Smooth Sensitivity and Sampling in Private Data Analysis*

Kobbi Nissim† Sofya Raskhodnikova‡ Adam Smith§

Draft Full Version, v1.0
May 17, 2011

Abstract

We introduce a new, generic framework for private data analysis. The goal of private data analysis is to release aggregate information about a data set while protecting the privacy of the individuals whose information the data set contains. Our framework allows one to release functions f of the data with instance-specific additive noise. That is, the noise magnitude is determined not only by the function we want to release, but also by the database itself. One of the challenges is to ensure that the noise magnitude does not leak information about the database. To address that, we calibrate the noise magnitude to the *smooth sensitivity* of f on the database x — a measure of variability of f in the neighborhood of the instance x. The new framework greatly expands the applicability of output perturbation, a technique for protecting individuals' privacy by adding a small amount of random noise to the released statistics. To our knowledge, this is the first formal analysis of the effect of instance-specific noise in the context of data privacy.

Our framework raises many interesting algorithmic questions. Namely, to apply the framework one must compute or approximate the smooth sensitivity of f on x. We show how to do this efficiently for several different functions, including the median and the cost of the minimum spanning tree. We also give a generic procedure based on sampling that allows one to release $f(x)$ accurately on many databases x. This procedure is applicable even when no efficient algorithm for approximating smooth sensitivity of f is known or when f is given as a black box. We illustrate the procedure by applying it to k-SED (k-means) clustering and learning mixtures of Gaussians.

Categories and Subject Descriptors: F.2.2 Nonnumerical Algorithms and Problems

General Terms: Algorithms, Security, Theory.

Keywords: differential privacy, private data analysis, privacy preserving data mining, output perturbation, sensitivity, clustering.

*A preliminary version of this work appeared at STOC 2007. Part of this research was done while the authors were visiting the Institute for Pure and Applied Mathematics (IPAM) at UCLA and also while S.R. and A.S. were at the Weizmann Institute of Science. A.S. was supported at Weizmann by the Louis M. and Anita L. Perlman postdoctoral fellowship. The work of K.N. was supported in part by the Israel Science Foundation (grant No. 860/06).

†Dept. of Computer Science, Ben-Gurion University of the Negev. kobbi@cs.bgu.ac.il.

‡Dept. of Computer Science and Engineering, Pennsylvania State University, sofya@cse.psu.edu.

§Dept. of Computer Science and Engineering, Pennsylvania State University, asmith@cse.psu.edu.

Contents

1 Introduction

Data privacy is a fundamental problem of the modern information infrastructure. Collections of personal and sensitive data, previously the purview of governments and statistical agencies, have become ubiquitous as database systems have grown larger and cheaper. Increasing volumes of information are collected and archived by health networks, financial organizations, search engines, intrusion detection systems, social networking systems, retailers and other enterprizes. The potential social benefits from analyzing these databases are enormous. The main challenge is to release aggregate information about the databases while protecting the privacy of individual contributors.

There is a vast body of work on the data privacy problem, both in statistics and computer science (for references, see [1,9,26,27,18]). However, the schemes proposed in most of the literature lack analysis that is up to the level of rigor expected in, for example, cryptography. Typically, the schemes have either no formal privacy guarantees or ensure security only against a specific suite of attacks. For example, it is widely believed in traditional research on data privacy that, since data mining algorithms are designed to reveal only 'global' information, it is safe to apply them 'as they are' to sensitive data and publish the results. There is currently no formal substantiation of such an assessment.

This work is part of a newly emerging rigorous study of data privacy, inspired by research in cryptography, which acquired the name of *private data analysis*. This line of work [11,17,16,4,7,8,12] presents precise mathematical definitions of data privacy, that give meaningful guarantees in the presence of a strong, realistic adversary. The most successful of these definitions, *differential privacy* [15] (see Definition 1.1), has lead to an explosion of new results [12,13,6,3,21] . Protocols satisfying the definitions employ a technique called *output perturbation*, according to which the results of data analysis are released after the addition of a small amount of random noise[1]. Blum et al. [4] and Dwork et al. [15] present frameworks for releasing a specified function f of the data x while preserving privacy. The noise added to $f(x)$ in these frameworks depends only on f, and not on the database x. This provides a provably secure method for releasing many useful functions, such as means, variances, histograms, contingency tables and the singular value decomposition.

Instance-Specific Noise. We introduce a new generic framework for private data analysis. Our framework allows one to release functions f of the data with instance-specific additive noise. That is, the noise magnitude is determined not only by the function we want to release, but also by the database itself. One of the challenges is to ensure that the noise magnitude does not leak information about the database. To address that, we calibrate the noise magnitude to the *smooth sensitivity* of f on the database x — a measure of variability of f in the neighborhood of the instance x. The new framework greatly expands the applicability of output perturbation. To our knowledge, this is the first formal analysis of the effect on privacy of instance-specific noise.

Our framework raises many interesting algorithmic questions. To apply the framework one must compute or approximate the smooth sensitivity of f on x. These computations are non-trivial and, for some functions f, NP-hard. The approximation problems are further complicated by the requirement that the approximation must be *smooth*, that is, it should not change quickly in any neighborhood of its input space. This requirement prevents the noise magnitude, which is based on the approximation to smooth sensitivity, from leaking information.

Our framework generalizes the results of [4,15]. Those works calibrate noise to the *global sensitivity*, a simple but often crude upper bound on the smooth sensitivity. For many functions f and databases x, we

[1] Since the conference version of our work has been published, promising techniques other than output perturbation have been developed (see, for example, [21,5]). However, output perturbation is still the dominant method in the design of differentially private protocols.

can release $f(x)$ with less noise than in [4,15]. In particular, there are many functions for which previous frameworks are not applicable because they would add so much noise that the output would be meaningless. We make several observations useful for computing and approximating smooth sensitivity, and demonstrate their applicability for two types of data: real numbers contained in a bounded range, and graphs where individuals hold information about edges. For real numbers, we give efficient algorithms for computing and approximating the median and the minimum. For graphs, we give polynomial-time algorithms for computing the cost of the minimum spanning tree, and the number of triangles.

Sample and Aggregate. The framework of smooth sensitivity is very powerful. However, in order to use it directly for releasing a function f, one needs to design an efficient algorithm for computing or approximating the smooth sensitivity of f. For a given function, such an algorithm might not exist or might be difficult to design. More importantly, this step cannot be automated. In the interactive model, where users of the statistics can specify the function f to be evaluated on the database, it implies that the smooth sensitivity for all allowed user requests has to be analyzed and known in advance.

We present a *sample and aggregate* framework that circumvents these difficulties – a generic method that is efficient for a large class of functions, and can be applied without an explicit computation of smooth sensitivity. This method can be fully automated and works even when f is given as a black box. It allows users to specify their query function f simply by giving a computer program.

For example, many data mining algorithms perform some kind of clustering on their data. The global sensitivity framework [15] allows one to release the cost of clustering, but not cluster centers. In general, we do not know how to compute or approximate smooth sensitivity of cluster centers. We illustrate the sample and aggregate framework by applying it to two clustering problems: k-Squared Error Distortion clustering (also called k-means) and learning mixtures of Gaussian distributions. In both cases, we show that on interesting instances we can release the set of centers with very little noise.

In the following we review some of the prior research, and state our results.

1.1 Privacy Definition

As in [17,11,16,4,15,12], we consider a trusted agency setup, where users' queries are answered by a trusted agency that holds the database. Each query is a function f to be evaluated on the database. The database is modeled as a vector $x \in D^n$, where each entry x_i represents information contributed by one individual. Each entry could be an individual's salary, or the weight of an edge in a network, or any other arbitrary complex data. The server runs an algorithm $\mathcal{A}$ on the database x and sends back $\mathcal{A}(x)$. For example, $\mathcal{A}(x)$ could be $f(x)$ with added random noise. Our goal is to make $\mathcal{A}(x)$ as close to $f(x)$ as possible, thus enabling the user to learn his target value as accurately as possible, while preserving the privacy of individual contributors.

We use the privacy definition from [15,13], which limits the *incremental* information a user might learn in addition to what he knew before seeing the released statistics.

Notation. The *Hamming distance* $d(x, y)$ between two databases is the number of entries on which x and y differ, i.e., $d(x, y) = |\{i : x_i \neq y_i\}|$. Two databases are *neighbors* if they differ in a single individual's data, i.e., $d(x, y) = 1$.

For a particular query f and a database x, the randomized algorithm $\mathcal{A}$ defines a distribution $\mathcal{A}(x)$ on the outputs. An algorithm $\mathcal{A}$ is private if neighbor databases induce nearby distributions on the outputs. The intuition is that, all other things being equal, roughly the same information should be released no matter what a particular individual's data is.

Definition 1.1 (Differential Privacy[2] [15,13]). *A randomized algorithm $\mathcal{A}$ is (ϵ, δ)-differentially private if for all $x, y \in D^n$ satisfying $d(x, y) = 1$, and for all sets $\mathcal{S}$ of possible outputs*

$$\Pr[\mathcal{A}(x) \in \mathcal{S}] \leq e^{\epsilon} \Pr[\mathcal{A}(y) \in \mathcal{S}] + \delta \ .$$

When $\delta = 0$, the algorithm is ϵ-differentially private.

If the agency uses a differentially private algorithm then no individual has a pronounced effect on the statistics published by the agency, in the sense that the output distribution is almost the same whether the individual supplies his actual data or something irrelevant.

Differential privacy can be defined analogously for interactive protocols where each round consists of the server publishing one statistics in response to one query f by a user. This definition composes smoothly: a t-round protocol, in which each round is individually ϵ-differentially private, is itself $t\epsilon$-differentially private. Similarly, a t-round protocol, in which each round is (ϵ, δ)-differentially private, is $(t\epsilon, t\delta)$-differentially private. More generally:

Lemma 1.2 (Composition of Differentially Private Algorithms [14,20]). *Let $\mathcal{A}_1, \ldots, \mathcal{A}_t$ be a sequence of (ϵ, δ) differentially private algorithms, where for each i, $\mathcal{A}_i$ is selected based on the outputs of $\mathcal{A}_1(x), \ldots, A_{i-1}(x)$. Given any strategy for selecting the algorithms $\mathcal{A}_i$, the protocol which outputs $\mathcal{A}_1(x), \ldots, A_t(x)$ is $(t\epsilon, t\delta)$-differentially private.*

We focus on constructing differentially private algorithms that can be used on their own and in multi-round protocols. Relying on Lemma 1.2, we omit from our discussion the noise magnitude dependency on number of queries.

