Implementing Differential Privacy Using Randomized Response Algorithms

Benjamin Steephenson
Tufts University, Introduction to Computer Security

December 13, 2017

Contents

Abstract

Differential privacy presents a way to measure the extent to which an algorithm is privacy preserving. Randomized response is a data collection method that ensures that only a noisy version of the true dataset is collected. This paper explores the need for a rigorous measure of privacy, the definition and properties of differential privacy, and how randomized response algorithms can be used to implement differential privacy.

Introduction

With the rise of big data and consumer facing machine learning, systems designers and data scientists must be cautious regarding the ways that sensitive private information about individual users can leak out of products and queries. Companies often collect much data about their users and their activity in order to train machine learning models and compute aggregate statistics to improve their products for all users. A problem with this is that it creates a pathway from one users' data to another user. How do we actually know if a sophisticated attacker can't use these machine learning models and aggregate statistics to uncover private data? How can we prove that something is actually privacy preserving?

Our intuitive notions of privacy lack rigor and aren't future proof. Consider the case of data anonymization. In one incident, a publicly released and anonymized health dataset was exploited to uncover the health records of individuals [1]. Because the dataset contained each patient's date of birth, gender, and zip code, a researcher was able to uncover the governor's health records by cross referencing against public voter registration records. Another example of this is the Netflix competition dataset. Here, attackers were able to deanonymize publicly released Netflix user data by cross referencing against an IMDB dataset [1].

Aggregate statistics can seem benign, but they too can be exploited to uncover private data. For example, suppose there existed a database of users and their incomes. If an attacker wanted to find the income of user 42, they can query the database for '(mean income of all users) - (mean income of all users except user 42)'. Alternatively, they could also query for '(mean income of all users)' and then later query '(mean income of all users except user 42)'. These example queries infer a specific user's data indirectly via aggregate statistics. Notable in the latter case, neither query leaks private data, but the combination of the two queries does. Even if the database designer were to add noise to the output of aggregate queries to prevent this kind of attack, an attacker could repeatedly make both queries and use the results to infer the true means. Simply adding noise to the query output is insufficient.

A fundamental question worth asking is how can we predict with some certainty that an anonymization structure, aggregate statistic, or any algorithm can't be used to reveal underlying private data. Thankfully, differential privacy provides a framework for measuring how privacy preserving an algorithm is, letting practitioners make the most of useful datasets without compromising the confidentiality of any single data point.

To the community

Privacy of user data ought to be of primary concern to statisticians and machine learning practitioners. It is hard to predict beforehand how private data can leak into a model that trains on it and how querying that model can leak model parameters. Furthermore, with studies that require external verification and replicability, it is hard ensure that one has appropriately anonymized and randomized data before making it publicly available. Differential privacy gives us a set of tools to approach these questions with some degree

of certainty.

In this paper, I specifically delving into implementing differential privacy using randomized response methods. By only collecting a noisy derivative of the true user data, practitioners never really have access to the private data they are trying to protect. This promotes user trust by giving users plausible deniability of the collected data, and these methods demonstrate the ways we can protect users at the expense of accuracy.

Differential privacy

Differential privacy gives us a way to measure how privacy preserving a randomized algorithm is. One way to describe the privacy of an algorithm is the extent to which one individual's contribution can affect the ultimate results. More formally, let M be a randomized algorithm. M is ϵ-differentially private if for any $s \in range(M)$ and for any pair of databases $d, d' \in domain(M)$ that differ by only one row, $\frac{P(M(d)=s)}{P(M(d')=s)} \leq e^{\epsilon}$ [2]. The more unlikely it would be to get the same result with a slightly perturbed database, the larger we would need to make ϵ.

Suppose we had a database of users' heights. Let us consider the differential privacy of a max finding algorithm MAX that finds the height of the tallest person in our database. Further suppose that the height of each individual in the database is unique. Suppose that user 42 was the tallest user with height 7 ft. We can consider how much the result of MAX would change if we removed user 42. Note that it would be impossible to achieve the same result of 7 ft if we removed user 42. Using the definition of ϵ-differential privacy, the ratio $\frac{P(M(\text{database with user 42})=7)}{P(M(\text{database without user 42})=7)}$ is unbounded because the denominator is 0 (because all users' heights are unique). Consequently, MAX is not ϵ-differentially private for any value of ϵ. This make intuitive sense because the MAX query reveals an entire data point in our database.

