and denominator of β(ωi) by P (κf (ωi) = γ), after some mathematical derivation, the conclusion is listed following:

$$\beta(\omega i)) = \frac{1}{1+(n-1)e^{-\frac{\varepsilon \Delta v}{\Delta f}}}$$

To compare these two methods, we can see that both the methods for a differential private algorithm can solve for the smallest value of the differential privacy parameter (ε), and other

parameter values needed to be set. Run the private algorithm we can get the result ε. But because Upper Bound theorems tend to be worst-case bounds, this approach will generally be extremely conservative, leading to a much larger value of ε than it truly needed, and hence a much larger leakage of information than is necessary for the problem at hand, which is very bad and need to find way to avoided.

4 Differential Privacy Algorithms with Accuracy Requirements

4.1 Laplace mechanism algorithm

The key part of the differential privacy, and the most basic subroutine we will use is the Laplace mechanism. The Laplace Distribution centered at 0 with scale b is the distribution with probability density function, add the Laplace noise into the original database to get another noisy database

Main Code:

```
import math
import random
import numpy as np
import matplotlib.pyplot as plt
from collections import Counter
def laplace():
    start, scale = 0.0, 2.0
    sample = np.random.laplace(start, scale, 1000)
    vals = np.arange(-8.0, 8.0, 0.01)
    dFunc = np.exp(-abs(vals-start)/scale)/(2.0*scale)
    plt.plot(vals, dFunc)
    plt.show()
def generate_new_database (name_list):
    probTable = []
    dataBase = []
    count = 0
    index = 0
    for item in name_list:
        count+=item[1]
```

```
        probTable.append(count)
    for x in range(10000):
        y = random.uniform(0,1)
        for item in probTable:
            if y>item:
                index+=1
            else:
                dataBase.append(name_list[index][0])
                break
     index = 0
    return database
    def genLapN():
    start, scale = 0.0, 20.0
    sample = np.random.laplace(start, scale, 1000)
    listN = []
    for x in range(20):
        listN.append(sample[x])
    return listN


def add_noise():
    name_base = generate_new_database ()
    lapNum = genLapN()
    count = Counter(name_base)
    idx = 0
    for item in count:
        count[item]+=lapNum[idx]
        idx+=1
    comName = count.most_common(10)
    return comName[0][0]
```

4.2 The key parts of noise-reducing algorithm

4.2.1 Gradual private release

gradually releasing private data is a very important method when we carry out the noise reducing algorithm, by using the Laplace kernel mechanism with an increasing sequence of ε values. Seeing a privacy cost scaling with on the least private release, rather than the accumulation of the

privacy costs of independent releases. That's an extremely good things for us. Giving some examples, the algorithm can be depicted as a continuous random walk starting at a very private data point v with the property that the marginal distribution at each point in time is Laplace centered at v. Releasing the value of the random walk at a fixed point in time gives a certain output distribution, for example, v, with a certain privacy guarantee and then like walking toggle, you may go to a more private place or a more accuracy place. To produce v_0 whose ex-ante is more private, one can simply "fast forward" the random walk from a starting point of v to reach v_0; to produce a less private v_0, one can turn around. The total privacy cost is some certain place, given the least private point, all more private points can be derived by taking a random walk of a certain length starting at v, until you reach the least one. Note that were the Laplace random variables used for each release independent, the composition characteristic would require just summing the ε values after each release.

Main Code:

```
import numpy as np

def do_filt(prob, matrix):

f = lambda x: 0 if np.random.random() <= prob else x

  return np.vectorize(f)(matrix)

def gen_list(M, sensitivity, eps_list):

 try:

 steps = len(eps_list)

 except:

  steps = eps_list.shape[0]

  shape = M.shape

  rev_eps_list = eps_list[::-1]

  noise_list = [np.random.laplace(scale=sensitivity/eps, size=shape) for eps in rev_eps_list]

  walk = [M + noise_list[0]]

  filt_probs = [ (rev_eps_list[j] / rev_eps_list[j-1])**2 for j in range(1, steps) ]

  walk_steps = [do_filt(p, noise_list[1+j]) for j,p in enumerate(filt_probs)]

  for j in range(1, len(rev_eps_list)):

    walk.append(walk[j-1] + walk_steps[j-1])

  walk.reverse()

  return np.array(walk)
```

4.2.2 Over Threshold Method

This part takes the responsibility to terminate the program when the Over subroutine "gradual private release" already make the result satisfy our requirement. Overall approach to our eventual problem will be as following, generate a sequence of available parameters the 1, ……, T, each with increasing accuracy and decreasing privacy; then test their accuracy levels sequentially, output the first one whose accuracy is "good enough." The classical Over Threshold algorithm

takes in a dataset and a sequence of queries and privately outputs the index of the first query to exceed a given threshold.