1.2 Calibrating Noise to Sensitivity

Recall that most works in private data analysis [11,16,4,15] use *output perturbation*, where privacy is achieved by adding random noise that 'masks' the private information. To release a function f of the database x, the server computes $f(x)$ and publishes $A(x) = f(x) + Z$ for some random variable Z. Here we assume that f takes values in $\mathbb{R}^d$, and use the L_1 norm on $\mathbb{R}^d$ (denoted $\| \cdot \|_1$, or simply $\| \cdot \|$) as a distance metric on outcomes of f. This is mainly for ease of presentation as the following analysis may be generalized to other metric spaces, such as the L_2 (Euclidean) metric and the earthmover distance on sets of points in $\mathbb{R}^d$.

Definition 1.3 (Global Sensitivity [15]). *For $f : D^n \to \mathbb{R}^d$, the global sensitivity of f (with respect to the ℓ_1 metric) is*

$$GS_f = \max_{x,y:d(x,y)=1} \|f(x) - f(y)\|_1.$$

Dwork et al. [15] showed how to construct differentially private estimates of a real function f by adding noise sampled from the Laplace distribution with magnitude proportional to the global sensitivity:

Definition 1.4 (Laplace Distribution). *The probability density function (p.d.f.) of the Laplace distribution $\mathrm{Lap}(\lambda)$ is $h(z) = \frac{1}{2\lambda} e^{-|z|/\lambda}$. It has zero mean, and standard deviation $\sqrt{2}\lambda$.*

Claim 1.5 ([15]). *For all $f : D^n \to \mathbb{R}^d$, the database access mechanism $\mathcal{A}_f(x) = f(x) + (Z_1, \ldots, Z_d)$, where the Z_i are drawn i.i.d. from $\mathrm{Lap}(GS_f/\epsilon)$, is ϵ-differentially private.*

[2] The term "differential privacy" was coined by Mike Schroeder and first appeared in Dwork [12]; The paper that defined this notion, [15], called it *ϵ-indistinguishability*. The variant with two parameters, ϵ and δ, first appeared in [13].

Claim 1.5 yields two generic approaches to constructing database access mechanisms for functions f. The first approach [15] is to show that GS_f is low, and hence $f(x)$ can be released with noise $Z \sim \text{Lap}(GS_f/\epsilon)$. The second approach [4] is to use Lemma 1.2 and express f in terms of functions $g_1, g_2, \ldots$ with low global sensitivity. One needs then to analyze how the addition of noise to $g_1, g_2, \ldots$ (needed for differential privacy) interferes with the computation of f.

These approaches are productive for many functions, such as sum queries [11,16]; Principle Component Analysis, the Perceptron algorithm, k-means, learning ID3 decision trees, statistical learning [4]; histograms, Singular Value Decomposition, distance to a property, and functions that can be estimated on all x using small random samples from x [15].

1.3 Smooth Sensitivity

In the global sensitivity framework of [15] described above, noise magnitude depends on GS_f and the privacy parameter ϵ, but not on the instance x. For many functions, such as the median f_{med}, this approach yields high noise, that does not reflect the function's typical insensitivity to individual inputs. We propose a local measure of sensitivity:

Definition 1.6 (Local Sensitivity)**.** *For $f : D^n \to \mathbb{R}^d$ and $x \in D^n$, the local sensitivity of f at x (with respect to the ℓ_1 metric) is*

$$LS_f(x) = \max_{y:d(x,y)=1} \|f(x) - f(y)\|_1.$$

Observe that the global sensitivity from Definition 1.3 is $GS_f = \max_x LS_f(x)$. The notion of local sensitivity is a discrete analogue of the Laplacian (or maximum magnitude of the partial derivative in different directions). It has appeared before in the (seemingly unrelated) context of concentration of measure [29]. The current work is the first to use it in the context of private data analysis.

Suppose we release the median $f_{med}(x)$ with noise magnitude proportional to $LS_{f_{med}}(x)$, instead of $GS_{f_{med}}$. The resulting algorithm would add significantly less noise for typical inputs. However, it is too naïve, and does not satisfy Definition 1.1 as the the noise magnitude itself reveals information about the database (see Section 2 for more details).

Our goal is hence to add instance-specific noise with smaller magnitude than the worst-case noise GS_f/ϵ, and yet satisfy Definition 1.1. The reader might be concerned that when the noise depends on the database, the client will not know the accuracy of the answer supplied by the database access mechanism. However, the noise magnitude itself is guaranteed to be an insensitive function of the database. Hence, if the client desires to query the database about the noise magnitude on the input x, he is guaranteed to get the answer to this query with very little noise added.

We define a class of *smooth* upper bounds S_f on LS_f such that adding noise proportional to S_f is safe. We define a special smooth function S_f^* that is optimal, in the sense that $S_f(x) \geq S_f^*(x)$ for every other smooth S_f, and show how to compute S_f^* as well as smooth approximation to it for the median, and the cost of a minimum spanning tree (MST).

There are many other functions for which global sensitivity framework yields unacceptably high noise levels while the smooth sensitivity framework performs well on many instances. For some of them (e.g., the minimum and the maximum), computing smooth sensitivity is trivial. (In fact, understanding the smooth sensitivity of the minimum is needed for our algorithm for computing the smooth sensitivity of the MST cost.) For others (e.g., the number of triangles in a graph), it requires more ingenuity. Finally, there are very natural classes of functions (e.g., the cluster centers for various clustering problems and the problem

of learning the mixtures of Gaussians), for which no efficient algorithms for approximating smooth sensitivity are known. The sample and aggregate framework circumvents this difficulty by providing an efficient database access mechanism that treats the query function as a black box.

1.4 The Sample and Aggregate Framework

Sample and aggregate works by replacing f with a related function $\bar{f}$ for which smooth sensitivity is low and efficiently computable. The function $\bar{f}$ can be thought of as a "smoothed" version of f. First, f (restricted to smaller inputs) is evaluated on a sublinear number of random samples from database x. Such evaluations are performed several times and the results are combined with a novel aggregation function that we call the *center of attention*. The output of this computation, denoted by $\bar{f}$, is released using the smooth sensitivity framework. The released value is close to the desired answer, $f(x)$, on databases for which $f(x)$ is approximated well by evaluating f on the random samples. The intuition is that for such x each entry x_i can be changed without affecting the value of the function significantly, since this entry is not likely to appear in the sample.

Dwork et al. [15, Lemma 1] proved that if f can be approximated well from random samples on *all inputs* then the global sensitivity of f is low, and consequently f can be released with a small amount of noise. This result looks similar to our claim that in the sample and aggregate framework, $f(x)$ will be released accurately on inputs for which $f(x)$ is approximated well from random samples. However, our result is qualitatively stronger in two respects. First, our result is instance-specific: it applies to input x even if for some other $x' \neq x$, evaluating f on a random sample from x' does not yield a good approximation to $f(x')$. Second, the result of [15] is not algorithmic: since the approximation guarantee must hold for all instances, it only gives a proof technique to bound the global sensitivity of f. In contrast, sample and aggregate yields efficient database access mechanisms for all query functions that can be evaluated efficiently on samples from the database.

Our mechanism releases accurate answers on several interesting classes of inputs. For example, we prove that k-SED (k-means) cluster centers are released accurately when the data is *well-separated*, according to the definition proposed by Ostrovsky et al. [25]. This definition implies that all near-optimal clusterings of x induce similar partitions of the points of x. [25] use this fact to show that well-separated data sets are amenable to heuristics based on Lloyd's algorithm. Our techniques also allow one to learn and publish accurate parameters of a mixture of k spherical Gaussian distributions when the data x consists of polynomially-many (in the dimension and k) i.i.d. samples from the distribution.

Previously, Blum et al. [4] showed that if there is an algorithm for approximating $f(x)$ using "noisy sum queries", then $f(x)$ can be released accurately while preserving privacy. Their framework can also be interpreted as identifying a "good" class of functions and inputs for which one can add relatively little noise. Their approach requires a fairly in-depth understanding of f, as one must be able to express f in terms of a limited class of queries to the data.

Using their framework, Blum et al. [4] gave a private version of a specific heuristic for k-SED clustering, called Lloyd's algorithm (or the k-means algorithm). They did not, however, prove guarantees on how close the final output of the algorithm is to the optimal cluster centers for x. To our knowledge, our algorithms are the first to provide such guarantees while preserving privacy.

2 Instance-Specific Additive Noise

Recall that in the interactive framework, the database is stored on the trusted server. When the user needs to obtain $f(x)$, he sends a query f to the server and gets $f(x)+N(x)Z$ as a reply, where Z is a random variable drawn from a noise distribution in $\mathbb{R}^d$ (fixed in advance and known to the user) with standard deviation 1 in each coordinate. The sample from the noise distribution is multiplied by the scaling factor $N(x)$, which we refer to as the *noise magnitude*. As explained in the Introduction, [15] gave ϵ-differentially private protocols where the noise magnitude $N(x)$ is proportional to global sensitivity of $f(\cdot)$ (and therefore independent of database x). In this section, we explain how to safely release $f(x)$ with potentially much smaller noise magnitude, tailored to database x.

2.1 Smooth Upper Bounds on LS_f and Smooth Sensitivity

For a query function f, our goal is to release $f(x)$ with less noise when the local sensitivity of f at x is lower. This would allow us to release functions with large global (worst case) sensitivity, but typically small local sensitivity with much greater accuracy than allowed in [15].

Example 1. Let $f_{med}(x) = \text{median}(x_1, \ldots, x_n)$ where x_i are real numbers from a bounded interval, say, $D = [0, \Lambda]$. For simplicity, assume n is odd and the database entries are sorted in the nondecreasing order: $x_1 \leq \cdots \leq x_n$. Let $m = \frac{n+1}{2}$ be the rank of the median element. Global sensitivity of the median, $GS_{f_{med}}$, is Λ, since for $x_1 = \cdots = x_m = 0$ and $x_{m+1} = \cdots = x_n = \Lambda$, $f_{med}(x_1, \ldots, x_n) = 0$ and $f_{med}(x_1, \ldots, x_{m-1}, \Lambda, x_{m+1}, \ldots, x_n) = \Lambda$. In this case, adding noise proportional to $GS_{f_{med}}$ completely destroys the information. However, on typical inputs, f_{med} is not very sensitive: $LS_{f_{med}}(x) = \max(x_m - x_{m-1}, x_{m+1} - x_m)$.