What if we added some noise to the result of MAX to create $NOISYMAX$? Consider the Laplacian distribution defined by

$$L(x|\mu, b) = \frac{1}{2b} e^{\frac{|x-\mu|}{b}}$$

We can define $NOISYMAX(d) = s$ where s is drawn from $L(MAX(d), b)$ for some noise parameter b. In other words, we take the regular result of MAX and add some Laplacian noise. Now for any value s, $P(M(\text{database with the tallest user}) = s) = L(s|\text{height of the tallest user}, b)$ and $P(M(\text{database without the tallest user}) = s) = L(s|\text{height of the second tallest user}, b)$. Within reason, we can add the restriction that users' heights all fall in the interval $[0, 10]$ measured in feet. Therefore, in the most extreme case, the height of the tallest user could be 10 ft, and the height of the second tallest user could be 0 ft. Using the definition of ϵ-differentially privacy, we can see that

$$\frac{P(NOISYMAX(d) = s)}{P(NOISYMAX(d') = s)} \leq \frac{L(s|10, b)}{L(s|0, b)} = \frac{\frac{1}{2b} e^{\frac{|s-10|}{b}}}{\frac{1}{2b} e^{\frac{|s-0|}{b}}} = e^{\frac{|s-10|}{b} - \frac{|s-0|}{b}} \leq e^{\frac{10-0}{b}} = e^{\epsilon}$$

where $\epsilon = \frac{10}{b}$. Thus, we have an ϵ-differentially private algorithm!

Since smaller values of ϵ yield more privacy, increasing b adds more noise to our results and further increases the privacy of the algorithm. However, this also decreases the usefulness of our results as $NOISYMAX$ can yield values that are far from the true maximum value. This demonstrates the inherent tradeoff between privacy and accuracy in implementing differential privacy preserving algorithms.

There are some key results relating to differential privacy worth noting here. First of all, now that we have an estimate for the maximum value of our database, doing anything with this value can't further leak privacy. A generalization of this is the post-processing theorem which states the composition of any randomized algorithm with an ϵ-differentially private algorithm is also an ϵ-differentially private algorithm [3].

However, if an attacker were to query $NOISYMAY(d)$ repeatedly, they could use the mean of all these randomized results to approximate the true maximum value, thereby exposing a data point of the database. This risk is a specific example of the combination theorem which states that if we have n functions $f_1, ...f_n$ each of which maintain differential privacy with epsilons $\epsilon_1, ...\epsilon_n$, then the function $f(x) = (f_1(x), ...f_n(x))$ maintains differential privacy with $\epsilon = \epsilon_1 + ...\epsilon_n$ [3]. In making the same query n times, we multiply ϵ by n, leaking n times as much privacy. A systems designer could implement a "privacy budget" and limit queries by ensuring that the sum of the corresponding epsilons doesn't exceed this budget.

Randomized response with coin flips

Randomized response is a differential privacy technique that promotes user trust and alleviates the repeated query problem. In this scheme, only a noisy version of user data is collected, creating a barrier between the true dataset and the collected dataset. Consequently, if an attacker were able to gain access or infer the collected dataset, this dataset is limited in its ability to be used to infer the true dataset.

For example, suppose we wanted to conduct a survey to measure what proportion of the population takes part in some illegal activity. If we just asked survey participants to respond truthfully, it would not be in the participants best interests to do so. The data collected in this survey could be used against them.

Suppose we used a randomized response scheme instead. We would ask the participant to secretly flip a fair coin. If the coin lands on heads, the user should respond truthfully. If the coin lands on tails, the participant should flip the coin again and respond Yes if it lands on heads or No otherwise. The person conducting the survey would not know or record the coin flip(s) and would only record the participant's response.

Although we deliberately added noise to the dataset, we can still use the this collected data to infer the parameter p = the proportion of people who take part in the illegal activity. The probability of a participant responding yes equals $(1/2) * p + (1 - (1/2)) * (1/2)$. We can then calculate the total likelihood of our dataset and find the value of p that maximizes this likelihood, thereby finding the maximum likelihood

estimate of p.

We can also show that this method satisfies our differential privacy condition. Here, our randomized algorithm M is the way we turn the true dataset into the collected dataset. We want to measure how less likely our collected dataset would be if the true dataset differed by one row. Suppose the true dataset d had values $x_1, x_2, ...x_i, ...x_n$. Our perturbed dataset d' would have values $x_1, x_2, ...\bar{x_i}, ...x_n$. Here, the ith row is the only one that changes. Suppose that a run of our randomized response collection scheme $M(d)$ yielded the value $Y = y_1, ...y_n$. We need to consider the ratio

$$\frac{P(Y|d)}{P(Y|d')} = \frac{P(y_1|x_1)...P(y_i|x_i)...P(y_n|x_n)}{P(y_1|x_1)...P(y_i|\bar{x_i})...P(y_n|x_n)} = \frac{P(y_i|x_i)}{P(y_i|\bar{x_i})}$$

Because of the independence of each row, we only need to look at the particular row that changed. For every value of y_i and for every value of x_i, we need to find the maximum of all these ratios and set it equal to e^ϵ. Through case analysis, we can see that the maximum value of the ratio is 3 [3]. Equating this to e^ϵ we find the our randomized response scheme satisfies ϵ-differential privacy with $\epsilon = ln\ 3$. This places a limit on how much we can infer about the true data points given the collected data points, and this gives the participants of the survey plausible deniability if the data were to leak. Each participant can claim that their yes or no response was due to a coin flip rather the truth.