Tousling Over Threshold to when to perform these accuracy checks. Unfortunately, there is a bad obstacle: for us, the “queries” themselves depend on the private data. Standard composition analysis would involve first privately publishing all the queries, then running Over Threshold on these now public queries. Though, until Over Threshold stop that produce, it would be much better to generate and publish the queries one at a time, at which point one would not publish any more queries. The problem is centered in speed of analyzing this approach that is we do not know when Over Threshold will terminate, when we need to analyze a large mount data, that’s can be a fatal drawback.

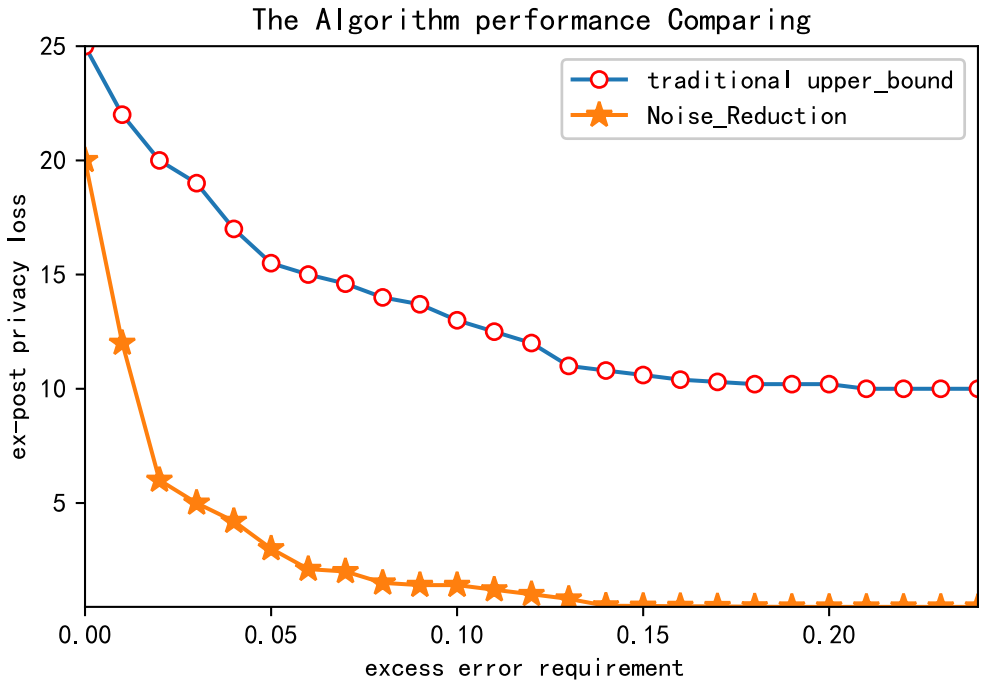
Main Code:

```
import numpy as np
import matplotlib.pyplot as plt
 return " ".join(list(map(str, arr)))
def Threshold (X, Y, lamb, alpha, gamma, max_norm, max_steps):
  n = len(X)
 dim = len(X[0])
  if max_norm <= 0.0:
    max_norm = logist.compute_max_norm(lamb)
  sv_sens = logist.get_sv_sensitivity(max_norm, n)
  opt_beta_sens = logist.compute_opt_sensitivity(n, dim, lamb)
  compute_err_func = lambda X,Y,beta_hat: logist.compute_err(X, Y, lamb, beta_hat)
 opt_beta, opt_res = get_opt(X, Y, lamb)
  opt_err = opt_res[0]
  for beta, res in [(output_beta_hat, output_res), (naive_beta_hat, naive_res)]:
    success, excess_err, sv_eps, my_eps, index = res
    two_norm = np.linalg.norm(beta)
    unreg_err = logist.compute_err(X, Y, 0.0, beta)
    print("1" if success else "0", excess_err, sv_eps, my_eps, index, two_norm, unreg_err)
    print(stringify(beta))
def parse_inputs(args):
  try:
    filename = args[1]
    lamb = float(args[2])
    alpha = float(args[3])
    gamma = float(args[4])
    max_norm = float(args[5])
```

```
max_steps = int(args[6])
X, Y = [], []
with open(filename) as f:
  for line in f:
    nums = [float(x) for x in line.split()]
    X.append(nums[:-1])
    Y.append(nums[-1])
X = np.array(X)
Y = np.array(Y)
return X, Y, lamb, alpha, gamma, max_norm, max_steps
```