Ideally, we would like to release $f(x)$ with noise magnitude proportional to $LS_f(x)$. However, noise magnitude might reveal information about the database. For example, in the case of the median, if the noise magnitude is proportional to $LS_{f_{med}}(x)$, then the probability of receiving a non-zero answer when $x_1 = \cdots = x_{m+1} = 0, x_{m+2} = \cdots = x_n = \Lambda$ is zero (since the median is 0 and the local sensitivity is also 0) whereas the probability of receiving a non-zero answer on the neighboring database $x_1 = \cdots = x_m = 0, x_{m+1} = \cdots = x_n = \Lambda$ is significant (since the median is 0 but the local sensitivity is now Λ). Thus, the protocol is not (ϵ, δ)-differentially private when δ is small (regardless of ϵ). $\diamondsuit$

The lesson from this example is that the noise magnitude has to be an insensitive function. To decide on the noise magnitude we will use a *smooth* upper bound on the local sensitivity, namely, a function S that is an upper bound on LS_f at all points and such that $\ln(S(\cdot))$ has low sensitivity. We say S is β-smooth if $GS_{\ln(S(\cdot))} \leq \beta$.

Definition 2.1 (A Smooth Bound on LS)**.** *For $\beta > 0$, a function $S : D^n \to \mathbb{R}^+$ is a β-smooth upper bound on the local sensitivity of f if it satisfies the following requirements:*

$$\forall x \in D^n : \qquad S(x) \geq LS_f(x)\,; \tag{1}$$

$$\forall x, y \in D^n, d(x, y) = 1 : \qquad S(x) \leq e^{\beta} \cdot S(y)\,. \tag{2}$$

Note that the constant function $S(x) = GS_f$ meets the requirements of Definition 2.1 with $\beta = 0$. When $\beta > 0$ it is a very conservative upper bound on LS_f. A function that is the *smallest* to satisfy Definition 2.1 is the smooth sensitivity of f:

Definition 2.2 (Smooth sensitivity)**.** *For $\beta > 0$, the β-smooth sensitivity of f is*

$$S^*_{f,\beta}(x) = \max_{y \in D^n} \left(LS_f(y) \cdot e^{-\beta d(x,y)} \right).$$

Lemma 2.3. *$S^*_{f,\beta}$ is a β-smooth upper bound on LS_f. In addition, $S^*_{f,\beta}(x) \leq S(x)$ for all $x \in D^n$ for every β-smooth upper bound S on LS_f.*

Proof. To see that $S^*_{f,\beta}$ is an upper bound on LS_f, observe that

$$\begin{aligned} S^*_{f,\beta}(x) &= \max \left(LS_f(x), \max_{y \neq x;\ y \in D^n} \left(LS_f(y) \cdot e^{-\beta d(x,y)} \right) \right) \\ &\geq LS_f(x). \end{aligned}$$

Next we show that $S^*_{f,\beta}$ is β-smooth, i.e, that $S^*_{f,\beta}(y) \geq e^{-\beta} S^*_{f,\beta}(x)$ for all neighboring databases x and y. Fix $x, y \in D^n$ with $d(x,y) = 1$. Let $x' \in D^n$ be such that $S^*_{f,\beta}(x) = LS_f(x') \cdot e^{-\beta d(x,x')}$. By the triangle inequality, $d(y,x') \leq d(y,x) + d(x,x') = d(x,x') + 1$. Therefore,

$$\begin{aligned} S^*_{f,\beta}(y) &\geq LS_f(x') \cdot e^{-\beta d(y,x')} \\ &\geq LS_f(x') \cdot e^{-\beta(d(x,x')-1)} \\ &= e^{-\beta} \cdot LS_f(x') \cdot e^{-\beta d(x,x')} = e^{-\beta} \cdot S^*_{f,\beta}(x). \end{aligned}$$

Now let S be a function satisfying Definition 2.1. We will show that $S(x) \geq S^*_{f,\beta}(x)$ for all $x \in D^n$. To prove this, it is enough to establish that $S(x) \geq LS_f(y) \cdot e^{-\beta d(x,y)}$ for all $x, y \in D^n$. We demonstrate it by induction on $d(x,y)$.

The base case, $S(x) \geq LS_f(x)$, is the requirement (1) of Definition 2.1. For the induction step, suppose $S(x') \geq LS_f(y) \cdot e^{-\beta d(x',y)}$ for all x', y at distance k. Consider x, y at distance $k+1$. There exists x': $d(x,x') = 1, d(x',y) = k$. By requirement (2) of Definition 2.1, $S(x) \geq S(x') \cdot e^{-\beta}$. Using the induction hypothesis, $S(x') \geq LS_f(y) \cdot e^{-\beta d(x',y)}$, we get $S(x) \geq LS_f(y) \cdot e^{-\beta(d(x',y)+1)} = LS_f(y) \cdot e^{-\beta d(x,y)}$, as required. □

2.2 Calibrating Noise According to Smooth Upper Bounds on LS_f

This section explains how to select a noise distribution so that adding noise proportional to a smooth upper bound on the local sensitivity results in a differentially private algorithm. In our smooth sensitivity framework, the noise magnitude is proportional to $\frac{S_f(x)}{\alpha}$, where S_f is a β-smooth upper bound on the local sensitivity of f, and α, β are parameters of the noise distribution.

For functions that return a single real value, we obtain the following concrete bounds, which follow from Lemmas 2.7 and 2.9.

Corollary 2.4 (Calibrating Noise to Smooth Bounds on the Sensitivity, 1-Dimensional Case)**.** *Let $f : D^n \to \mathbb{R}$ be any real-valued function and let $S : D^n \to \mathbb{R}$ be a β-smooth upper bound on the local sensitivity of f. Then*

1. *If $\beta \leq \frac{\epsilon}{2(\gamma+1)}$ and $\gamma > 1$, the algorithm $x \mapsto f(x) + \frac{2(\gamma+1)S(x)}{\epsilon} \cdot \eta$, where η is sampled from the distribution with density $h(z) \propto \frac{1}{1+|z|^\gamma}$, is ϵ-differentially private.*

2. *If* $\beta \leq \frac{\epsilon}{2\ln(\frac{2}{\delta})}$ *and* $\delta \in (0,1)$*, the algorithm* $x \mapsto f(x) + \frac{2S(x)}{\epsilon} \cdot \eta$*, where* $\eta \sim \mathrm{Lap}(1)$*, is* (ϵ, δ)*-differentially private.*

For functions taking values in $\mathbb{R}^d$, the situation is more complicated since the smoothing parameter β will depend on d as well as ϵ and δ. Moreover, there are many natural choices of metrics with respect to which one may measure sensitivity. We discuss the ℓ_1 (Lemma 2.9) and ℓ_2 metrics below (Lemma 2.10).

2.2.1 Admissible Noise Distributions

We start by abstracting out a requirement on admissible noise distributions in Definition 2.5. In Lemma 2.6, we prove that adding admissible noise and releasing the result is differentially private. Then we give several examples of admissible noise distributions, including Laplace and Gaussian, and work out their parameters.

Notation. For a subset $\mathcal{S}$ of $\mathbb{R}^d$, we write $\mathcal{S} + \Delta$ for the set $\{z + \Delta \mid z \in \mathcal{S}\}$, and $e^\lambda \cdot \mathcal{S}$ for the set $\{e^\lambda \cdot z \mid z \in \mathcal{S}\}$. We also write $a \pm b$ for the interval $[a - b, a + b]$.

Definition 2.5 (Admissible Noise Distribution)**.** *A probability distribution on* $\mathbb{R}^d$*, given by a density function* h*, is* (α, β)-admissible *(with respect to* ℓ_1*) if, for* $\alpha = \alpha(\epsilon, \delta), \beta = \beta(\epsilon, \delta)$*, the following two conditions hold for all* $\Delta \in \mathbb{R}^d$ *and* $\lambda \in \mathbb{R}$ *satisfying* $\|\Delta\|_1 \leq \alpha$ *and* $|\lambda| \leq \beta$*, and for all measurable subsets* $\mathcal{S} \subseteq \mathbb{R}^d$*:*

Sliding Property:
$$\Pr_{Z\sim h}\left[Z \in \mathcal{S}\right] \leq e^{\frac{\epsilon}{2}} \cdot \Pr_{Z\sim h}\left[Z \in \mathcal{S} + \Delta\right] + \tfrac{\delta}{2}\,.$$

Dilation Property:
$$\Pr_{Z\sim h}\left[Z \in \mathcal{S}\right] \leq e^{\frac{\epsilon}{2}} \cdot \Pr_{Z\sim h}\left[Z \in e^\lambda \cdot \mathcal{S}\right] + \tfrac{\delta}{2}\,.$$

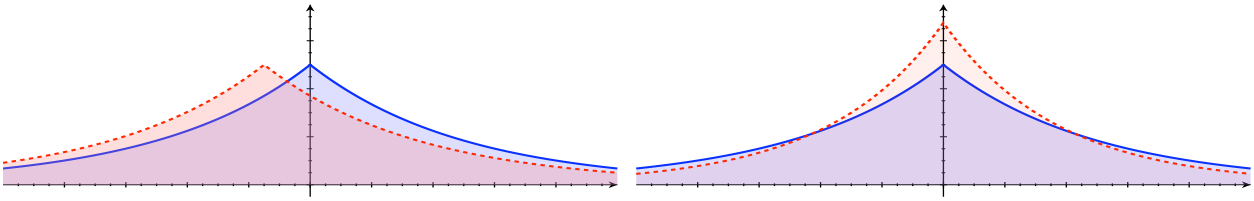

Figure 1: Sliding and dilation for the Laplace distribution with p.d.f. $h(z) = \frac{1}{2}e^{-|z|}$, plotted as a solid line. The dotted lines plot the densities $h(z + 0.3)$ (left) and $e^{0.3}h(e^{0.3}z)$ (right).