Because of the post-processing theorem, any data analytics we do on this collected dataset can't create any further privacy leaks. In sacrificing accuracy by placing a noisy barrier between the true dataset and the collected dataset, we are still able to do useful data analysis without running the risk of leaking sensitive information. This method lets us make inferences about the population without every truly seeing any one individual's data. If responsibly communicated, this can also promote users' trust of the product as users can maintain plausible deniability for what exists in the collected dataset.

More sophisticated randomized response with RAPPOR

Moving beyond just binary data, we can extend this randomized response principle to collect categorical data and vectors of real values. Researchers at Google created a technology called RAPPOR [4] (Randomized Aggregatable Privacy-Preserving Ordinal Response) which uses randomized response to study user data without compromising the privacy of any individual user.

Suppose we wanted to collect a certain value from all users of a piece of software (for example, the age of the user). In this scheme, the client side software would hash the true value onto a bloom filter, creating a resultant bit vector A. Then, using a coin flip mechanism similar to the survey example, a new bit vector B would be created, and this bit vector would be memoized by the client. Finally, a third bit vector C would be created using a similar same coin flip mechanism. This is the bit vector that gets sent to the server. The next time the server asks for the same value, the client server creates another randomized version of B and sends that over.

If the client simply just used A to create B every time the server asked for the value, the server could potentially ask for the value too many times, creating a way for the server to infer the true A. That is why the client needs to memoize B. However, just sending over B every time is also insufficient because now we have created a unique identifier B for every user. This could be disasterous. To avoid this, the client would perform a second layer of randomized response using the memoized B.

Despite the amount of noise this algorithm adds to data points, when aggregated over large samples, RAPPOR can be used to gain meaningful insights about user data. This algorithm also maintains differential privacy for each user, even when querying for the same value repeatedly.

Action items

Machine learning practitioners, statisticians, and data collectors ought to

- Be aware that anonymization, aggregate statistics, and machine learning open up ways for private data to leak out of products and datasets
- Measure the differential privacy aspects of the algorithms they use to protect sensitive data
- Restrict how many queries one can perform using a privacy budget
- Consider using two level randomized response algorithms with memoization to avoid ever seeing the true user data in order to promote user trust and protect user data

Conclusion

It can be difficult to intuitively know how an attacker could use anonymized data or queries to infer private data about individuals. Thankfully, differential privacy gives us a way to measure how much of an individual's privacy can leak from a randomized algorithm. However repeatedly querying the dataset can leak private information as differential privacy degrades with the number of queries performed. A solution to this is to only collect a noisy version of the true user data via randomized response. The two level randomized response mechanism of the RAPPOR algorithm presents a way to collect user data over time without compromising any individual's privacy.

References

1 Klarreich, E. (2012, December 31). Privacy by the Numbers: A New Approach to Safeguarding Data. Scientific American. Retrieved December 13, 2017, from https://www.scientificamerican.com/article/privacy-by-the-numbers-a-new-approach-to-safeguarding-data/

2 Yuxiang, W. Differential Privacy: a Short Tutorial. Retrieved from www.cs.cmu.edu/ yuxiangw/docs/Differential

3 Dwork, C., & Roth, A. (2014). The algorithmic foundations of differential privacy. Boston, MA: Now Publ.

4 Erlingsson, U., Pihur, V., & Korolova, A. (2014). RAPPOR: Randomized Aggregatable Privacy-Preserving Ordinal Response. Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security - CCS 14. doi:10.1145/2660267.2660348

5 Wang, Y., Wu, X., & Hu, D. (2016). Using Randomized Response for Differential Privacy Preserving Data Collection. CEUR Workshop Proceedings. Retrieved from http://ceur-ws.org/Vol-1558/paper35.pdf

6 Zumel, A. N. (2015, October 03). A Simpler Explanation of Differential Privacy. Retrieved December 13, 2017, from http://www.win-vector.com/blog/2015/10/a-simpler-explanation-of-differential-privacy/