April 2020

A Method to Efficiently Generate Noise for Differential Privacy

Brett Krueger

A Method to Efficiently Generate Noise for Differential Privacy

Differential privacy uses noise to obfuscate raw statistical results so that individual contributors cannot be identified. This allows data to be shared publicly by describing patterns of groups without revealing information about any specific individual (i.e., anonymizes the data). To balance privacy and utility, the noise is generally generated as part of a Laplace or geometric distribution. However, currently available techniques of generating noise are usually inaccurate and/or leak information unless the noise level is further increased as compensation. Thus, traditional methods that are sufficiently accurate (e.g., Bernoulli trials) are typically very slow.[1] Previous works have described the inaccuracies commonly present (e.g., sampling from the Laplace distribution) and offer solutions to increase privacy, but at the cost of reduced utility.[2] Other works have presented some optimizations to known methods (e.g., the staircase method), but ignore floating point implementation issues.[3] Yet other works may be modified to include sufficient accuracy properties, but lack the necessary efficiency for frequent use.[4]

Typically, differential privacy is a property of a randomized algorithm that is executed over a database. This requires that the probability distributions from the output of the algorithm on adjacent valid databases are "similar". That is, it is necessary that the probability distributions are mutually absolutely continuous with respect to each other barring a very small probability (i.e., the delta parameter). This paper discusses a software library or program that generates random numbers from the geometric distribution and uses an Institute of Electrical and

[1] J. von Neumann, "Various techniques used in connection with random digits. Monte Carlo methods", Nat. Bureau Standards, 12 (1951), pp. 36–38.
[2] See https://dl.acm.org/citation.cfm?id=2382264
[3] See https://arxiv.org/pdf/1212.1186.pdf
[4] See https://sourceforge.net/projects/exrandom/ and https://arxiv.org/abs/1303.6257v2

Electronics Engineers (IEEE) 754 floating point format to generate samples such that the probability of each numerical value is off by a very small multiplicative factor from the distribution obtained by truncating an ideal exponential distribution to a predetermined number of bits. This random number may be added to a statistical result to add a random amount of noise to the result to help anonymize the data.

Finite precision Laplace/exponential random variables (both fixed and floating point) must have the same distribution as an appropriately scaled geometric random variable, thus generation of a high quality geometric random variable is necessary. To this end, consider a parameter λ:

$$\lambda = 1 - \ln(1 - p)$$

Here, λ is the parameter for an exponential distribution with an integer portion distributed as the geometric distribution of p. When ignoring floating point rounding errors, conveniently $\lambda = \varepsilon$, where ε is a positive number equal to the privacy loss associated with any data release drawn from a statistical database (i.e., the "privacy budget"). Based on this, the range of λ may be:

$$\lambda > \ln\frac{9}{10}$$

In this range, the probability of the random variable being equal to 0 is greater than 0.1. From here, p is calculated from:

$$p = 1 - e^{-\varepsilon}$$

Based on the calculated p, Bernoulli trials may be generated with Bernoulli(p) until a success occurs. On average, less than 10 trials are needed to get a success. The number of trials

needed is used as the magnitude of noise added to the statistical result. Lower values of λ may be achieved by first generating a value using the above techniques and performing repeated bisection to generate additional bits of the random value. To satisfy the privacy requirement, a λ that is less than or equal to ε must be chosen. Generally, a λ that is equal to ε is chosen to minimize noise. However, in some scenarios (e.g., when using a Fixed Point Laplace Mechanism), other values may be selected (e.g., an ε that is scaled by a power of two such as ε / 250).