The definition requires the noise distribution to not change much under translation (sliding) and scaling (dilation). See Fig. 1 for an example. A distribution satisfying the two properties can be used to add noise proportional to a smooth upper bound on local sensitivity:

Lemma 2.6. *Let* h *be an* (α, β)*-admissible noise probability density function, and let* Z *be a fresh random variable sampled according to* h*. For a function* $f : D^n \to \mathbb{R}^d$*, let* $S : D^n \to \mathbb{R}$ *be a* β*-smooth upper bound on the local sensitivity of* f*. Then algorithm* $\mathcal{A}(x) = f(x) + \frac{S(x)}{\alpha} \cdot Z$ *is* (ϵ, δ)*-differentially private.*

For two neighbor databases x and y, the output distribution $\mathcal{A}(y)$ is a shifted and scaled version of $\mathcal{A}(x)$. The sliding and dilation properties ensure that $\Pr[\mathcal{A}(x) \in \mathcal{S}]$ and $\Pr[\mathcal{A}(y) \in \mathcal{S}]$ are close for all sets $\mathcal{S}$ of outputs.

Proof of Lemma 2.6: For all neighboring $x, y \in D^n$ and all sets $\mathcal{S}$, we need to show that

$$\Pr[\mathcal{A}(x) \in \mathcal{S}] \leq e^\epsilon \cdot \Pr[\mathcal{A}(y) \in \mathcal{S}] + \delta.$$

Denote $\frac{S(x)}{\alpha}$ by $N(x)$. Observe that $\mathcal{A}(x) \in \mathcal{S}$ if and only if $Z \in \mathcal{S}_1$, where $\mathcal{S}_1 = \frac{\mathcal{S}-f(x)}{N(x)}$. Let $\mathcal{S}_2 = \mathcal{S}_1 + \frac{f(y)-f(x)}{N(x)} = \frac{\mathcal{S}-f(y)}{N(x)}$ and $\mathcal{S}_3 = \mathcal{S}_2 \cdot \frac{N(x)}{N(y)} = \frac{\mathcal{S}-f(y)}{N(y)}$. Then

$$\begin{aligned}
\Pr[\mathcal{A}(x) \in \mathcal{S}] &= \Pr_{z\sim h}[z \in \mathcal{S}_1] \\
&\leq \Pr_{z\sim h}[z \in \mathcal{S}_2] \cdot e^{\epsilon/2} + \frac{\delta}{2} \\
&\leq \Pr_{z\sim h}[z \in \mathcal{S}_3] \cdot e^{\epsilon} + \frac{\delta}{2} \cdot e^{\epsilon/2} + \frac{\delta}{2} \\
&= \Pr_{z\sim h}[\mathcal{A}(y) \in \mathcal{S}] \cdot e^{\epsilon} + \delta.
\end{aligned}$$

The first inequality holds since h satisfies the Sliding Property of Definition 2.5 and since

$$\frac{\|f(y) - f(x)\|}{N(x)} = \alpha \cdot \frac{\|f(y) - f(x)\|}{S(x)} \leq \alpha \cdot \frac{\|f(y) - f(x)\|}{LS_f(x)} \leq \alpha.$$

The second inequality holds since h satisfies the Dilation Property of Definition 2.5 and since $S(x)$ is β-smooth, which implies that $\left|\ln \frac{N(x)}{N(y)}\right| = \left|\ln \frac{S(x)}{S(y)}\right| \leq |\ln e^{\pm\beta}| \leq \beta$. □

2.2.2 Examples of Admissible Noise Distributions

We discuss three families of admissible distributions on the real line; in higher dimensions, we use products of these distributions (that is, we add independent noise in each coordinate). The first family is a generalization of the Cauchy distribution, which has density proportional to $\frac{1}{1+z^2}$. It yields "pure" ϵ-differential privacy (that is, with $\delta = 0$). We subsequently consider the Laplace and Gaussian distributions, which lead to mechanisms with $\delta > 0$ (but tighter concentration and sometimes smaller noise).

Throughout the following proofs, we will use the following idea: to show that a dsitribution with density h satisfies a property like dilation, it suffices to show that the logarithm of the ratio $\ln(\frac{h(z+\Delta)}{h(z)})$ is bounded with high probability over z drawn according to the distribution given by h. Simialrly, for dilation we analyze $\ln(\frac{e^{\lambda}h(e^{\lambda}z)}{h(z)})$.

Lemma 2.7. *For any $\gamma > 1$, the distribution with density $h(z) \propto \frac{1}{1+|z|^{\gamma}}$ is $(\frac{\epsilon}{2(\gamma+1)}, \frac{\epsilon}{2(\gamma+1)})$-admissible (with $\delta = 0$). Moreover, the d-dimensional product of independent copies of h is $(\frac{\epsilon}{2(\gamma+1)}, \frac{\epsilon}{2d(\gamma+1)})$- admissible.*

Proof. We first consider a dilation by a factor of e^{λ}, where $|\lambda| < \frac{\epsilon}{2\gamma}$. To show the dilation property, it is sufficient to show that the logarithm of the ratio of the densities, $\ln(\frac{e^{\lambda}h(e^{\lambda}z)}{h(z)}) = \ln(\frac{e^{\lambda}(1+(\lambda|z|)^{\gamma})}{1+|z|^{\gamma}})$, is at most $\epsilon' \stackrel{\text{def}}{=} \epsilon/2$. For $\lambda \geq 0$, we can bound $\frac{(1+(e^{\lambda}|z|)^{\gamma})}{1+|z|^{\gamma}}$ above by $\frac{(e^{\lambda}|z|)^{\gamma}}{|z|^{\gamma}} = e^{\lambda\gamma}$, so we get $\ln(\frac{e^{\lambda}h(e^{\lambda}z)}{h(z)}) < \lambda(\gamma+1)$. This is at most ϵ' since $\lambda < \frac{\epsilon}{2(\gamma+1)}$. A symmetric argument works for $\lambda < 0$.

To prove the sliding property, write the logarithm of the ratio of the densities as $\ln(\frac{h(z+\Delta)}{h(z)})$ as a difference $\phi(|z|) - \phi(|z+\Delta|)$, where $\phi(z) = \ln(1+z^{\gamma})$. By the mean value theorem, there exists a point $\zeta > 0$ such that $|\phi(|z+\Delta|) - \phi(|z|)| \leq \Delta|\phi'(\zeta)|$. The magnitude of the derivative ϕ' is bounded by γ: for any ζ, $\phi'(\zeta) = \frac{\gamma\zeta^{\gamma-1}}{1+\zeta^{\gamma}} = \frac{\gamma}{\zeta+\zeta^{-(\gamma-1)}}$, and one of the terms ζ and $\zeta^{-(\gamma-1)}$ is at least 1 (recall $\gamma - 1 > 0$). Combining these bounds, we get $|\ln(\frac{h(z+\Delta)}{h(z)})| \leq \Delta\gamma$. Since $\Delta < \epsilon/2(\gamma+1)$, the logarithm of the densities' ratio is at most $\epsilon' = \epsilon/2$, as desired.

If we consider the d-wise product of h, then dilation by e^λ can increase each of the components of the product density by $e^{\lambda(\gamma-1)}$, for a total increase of at most $e^{d\lambda(\gamma-1)}$. It suffices to take $\lambda \leq \frac{\epsilon}{2d(\gamma+1)}$. In the case of translation by a vector $\Delta = (\Delta_1, ...\Delta_d) \in \mathbb{R}^d$, the i-th component gets increased by a factor of $e^{\Delta_i\gamma}$, so the overall increase is at most $e^{\|\Delta\|_1\gamma}$. It therefore suffices to take $\|\Delta\|_1 \leq \frac{\epsilon}{2(\gamma+1)}$. □

A simple observation gives an intuition to why $\delta = 0$ implies an inverse polynomial decrease. Consider a distribution $h(z)$ that behaves asymptotically as $e^{-f(z)}$ for some f. By the dilation property, $e^{-f(z)}/e^{-f(e^\beta z)} = e^{-f(z)+f(e^\beta z)} < e^\epsilon$ for some fixed ϵ or, equivalently, $f(e^\beta z) - f(z) < \epsilon$, for all $z \in \mathbb{R}$. If ϵ/β is bounded above, the constraint implies that $f(z) > c\ln(|z|)$ for some fixed c. Hence $h(z) = \Omega(1/z^c)$. To allow noise distributions with exponentially decreasing tails (such as Gaussian and Laplace), we must therefore take $\delta > 0$.

To prove sliding, it suffices to prove that the log-ratio $\ln(\frac{h(z+\Delta)}{h(z)}$ (respectively, $\ln(\frac{e^{d\lambda}h(e^\lambda z)}{h(z)})$) is bounded above by $\frac{\epsilon}{2}$ with probability at least $1 - \frac{\delta}{2}$. For both Gaussian and Laplace variables, the log-ratio is tied to the probability that the noise variable has large norm (i.e. lies far out on the tail). In high dimension, this probability is somewhat messy to state, so we first introduce some notation.

Definition 2.8. *Given a real-valued random variable Y and a number $\delta \in (0,1)$, let $\rho_\delta(Y)$ be the $(1-\delta)$-quantile of Y, that is, the least solution to $\Pr(Y \leq \rho_\delta) \geq 1 - \delta$.*

Lemma 2.9. *For $\epsilon, \delta \in (0,1)$, the d-dimensional Laplace distribution, $h(z) = \frac{1}{2^d} \cdot e^{-\|z\|_1}$, is (α,β)-admissible with $\alpha = \frac{\epsilon}{2}$, and $\beta = \frac{\epsilon}{2\rho_{\delta/2}(\|Z\|_1)}$, where $Z \sim h$. In particular, it suffices to use $\alpha = \frac{\epsilon}{2}$ and $\beta = \frac{\epsilon}{4(d+\ln(2/\delta))}$. For $d = 1$, it suffices to use $\beta = \frac{\epsilon}{2\ln(2/\delta)}$.*

Proof. The sliding property for Laplace was proven in [15]. For the dilation property, consider $\lambda > 0$. In this case, $h(e^\lambda z) < h(z)$, so the log-ratio $\ln(\frac{e^{d\lambda}h(e^\lambda z)}{h(z)})$ is at most $d\lambda \leq \frac{\epsilon}{2\rho_{\delta'}(\|Z\|_1)}$ (where $\delta' \stackrel{\text{def}}{=} \frac{\delta}{2}$). The median of the distribution $\|Z\|_1$ is at most d and $\delta' < 1/2$, so $\rho_{\delta'}(\|Z\|_1) > d$. Thus λd is at most $\frac{\epsilon}{2}$.