5 Experiments and Comparison

To evaluate the noise reducing methods which we described above, we conducted practical evaluations. We use the practical data, which is downloaded from “http://ama.liglab.fr”, this dataset contains two different social networks: Twitter, a micro-blogging platform with exponential growth and extremely fast dynamics, and Tom’s Hardware, a worldwide forum network focusing on new technology with more conservative dynamics but distinctive features. In this experiment we use the Twitter part. We tested the algorithm’s average ex-post privacy loss vs a large range of input accuracy goals. The results show that noise reducing algorithm gives a large advance over the traditional upper bound approach, which just inverting utility theorems. Obviously, the improvement over traditional method is enormous, for instance, in this experiment, Noise Reduction has about 5 ex-post privacy loss, when the excess error requirement is about 0.05, but the other method has about 20 ex-post privacy loss. This is a very tremendous improvement. Out final result is showed following:



6 Conclusion

There is no doubt that the differential privacy has received considerable attention, and it has been deployed worldwide to protect users' data by the data giant like, Google, Apple, Facebook and so on. ε is the most important parameter for differential privacy, but it does not directly correlate to a practical privacy level. What's more, under a certain accuracy requirement, how to choose a suitable ε efferently is very meaningful. it is possible to determine an appropriate value of ε. We have shown that given an accuracy level, how to choose the ε to get a most high privacy level, up to now, there already are many different algorithm, which can achieve the goal, we choose two representatives, depict their detail, and compare their performances. The result show that the newer one—noise reducing algorithm has very large advance over the traditional one. In a word, any discussion of a differentially private mechanism requires a discussion of how to set an appropriate ε for that mechanism with users' requirement, it's will always be the hot topic in the field of differential privacy.

Acknowledgments. Thanks to Prof. Xinbing Wang, Prof. Luoyi Fu, and Prof. Xiaohua Tian. They are knowledgeable and passionate, giving me great help on the curriculum “Principles of Wireless Communications and Mobile Networks”. I got a lot of knowledge and new ideas in this area from them. As for me, there is no doubt that this course is a one of the greatest courses. At the same time, I also want to thank this course’s TA - Dr. Jiaqi Liu, she is a smart doctor and enthusiastic to help us. Thanks for her help.

Reference

[1] Frank McSherry, Kunal Talwar. Mechanism Design via Differential Privacy. Foundations of Computer Science, 2007. FOCS '07. 48th Annual IEEE Symposium on.

[2] Cynthia Dwork and Aaron Roth (2014), "The Algorithmic Foundations of Differential Privacy", Foundations and Trends in Theoretical Computer Science: Vol. 9: No. 3–4.

[3] Arik Friedman, Assaf Schuster. Data mining with differential privacy. KDD '10 Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining

[4] Kamalika Chaudhuri and Claire Monteleoni. Privacy-preserving logistic regression. In Advances in Neural Information Processing Systems 21, Proceedings of the Twenty-Second Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 8-11, 2008, pages 289–296, 2008

[5] Fragkiskos Koufogiannis, Shuo Han, and George J. Pappas. Gradual release of sensitive data under differential privacy. Journal of Privacy and Confidentiality, 7, 2017

[6] Adam Smith, Jalaj Upadhyay, and Abhradeep Thakurta. Is interaction necessary for distributed private learning? IEEE Symposium on Security and Privacy, 2017.
[7] Blum, A., Ligett, K., Roth, A.: A learning theory approach to non-interactive database privacy. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing

[8] Katrina Ligett, Seth Neel, Aaron Roth, Bo Waggoner, and Z. Steven Wu. Accuracy First: Selecting a Differential Privacy Level for Accuracy-Constrained ERM. TPDP 2017 - Theory and Practice of Differential, Privacy Dallas, TX, USA - October 30, 2017

[9] Jaewoo Lee and Chris Clifton How Much Is Enough?Choosing ε for Differential Privacy. Information Security: 14th International Conference, ISC 2011

[10] C.Dwork. The differential privacy frontier. Theory of Cryptography Conference, 2009