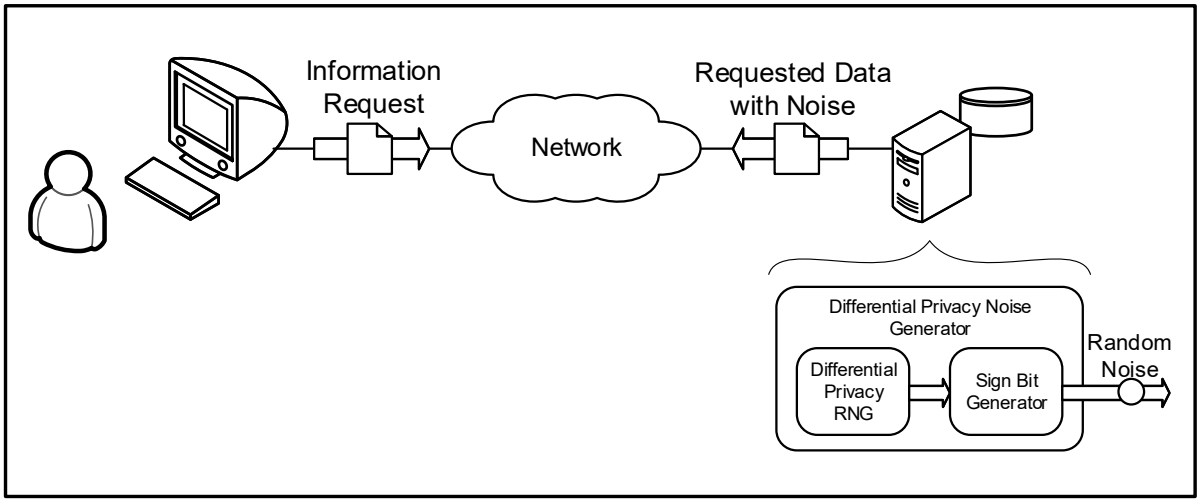

Figure 1

As shown in Figure 1, a differential privacy noise generator receives an information request for statistical information. The differential privacy noise generator includes a differential privacy random number generator that generates a magnitude of random noise. That is, the differential privacy random number generator generates a geometric random variable with a success probability of $1 - e^{-\varepsilon}$ (i.e., the number of Bernoulli trials needed to get a success). The differential privacy noise generator also includes a sign bit generator. The sign bit generator receives the noise magnitude from the differential privacy random number generator. The sign bit generator generates a sign bit with a 50% probability of positive or negative sign and combine

the generated sign with the noise magnitude to generate the random noise. The generated value may be discarded using rejection sampling to maintain privacy guarantees. The random noise is added to the requested data (i.e., increasing the number when the sign of the random noise is positive and decreasing the number when the sign of the random noise is negative).

The differential privacy noise generator may implement various optimizations to increase computational speed. For example, the differential privacy noise generator may implement reduced entropy usage for Bernoulli sampling or delay performing bit-by-bit bisection in order to make use of bitwise operations. Additionally, the hyperbolic tangent (i.e., "tanh") may be improved by using the Taylor series as well as the continued fraction expansions of tanh to perform fewer computations to achieve the same level of accuracy.

Compared to known existing implementations, this software library or application offers significant accuracy (i.e., less than 1 part in 10^{15} error with improvements to 1 part in 10^{18} achievable). The software library may be implemented in C++ and uses bit operations to reduce overall computation and increase efficiency.

In conclusion, the software library or application may be used to generate a random number and add the random number to a result of a statistical query. For example, a query may request a count of a number of people performing a specific activity. In addition to high accuracy, the differential privacy noise generator discussed herein is much more efficient than previous algorithms. For example, the differential privacy noise generator requires less than 50 nanoseconds per sample for a typical desktop computer for the majority of parameter values and less than 20 nanoseconds for parameter values commonly used in practice.

The methods described may be used to anonymize private data accurately and efficiently by generating differential noise using the geometric distribution.

ABSTRACT

Noise for differential privacy is often generated to obfuscate raw statistical results to hide identifies of specific individuals in data. Traditional methods are often inaccurate and/or computationally inefficient. In this work, a software library or program uses a differential privacy noise generator to generate high quality random numbers from the geometric distribution to both accurately and efficiently generate noise for differential privacy.