Next, we prove the dilation property for $\lambda < 0$. The ratio $\frac{h(e^\lambda z)}{h(z)}$ is $\exp(|z|(1-e^\lambda))$. Since $1 - e^\lambda \leq |\lambda|$, we get that $\ln(\frac{e^\lambda h(e^\lambda z)}{h(z)})$ is at most $|z|\lambda$. Consider the event $G = \{z : |z| \leq \rho_\delta\}$. Under this event, the log-ratio above is at most $\epsilon/2$. The probability of G under density h is $1 - \delta$. Thus, probability of a given set $\mathcal{S}$ under h is at most $\Pr_h[\mathcal{S} \cap G] + \Pr_h[\bar{G}] \leq e^{\epsilon/2}\Pr_{h'}[\mathcal{S} \cap G] + \frac{\delta}{2} \leq e^{\epsilon/2}\Pr_{h'}[\mathcal{S}] + \frac{\delta}{2}$, where $h'(z) = e^\lambda h(e^\lambda z)$ is the density of the dilated distribution.

The norm $\|Z\|_1$ is a sum of d indpendent exponential random variables. By Fact A.1, the quantile $\rho_{\delta'}$ is at most $2d + 2\ln(1/\delta')$, so it suffices to take $\beta = \frac{\epsilon}{4(d+\ln(2/\delta))} < \frac{\epsilon}{2\rho_{\delta'}(\|Z\|_1)}$. □

Here, we work out the details for the case of Gaussian noise. This type of noise is useful since it allows us to tailor the noise level to a function's sensitivity in the ℓ_2 (Euclidean) norm. This will be useful when we consider k-means clustering in high dimensions, for example, where the Euclidean metric is natural.

Lemma 2.10 (Gaussian Distribution)**.** *For $\epsilon, \delta \in (0,1)$, the d-dimensional Gaussian distribution, $h(z) = \frac{1}{(2\pi)^{d/2}} \cdot e^{-\frac{1}{2}\|z\|_2^2}$, is (α,β)-admissible for the Euclidean metric with $\alpha = \frac{\epsilon}{5\rho_{\delta/2}(Z_1)}$, and $\beta = \frac{\epsilon}{2\rho_{\delta/2}(\|Z\|_2^2)}$, where $Z = (Z_1, ..., Z_d) \sim h$.*

In particular, it suffices to take $\alpha = \frac{\epsilon}{5\sqrt{2\ln(2/\delta)}}$ and $\beta = \frac{\epsilon}{4(d+\ln(2/\delta))}$.

Proof. The sliding property was proven implicitly by Blum, Dwork, McSherry and Nissim [4]. Consider a shift $\Delta \in \mathbb{R}^d$, with $\|\Delta\|_2 \leq \alpha$. Since Gaussian noise is spherically symmetric, we can consider $\Delta =$

$(\alpha, 0, ..., 0)$ without loss of generality. Then the ratio $\frac{h(z+\Delta)}{h(z)}$ reduces to the ratio of a one-dimensional Gaussian density evaluated at points z_1 and $z_1 + \alpha$. This ratio is $\exp(|z_1|^2 - |z_1 + \alpha|^2)$, which is at most $\exp(2\alpha|z_1| + \alpha^2)$.

It therefore suffices to prove that $2\alpha|z_1| + \alpha^2 \leq \epsilon/2$ with probability at least δ. Let G be the event $|z_1| < \rho_{\delta'}(z_1)$. Conditioned on G, $2\alpha|z_1| + \alpha^2 \leq \frac{2\epsilon}{5} + \frac{\epsilon^2}{25} \leq \epsilon/2$, as desired. (The last inequality uses the assumption that $\epsilon < 1$, and $\rho_{\delta'}(|Z_1|) > 1$.) The probability of G is $\delta' = \delta/2$, which bounds the additive term in the sliding property. Because $|z_1|$ is is normal with mean 0 and variance 1, we have $\rho_{\delta'}\sqrt{2\ln(2/\delta)}$, by a standard tail bound (Fact A.2). So it suffices to take $\alpha = \frac{\epsilon}{5\sqrt{2\ln(2/\delta)}}$.

For dilation by a factor of $\lambda > 0$, the density $h(e^\lambda z)$ is less than $h(z)$ for all z, so $\ln(\frac{e^{d\lambda}h(e^\lambda z)}{h(z)}) \leq d\lambda < \epsilon/2$ (since the median of $\|Z\|_2^2$ is at least d).

For $\lambda < 0$, note that $\ln(\frac{h(e^\lambda z)}{h(z)}) = \exp(\frac{1}{2}\|z\|_2^2(1 - e^{2\lambda})) \leq \|z\|_2^2 \cdot |\lambda|$. By the definition, the norm $\|z\|_2^2$ exceeds $\rho_{\delta'}(\|Z\|_2^2) = \frac{\epsilon}{2|\lambda|}$ with probability at most $\delta' = \delta/2$. Under this condition, $\|z\|_2^2 \cdot |\lambda| \leq \frac{\epsilon}{2}$, which implies the dilation condition.

Finally, we can bound $\rho_{\delta'}(\|Z\|_2^2)$ by $2d + 4\ln(1/\delta')$, to show that $\beta = \frac{\epsilon}{4(d+\ln(2/\delta))}$ is sufficient. □

3 Computing Smooth Sensitivity

In this section we show how to compute smooth sensitivity $S^*_{f,\epsilon}(x)$, as in Definition 2.2, for several specific functions: median, minimum, the cost of a minimum spanning tree and the number of triangles in a graph. We also construct a function for which the smooth sensitivity is hard to compute or even approximate.

First we give some generic observations on computing smooth sensitivity. We start by defining a function that describes how much the sensitivity can change when up to k entries of x are modified. This function has to be well understood in order to compute the smooth sensitivity of f.

Definition 3.1. *The sensitivity of f at distance k is*

$$A^{(k)}(x) = \max_{y \in D^n:\ d(x,y) \leq k} LS_f(y) \ .$$

Now smooth sensitivity can be expressed in terms of $A^{(k)}$:

$$\begin{aligned} S^*_{f,\epsilon}(x) &= \max_{k=0,1,...,n} e^{-k\epsilon}\left(\max_{y:\ d(x,y)=k} LS_f(y)\right) \\ &= \max_{k=0,1,...,n} e^{-k\epsilon} A^{(k)}(x)\,. \end{aligned}$$

Thus, to compute the smooth sensitivity of f at x, it suffices to understand $A^{(k)}(x)$.

For functions for which we cannot compute $S^\star$ efficiently, we might be able to give an efficient approximation algorithm. We stress that not every approximation to S^* is appropriate in our framework: some approximations to S^* might leak information. The function computed by an approximation algorithm is acceptable only if it is a smooth upper bound on $LS_f(x)$. The next claims provide methods for giving smooth upper bounds on local sensitivity.

Claim 3.2. *For a given value $k_0(n)$, let*

$$\hat{S}_{f,\epsilon}(x) = \max(GS_f \cdot e^{-\epsilon k_0}, \max_{k=0,...,k_0-1} e^{-\epsilon k} \cdot A^{(k)}(x)).$$

Then $\tilde{S}_{f,\epsilon}(x)$ is an ϵ-smooth upper bound on local sensitivity.

Proof. Consider two neighboring inputs x and y. □

Claim 3.3. *Let $\tilde{S}_{f,\epsilon}(x) = \max_{k=0,\ldots,n}(U_k(x) \cdot e^{-\epsilon k})$ where U_k satisfies*

1. *$LS_f(x) \leq U_0(x)$, and*
2. *$U_k(x) \leq U_{k+1}(y)$ for all x, y such that $d(x, y) = 1$.*

Then $\tilde{S}_{f,\epsilon}(x)$ is an ϵ-smooth upper bound on local sensitivity.

Proof. We need to show that equations (1) and (2) of Definition 2.1 hold for $\tilde{S}_{f,\epsilon}(x)$. Equation (1) holds as

$$LS_f(x) \leq U_0(x) \leq \tilde{S}_{f,\epsilon}(x).$$

Equation (2) holds as

$$\tilde{S}_{f,\epsilon}(x) = \max_{k=0,\ldots,n} e^{-\epsilon k} \cdot U_k(x) \leq e^{\epsilon} \cdot \max_{k=1,\ldots,n} e^{-\epsilon k} \cdot U_k(y) \leq e^{\epsilon} \cdot \tilde{S}_{f,\epsilon}(y).$$

□

3.1 Smooth Sensitivity of the Median

Recall that $f_{med}(x_1, ..., x_n)$ was defined to be the median of values in $D = [0, \Lambda]$. For simplicity, assume n is odd, and the database elements are in nondecreasing order: $0 \leq x_1 \leq \cdots \leq x_n \leq \Lambda$. In Example 1 we observed that $GS_{f_{med}} = \Lambda$, and $LS_{f_{med}} = \max(x_m - x_{m-1}, x_{m+1} - x_m)$ for $m = \frac{n+1}{2}$. For notational convenience, define $x_i = 0$ for $i \leq 0$ and $x_i = \Lambda$ for $i > n$.

Proposition 3.4. *The smooth sensitivity of the median is*

$$S^*_{f_{med},\epsilon}(x) = \max_{k=0,\ldots,n}(e^{-k\epsilon} \cdot \max_{t=0,\ldots,k+1}(x_{m+t} - x_{m+t-k-1})).$$

It can be computed in time $O(n \log n)$.

Before we prove the proposition, we illustrate the result with an example. Consider an instance where the points x_i are restricted to the interval $[0, 1]$ (that is, $\Lambda = 1$) and the points are evenly spaced the interval (that is, $x_i = \frac{i}{n}$ for $i = 1, ..., n$). In this case, $S^*(x) = \max_k e^{-\epsilon k} \cdot \frac{k+1}{n}$. The maximum occurs at $k = 1/\epsilon$. We get $S^* \leq \frac{1}{\epsilon n}$ and so the magnitude of the noise we add is $\frac{1}{\epsilon^2 n}$. For comparison, the noise magnitude for f_{med} in the global sensitivity framework of [15] is $1/\epsilon$; adding noise of that magnitude essentially wipes out all information about the median since the extreme values, 0 and 1 are hard to distinguish.

Proof. By changing up to k entries in $x_1, \ldots, x_n$ to 0 or Λ, one can shift the median anywhere in the interval $[x_{m-k}, x_{m+k}]$. The local sensitivity at distance k is maximized when the new median is an end point of a large empty interval. This is achieved when entries $x_{m-k+t}, \ldots, x_{m-1+t}$ for some $t = 0, \ldots, k+1$ are modified as follows: x_i with $i < m$ are set to 0 and x_i with $i \geq m$ are set to Λ. Thus,

$$A^{(k)}(x) = \max_{y:\ d(x,y) \leq k} LS(y) = \max_{0 \leq t \leq k+1}(x_{m+t} - x_{m+t-k-1}) \ .$$

As $A^{(k)}$ is computable in time $O(k)$, we get that $S^*_{f_{med}}(x) = \max_k e^{-\epsilon k} A^{(k)}(x)$ is computable in time $O(n^2)$. However, one in fact give an $O(n \log n)$-time algorithm for computing $S^*_{f_{med}}(x)$. The algorithm we give here is due to Sergey Orshanskiy.

The idea behind the algorithm is to collapse and reorder the terms in the "max" expression that defines $S^*_{f_{med}}(x)$. First, note that $S^*_{f_{med}}(x) = \max_{i<j}(x_j - x_i)e^{\epsilon(j-i+1)}$. For every index $i < \frac{n}{2}$, let $j^*(i)$ be the index of the right endpoint of the "best" interval whose left endpoint is x_i, that is, let $j^*(i) = \text{argmax}_{j \geq n/2}(x_j - x_i)e^{\epsilon(j-i+1)}$. In order to compute $S^*_{f_{med}}(x)$, it suffices to compute $j^*(i)$ for all values of i, since given the j^* values one can compute $S^*_{f_{med}}(x)$ in time $O(n)$. So we focus on computing the list $j^*(1), ..., j^*(n)$ in time $O(n \log n)$.

Suppose we know $j^*(a)$ and $j^*(c)$ where $a < c$ are indices in $\{1, ..., n\}$. For any index b between a and c, we can compute $j^*(b)$ in time $O(j^*(c) - j^*(a))$ since the monotonicity of j^* allows us to restrict the search to $\{j^*(a), ..., j^*(c)\}$. A natural strategy to compute all the j^* values in $\{a, ..., c\}$ is thus to first compute $j^*(\frac{a+c}{2})$ and use it to speed up computation of the remaining values. We obtain the following recursive algorithm:

Algorithm 1: `J-List`(a, c, L, U)

```
// Returns the list j*(a), ..., j*(c) assuming that L ≤ j*(a) and j*(c) ≤ U
if c < a then
    return (empty list);
else
    b ← ⌊(a + c)/2⌋;
    j*(b) ← argmax_{L≤j≤U}(x_j − x_b)e^{ε(j−b+1)}  // This takes time O(U − L).
    return Concatenate(J-List(a, b − 1, L, j*(b)), j*(b), J-List(b + 1, c, j*(b), U));
```

The initial call of the algorithm is `J-List`$(1, n, 1, n)$. On inputs a, c, L, U, the algorithm above runs in time $O(n + w \log n)$ where $n = c - a$ and $w = U - L$. This can be proved by straightforward induction on n and w. □

3.1.1 Linear and Sublinear Approximations to the Median's Smooth Sensitivity

We present two approximation algorithms, based on Claims 3.2 and 3.3, that give smooth upper bounds on $S^*_{f_{med},\epsilon}$ and run faster than the (known) algorithm for exact computation. The first one computes a factor of 2 approximation in time $O(n)$. The second has additive error $\frac{\Lambda}{\text{poly}(n)}$ and runs in (sublinear) time $O((\log^2 n)/(\epsilon^2))$. In both cases, we assume that the data is initially sorted. The second algorithm is able to run in sublinear time because it inspects only the values x_i that have rank close to $n/2$.

Define $\hat{S}_{f_{med},\epsilon}$ as in Claim 3.2 and $\tilde{S}_{f_{med},\epsilon}$ as in Claim 3.3, where $U_k(x) = x_{m+k+1} - x_{m-k-1}$ and $k_0 = O(\log n/\epsilon)$.

Claim 3.5. *Both $\tilde{S}_{f_{med},\epsilon}$ and $\hat{S}_{f_{med},\epsilon}$ are ϵ-smooth upper bounds on $LS_{f_{min}}$. Furthermore:*

1. *$\tilde{S}_{f_{med},\epsilon}(x) \leq 2S^*_{f_{min},\epsilon}(x)$; moreover, $\tilde{S}_{f_{med},\epsilon}(x)$ can be computed in time $O(n)$; and*
2. *$\hat{S}_{f_{med},\epsilon}(x) \leq S^*_{f_{min},\epsilon}(x) + \frac{\Lambda}{\text{poly}(n)}$; moreover, $\hat{S}_{f_{med},\epsilon}(x)$ can be computed in time $O\left(\frac{\log^2 n}{\epsilon^2}\right)$.*

Proof. That $\tilde{S}_{f_{med},\epsilon}$ and $\hat{S}_{f_{med},\epsilon}$ are ϵ-smooth upper bounds on $LS_{f_{min}}$ immediately follows from claims 3.2 and 3.3. For the first part, observe that $U_k(x) = (x_{m+k+1} - x_m) + (x_m - x_{m-k-1}) \leq 2A^{(k)}(x)$. The running time bound is straightforward. For the second part, note that $\hat{S}_{f,\epsilon}(x)$ is always within additive error $e^{-\epsilon k_0}$ from $S^*_{f,\epsilon}(x)$. Substituting $k_0 = O(\log n/\epsilon)$ gives the required additive bound. Recall that $A^{(k)}(x)$ can be computed in time $O(k)$. Therefore, $\hat{S}_{f_{med},\epsilon}(x)$ is computable in time $O(k_0^2) = O\left(\frac{\log^2 n}{\epsilon^2}\right)$. □

3.2 Smooth Sensitivity of the Minimum

Computing the smooth sensitivity of the minimum function is very simple. We present it here for completeness, as it is required for understanding the smooth sensitivity of the MST cost.

Let $f_{\min}(x) = \min(x_1, \ldots, x_n)$, and assume for simplicity that $0 \leq x_1 \leq \ldots \leq x_n \leq \Lambda$. The global sensitivity $GS_{f_{\min}} = \Lambda$. For notational convenience, let $x_k = \Lambda$ for $k > n$.

Claim 3.6. *$S^*_{f_{\min},\epsilon}(x) = \max_{k=0,\ldots,n}(\max(x_{k+1}, x_{k+2} - x_1) \cdot e^{-k\epsilon})$ and can be computed in time $O(n)$.*

Proof. It is easy to see that the largest change in $f_{\min}(x)$ is observed either when some x_i is dropped to 0 or when x_1 is raised to a value at least x_2. Thus, $LS_{f_{\min}}(x) = \max(x_1, x_2 - x_1)$. Similarly, the sensitivity at distance k is $A^{(k)}(x) = \max(x_{k+1}, x_{k+2} - x_1)$. It follows that $S^*_{f_{\min},\epsilon}(x) = \max_{k=0,\ldots,n}(A^{(k)}(x) \cdot e^{-k\epsilon})$ and therefore can be computed in time $O(n)$. □

3.3 Smooth Sensitivity of The Cost of a Minimum Spanning Tree

Let $G = (V, E)$ be an undirected connected graph with edge weights $w(e) \in [0, \Lambda]$ for all $e \in E$, where each edge weight is reported to the database by an individual. Let $f_{\text{MST}}(G)$ be the MST cost in G. The global sensitivity, $GS_{f_{\text{MST}}}$, is Λ because for the complete graph with all weights equal to Λ, the MST cost decreases by Λ when one of the weights is changed to 0. The local (and smooth) sensitivity can be much lower, though. For example, in a 2-connected graph where all edge weights are less than some number w^*, the local sensitivity will be at most w^* (since increasing the weight of one edge drastically will simply lead to it being replaced in the MST by an edge with weight at most w^*). Here we show how to compute the smooth sensitivity of f_{MST} in polynomial time.

The main idea in the analysis is to express the local sensitivity of f_{MST} in terms of the local sensitivity of the minimum function. Let $f_{\min}(x) = \min(x_1, \ldots, x_n)$, where $0 \leq x_1 \leq \ldots \leq x_n \leq \Lambda$. It is not hard to verify that the sensitivity of $f_{\min}$ at distance k is $A^{(k)}(x) = \max(x_{k+1}, x_{k+2} - x_1)$, where $x_k = \Lambda$ for $k > n$.

We will show that the local sensitivity of f_{MST} is the maximum of the local sensitivities of minimum functions, where the maximum is taken over all cuts of G.

First, some notation: A *cut* in G is a partition of the vertices V in two nonempty subsets, S and V/S. With some abuse of terminology, we call $S \subset V$ a cut when we mean partition $(S, V/S)$. We say an edge (i, j) crosses the cut S when $i \in S$ and $j \in V/S$. For a cut $S \subset V$, let $w_t(S)$ denote the weight of the t-th lightest edge in the cut, i.e., the t-th element in the list $\{w((i, j)) \mid i \in S, j \in V/S\}$, sorted in non-decreasing order. Note that $w_t(s) = \Lambda$ if the cut S has less than t edges. Let $\ell(S) = |\{(i, j) \mid i \in S, j \in V/S\}|$ denote the number of edges crossing the cut.

Lemma 3.7. *The local sensitivity of f_{MST} at distance k is*

$$A^{(k)}_{f_{MST}}(G) = \max_{S \subset V} A^{(k)}_{f_{\min}}\left(w_1(S), w_2(S), \ldots, w_{\ell(S)}(S)\right) .$$

We defer the proof of Lemma 3.7 to the next section (Section 3.3.1). For now, note that one can substitute $A^{(k)}_{f_{\min}}(w_1, \ldots, w_t) = \max(w_{k+1}, w_{k+2} - w_1)$, where $w_k = \Lambda$ for $k > t$, into the expression for $A^{(k)}_{f_{\text{MST}}}(G)$ in Lemma 3.7 and exchange the order of the maxima to get

$$A^{(k)}_{f_{\text{MST}}}(G) = \max(\max_{S \subset V} w_{k+1}(S), \max_{S \subset V}(w_{k+2}(S) - w_1(S))) \tag{3}$$

Lemma 3.8. *The smooth sensitivity of f_{MST} can be computed in time polynomial in the number of edges.*

Proof. Recall that the smooth sensitivity of f_{MST}, denoted by $S^*_{f_{\text{MST}},\epsilon}(G)$, is $\max_{k\le n} e^{-k\epsilon} A^{(k)}_{f_{\text{MST}}}(G)$. It thus suffices to explain how to compute $A^{(k)}_{f_{\text{MST}}}(G)$. The expression for $A^{(k)}_{f_{\text{MST}}}$ in Equation 4 is a maximum of two terms. We explain how to compute each term separately.

To compute $\max_S w_k(S)$, we compute minimum cuts in several *unweighted* graphs related to G. Let $E_w = \{e \in E \mid w(e) \le w\}$. Let G_w be the (undirected) unweighted graph (V, E_w). Let $c_w = \text{cost}(\text{min-cut}(G_w))$, that is, the number of edges in the minimum cut of G_w. Let $0 \le wt_1 < wt_2 < \ldots < wt_t \le \Lambda$ be the sorted list of distinct weight values in G. Then $t \le |E| < n^2$. We can do a binary search on the list of weights to find i such that $c_{wt_i} \ge k$ and $c_{wt_{i-1}} < k$. (For notational convenience, let $c_{wt_0} = 0$.) Then $\max_{S\subset V} w_k(S) = wt_i$. To see this, note that $c_{wt_i} \ge k$ implies that all cuts in G have at least k edges of weight $\le wt_i$, i.e., $w_k(S) \le wt_i$ for all cuts S in G. On the other hand, $c_{wt_{i-1}} < k$ implies that some cut S in G has $< k$ edges of weight $\le wt_{i-1}$. Thus, $w_k(S) = wt_i$ for this cut S. It follows that $\max_{S\subset V} w_k(S) = wt_i$. If $c_{w_t} < k$ then some cut in G has $< k$ edges. In this case, we set $\max_{S\subset V} w_k(S)$ to Λ. (Recall that for computing the sensitivity of $f_{\min}(x_1, \ldots, x_n)$ we used the convention $x_i = \Lambda$ for $i > n$.) To summarize, $max_{S\subset V} w_k(S)$ can be computed with $O(\log t)$ min-cut computations on G_{wt_i}'s.

Let T be a MST of G. An (i, j)-cut of G is a cut S such that $i \in S$ and $j \in V/S$.

$$\max_{S\subset V}(w_t(S) - w_1(S)) = \max_{e\in T} \max_{e\text{-cuts } S}(w_t(S) - w(e)) = \max_{e\in T}(\max_{e\text{-cuts } S} w_t(S) - w(e)).$$

We can compute $\max_{e\text{-cuts } S} w_t(S)$ the same way as $\max_{S\subset V} w_k(S)$, using calls to a procedure that computes the minimum cut separating a given pair of nodes instead of a general minimum cut. □

3.3.1 Analyzing the Sensitivity of the MST: Proof of Lemma 3.7

We first analyze the local sensitivity of the MST:

Lemma 3.9. [3.7a] *The local sensitivity* $LS_{f_{MST}}(G) = \max_{S\subset V} LS_{f_{\min}}\left(w_1(S), w_2(S), \ldots, w_{a(S)}(S)\right)$.

The lemma follows from Claims 3.10 and 3.11, which give upper and lower bounds, respectively, on the local sensitivity of f_{MST}, and the fact that $LS_{f_{\min}}(x) = \max(x_1, x_2 - x_1)$, which was shown in the proof of Claim 3.6.

Claim 3.10. $LS_{f_{MST}}(G) \le \max(w_1(S), w_2(S) - w_1(S))$ *for some* $S \subset V$.

Proof. Let $e = (i, j)$ be an edge such that changing the weight of e from w to w' maximizes the change in f_{MST}. Let $f = f_{\text{MST}}(G)$ and $f' = f_{\text{MST}}(G')$ where G' is G with $w(e)$ changed to w'. We will consider two cases: $w < w'$ and $w > w'$. In the first case, we will show that $f' - f \le w_2(S) - w_1(S)$ for some $S \subset V$. In the second case, that $f - f' \le w_1(S)$ for some $S \subset V$.

Case 1: $w < w'$. Observe that when the weight of e is increased, f_{MST} can increase only if e is in all MSTs of G. Let T be a MST of G. Removing e from T breaks it into two connected components. Let S be the nodes of the connected component containing i. By a standard argument (see, e.g., [19]), $w = w_1(S)$, since e is the only edge in T crossing the cut S. (Otherwise, one can replace it with a lighter edge crossing S to get a lighter MST for G.) Let T' be a graph obtained from T by replacing e with $e_2(S)$. T' is a spanning tree of G because it has $n - 1$ edges and is connected. Its cost in G is $f + w_2(S) - w_1(S)$. Since T' does not contain e, it has the same cost in G'. Therefore, $f' \le f + w_2(S) - w_1(S)$, as required.

Case 2: $w > w'$. Similarly, when the weight of e is decreased, f_{MST} can decrease only if e is in all MSTs of G'. Let T be a MST of G'. Define S as in the previous paragraph. T is a spanning tree in G with cost $f' + w - w'$. Thus, $f \le f' + w - w'$. If $w = w_1(S)$ (where the weights are ordered according to G, not

G') then $f - f' \leq w_1(S)$, as required. Otherwise (i.e., if w is not the lightest weight in the cut S), consider the graph T' obtained from T by replacing e with $e_1(S)$. As before, T' is a spanning tree of G. Its cost is $f' + w_1(S) - w'$. Thus, $f \leq f' + w_1(S) - w' \leq f' + w_1(S)$, as required. □

Claim 3.11. *$LS_{f_{MST}}(G) \geq \max(w_1(S), w_2(S) - w_1(S))$ for all $S \subset V$.*

Proof. Consider some $S \subset V$. We will show how to change one weight in G in order to (a) decrease f_{MST} by $w_1(S)$; (b) increase f_{MST} by $w_2(S) - w_1(S)$. Let T be a MST of G. By a standard argument, T contains an edge $e = (i, j)$ with $i \in S, j \in V/S$ and $w(e) = w_1(S)$.

(a) Let G' be a graph obtained from G by setting $w(e)$ to 0. T is a spanning tree in G' of weight $f - w_1(S)$. Thus, $f_{\text{MST}}(G') \leq f - w_1(S)$, and therefore $S_{f_{\text{MST}}}(G) \geq w_1(S)$.

(b) Let G' be a graph obtained from G by setting $w(e)$ to $w_2(S)$. Let $f' = f_{\text{MST}}(G')$. Suppose for contradiction that $f' < f + w_2(S) - w_1(S)$, namely, that some spanning tree T in G' has cost $<$ $f + w_2(S) - w_1(S)$. To reach a contradiction, we will show that G has a spanning tree of cost $< f$. There are two cases to consider: $e \in T$ and $e \notin T$. If $e \in T$ then T is a spanning tree in G of cost $< f$, and we are done. Suppose $e \notin T$. Recall that $e = (i, j)$. There is a unique path from i to j in T. Let $e' = (i', j')$ be the first edge on the path with $i' \in S$ and $j' \in V/S$. Let T be the graph obtained from T by replacing e' with e. This graph is a spanning tree. Its cost in G is $< f$, a contradiction. It concludes the proof that $S_{f_{\text{MST}}}(G) \geq w_2(S) - w_1(S)$. □

Lemma 3.9 expresses the local sensitivity of f_{MST} as the maximum of the local sensitivities of minimum functions. The next lemma gives an analogous statement for the local sensitivity at distance k. The lemma will be crucial for computing the smooth sensitivity of f_{MST}.

Lemma 3.12 (Lemma 3.7 restated)**.** *The local sensitivity at distance k is*

$$A^{(k)}_{f_{MST}}(G) = \max_{S \subset V} A^{(k)}_{f_{\min}}\left(w_1(S), w_2(S), \ldots, w_{\ell(S)}(S)\right).$$

Proof. Recall that $w_t(S)$ denotes the weight of the t-th lightest edge in the cut S. For this proof, we will make the notation more specific: $w^G_t(S)$ will be the weight of the t-th lightest edge in the cut S according to the weights of G.

$$\begin{aligned}
&A^{(k)}_{f_{\text{MST}}}(G) \\
&= \max_{G' : d(G,G') \leq k} LS_{f_{\text{MST}}}(G') \\
&= \max_{G' : d(G,G') \leq k} \left(\max_{S \subset V} LS_{f_{\min}}\left(w^{G'}_1(S), \ldots, w^{G'}_{\ell(S)}(S)\right)\right) \\
&= \max_{S \subset V} \left(\max_{G' : d(G,G') \leq k} LS_{f_{\min}}\left(w^{G'}_1(S), \ldots, w^{G'}_{\ell(S)}(S)\right)\right) \\
&= \max_{S \subset V} A^{(k)}_{f_{\min}}\left(w^{G}_1(S), \ldots, w^{G}_{\ell(S)}(S)\right).
\end{aligned}$$

The equalities follow from Definitions 1.6 and 3.1 and Lemma 3.9. □

By the proof of Claim 3.6, $A^{(k)}_{f_{\min}}(w_1, \ldots, w_t) = \max(w_{k+1}, w_{k+2} - w_1)$. Substituting this into the expression for $A^{(k)}_{f_{\text{MST}}}(G)$ in Lemma 3.12 and exchanging the order of the maxima, we get

$$A^{(k)}_{f_{\text{MST}}}(G) = \max\left(\max_{S \subset V} w_{k+1}(S), \max_{S \subset V}(w_{k+2}(S) - w_1(S))\right). \tag{4}$$

3.4 Smooth Sensitivity of the Number of Triangles in a Graph

In this subsection, we explain how to compute the smooth sensitivity of the function that counts the number of triangles in a graph. Consider an undirected graph on n nodes, represented by an adjacency matrix[3] x, where each entry x_{ij} is reported to the database by an individual. Since $x_{ij} = x_{ji} \forall i, j \in [n]$, we let the Hamming distance between adjacency matrices x and x' be the number of entries with $i \geq j$ on which they differ. Let $f_\triangle(x)$ be the number of triangles in the graph represented by x.

Observe that by adding or deleting an edge (i, j), we add or delete all triangles of the form (i, k, j) for all $k \in [n]$. Let a_{ij} represent the number of triangles involving a potential edge (i, j): that is, $a_{ij} = \sum_{k\in[n]} x_{ik} \cdot x_{kj}$. Similarly, let b_{ij} be the number of half-built triangles involving a potential edge (i, j): $b_{ij} = \sum_{k\in[n]} x_{ik} \oplus x_{kj}$. With this notation, $LS_{f_\triangle}(x) = \max_{i,j\in[n]} a_{ij}$. In contrast, $GS_{f_\triangle} = n - 2$.

Claim 3.13. *The local sensitivity of $f_\triangle$ at distance s is*

$$A^{(s)}(x) = \max_{i\neq j; i,j\in[n]} c_{ij}(s) \text{ where } c_{ij}(s) = \min\left(a_{ij} + \left\lfloor \frac{s + \min(s, b_{ij})}{2} \right\rfloor,\ n - 2\right).$$

Proof. First we show that $A^{(s)}(x) \geq c_{ij}$ for all $i, j \in [n]$. Fix distinct $i, j \in [n]$. Let y be the adjacency matrix for the graph represented by x with the following s additional edges: (1) for $\min(b_{ij}, s)$ indices k with $x_{ik} \neq x_{jk}$, add the missing edge (i, k) or (j, k); (2) if $s > b_{ij}$, for $\left\lfloor \frac{s-b_{ij}}{2} \right\rfloor$ indices k with $x_{ik} = x_{jk} = 0$, add edges (i, k) and (j, k). Let y' be the same as y with one exception: set $y'_{ij} = y'_{ji}$ to $1 - y_{ij}$. Then $d(x, y) \leq s, d(y, y') = 1$, and $|f_\triangle(y) - f_\triangle(y')| = c_{ij}(s)$. Thus, $A^{(s)}(x) \geq c_{ij}$.

Now we prove that $A^{(s)}(x) \leq c_{ij}$ for some $i, j \in [n]$. Let y, y' be the adjacency matrices such that $d(x, y) \leq s, d(y, y') = 1$ and $A^{(s)}(x) = |f_\triangle(y) - f_\triangle(y')|$. Let (i, j) be an edge on which y and y' differ. Then $A^{(s)}(x) \leq c_{ij}$ because graphs represented by y and y' only differ on triangles involving edge (i, j). There are a_{ij} such potential triangles in x, and one can add at most $\left\lfloor \frac{s+\min(s,b_{ij})}{2} \right\rfloor$ such potential triangles by adding at most s edges. □

Claim 3.14. *The ϵ-smooth sensitivity of $f_\triangle$ is computable in time $O(M(n))$, where $M(n)$ is the time required for multiplying two matrices of size $n \times n$.*

Proof. Observe that $a_{ij} = (x^2)_{ij}$ and $b_{ij} = degree(i) + degree(j) - 2a_{ij} - 2x_{ij}$. Thus, both a_{ij} and b_{ij} for all $i, j \in [n]$ can be computed in time $O(M(n))$. From these values, one can compute $A^{(k)}(x)$ for $k = 0, \ldots, n$ in time $O(n^2 \log n)$ as follows. Sort pairs (a_{ij}, b_{ij}) so that a_{ij}s are non-decreasing. If there are several pairs with the same a_{ij}, keep only the pair with the largest b_{ij}. Go through the list in the order of increasing a_{ij} and for each (a, b), keep this pair only if $2a - b > 2a' - b'$ for the previous surviving pair (a', b'). Call the surviving pairs $(a_1, b_1), \ldots, (a_t, b_t)$. Let $s_0 = 0, s_i = 2(a_i - a_{i+1}) + b_i$ for $i \in [t - 1]$, and $s_t = 2(n - 2 - a_t) - b_t + 1$. Then $A^{(s)}(x) = a_i + \left\lfloor \frac{s+\min(s,b_i)}{2} \right\rfloor$ if $s \in [s_{i-1}, s_i)$. □

To compare the noise magnitude $S^*_{f_\triangle,\epsilon}$ with $\frac{GS_{f_\triangle}}{\epsilon} = \frac{n}{\epsilon}$, the noise magnitude obtained from the global sensitivity framework, we will look at how much noise we are adding in random graphs. In one commonly studied random graph model, $G(n, p)$, a graph on n vertices is obtained by independently including an edge between each pair of vertices with probability p.

Under $G(n, p)$, for each (i, j) the distribution of a_{ij} is Binomial$(n - 1, p^2)$ and the distribution of b_{ij} is Binomial$(n - 1, 2p(1 - p))$. Thus, with high probability, $a_{ij} \leq O(np^2 \log n)$ and $b_{ij} \leq O(np \log n)$ for all

[3] What follows can be easily changed to deal with adjacency lists representation.

i, j. That is, $c_{ij}(s) \leq s + O(np^2 \log n)$ for all i, j. Therefore, with high probability $N^*_{f,\epsilon} \leq \frac{1}{\epsilon} \max_s((s + O(np^2 \log n))e^{-\epsilon s})$. As long as $1/\epsilon$ is less than $np^2 \log n$, this expression is maximized for $s = 1$, yielding $S^*_{f,\epsilon} = O(\frac{np^2 \log n}{\epsilon})$.

3.5 Computing Local Sensitivity is Hard for Some Functions

One can construct instances where $LS_f(x)$ and $S^*(x)$ are difficult to compute or even approximate. Suppose that $D = \{0,1\}^n$ and we have a function $f : D^n \to \{0,1\}$ defined as follows: $f(x) = \phi_{x_2,...,x_n}(x_1)$, where $\phi_{x_2,...,x_n}$ is a boolean formula defined by $x_2, ..., x_n$. Specifically, if $x_2 = \cdots = x_n$, and x_2 is the binary description of some formula over n variables, then let $\phi_{x_2,...,x_n}$ be that formula. Otherwise, set $\phi_{x_2,...,x_n} \equiv 0$. (To see why an n-bit string can describe a formula over n variables, imagine that it is restricted to formulas which depend only on, say, the first $n^{1/3}$ bits.) If $\phi_{x_2,...,x_n}$ is satisfiable, then the local sensitivity and $S^*(x)$ are 1. Otherwise, the local sensitivity is 0 and $S^*(x) = \exp(-\beta n)$.

4 Sample-Aggregate Framework

4.1 A Motivating Example: Clustering

One motivating example for the sample and aggregate framework is privately releasing k-means cluster centers. In k-squared-error-distortion (k-SED) clustering (also called "k-means"), the input is a set of points $x_1, ..., x_n \in \mathbb{R}^\ell$ and the output is a set of k centers $c_1, ..., c_k$ with minimum cost. The cost of a set of centers is the sum over i of the squared Euclidean distance between x_i and the nearest center:

$$\text{cost}_x(c_1, ..., c_k) = \frac{1}{n} \sum_{i=1}^{n} \min_j \|x_i - c_j\|_2^2 .$$

Let f_{cc} denote a deterministic function which takes a data set x and outputs a set of cluster centers. It is convenient to think of f_{cc} as producing optimal centers, although in general the output may only be approximately optimal.

Wasserstein Distance. To apply the sensitivity framework to clustering, we have to be able to compute distances between sets of cluster centers. To this end, we equip the output space of a clustering algorithm, $\mathcal{M} = \left(\mathbb{R}^\ell\right)^k$, with a meaningful metric. $L_2^{\ell k}$ is not appropriate, since under this metric, two permutations of the same set of centers can be very far apart. Instead, we use a variant of the earthmover metric, called the *Wasserstein distance* [30]. Given two sets of candidate centers, we take the $L_2^{\ell k}$ distance *under the best possible permutation of the centers in each set*, that is:

$$d_{\mathsf{W}}\big(\{c_1, ..., c_k\}, \{\hat{c}_1, ..., \hat{c}_k\}\big) = \Big(\min_{\pi \in S_k} \sum_{j=1}^{k} \|c_j - \hat{c}_{\pi(j)}\|_2^2\Big)^{\frac{1}{2}}.$$

It is easy to verify that this defines a metric modulo the equivalence between permutations of the same multi-set. (We ignore the technicality and call the resulting structure $\mathcal{M}$ a metric space). One can compute the Wasserstein distance d_{W} efficiently: Given two sets of centers, find the minimum weight perfect matching in the balanced bipartite graph on $2k$ vertices, where the weight of edge (i, j) is given by $\|c_i - \hat{c}_j\|_2^2$ (that is, the minimum weight perfect matching describes the optimal permutation in the distance definition), and take the square root of the result. Several variants on this definition would work well for our purposes; we use Wasserstein's for convenience.

Adding Noise with respect to Earthmover-like Metrics For the purposes of adding noise to point-sets in $\mathbb{R}^d$ (i.e., points in $\mathcal{M}$), we sample the output by first permuting the point set randomly, and then adding Gaussian noise, as in Lemma 2.10. This amounts to treating $\mathcal{M}$ as L^{dk} modulo the equivalence class induced by permuting the k pieces of dimension d randomly. If S is a smooth upper bound to the local sensitivity of $f : D^n \to \mathcal{M}$ (where sensitivity is measured w.r.t the Wasserstein metric on $\mathcal{M}$), then one can add noise with standard deviation $S(x)\frac{\log(1/\delta)}{\beta}$ to each coordinate of $f(x)$ and obtain an (ϵ, δ)-differentially private mechanism. This follows essentially from the fact that L_2 distance is always an upper bound on the Wasserstein metric.

Sensitivity of Clustering. Let $f_{\text{cc}}(x)$ denote the k-SED cluster centers, where x_i are points from a subset of $\mathbb{R}^\ell$ with diameter Λ. (If the set of optimal cluster centers is not unique, $f_{\text{cc}}(x)$ outputs the lexicographically first set of centers.) The *cost* of the optimal clustering has global sensitivity at most Λ/n, since moving one point changes its distance to the closest center by at most Λ. The global sensitivity of f_{cc} is much higher: $\Omega(\Lambda)$. For example, in the instance depicted in Fig. 2, changing a single point moves the two optimal centers by $\Omega(\Lambda)$. Thus, adding noise to $f_{\text{cc}}(x)$ according to the global sensitivity essentially erases all centers completely. In contrast, intuition suggests that in "well-clustered" instances (where there is a single reasonable clustering of the data), the local sensitivity should be low since moving a few data points should not change the centers significantly. However, we do not know how to approximate a smooth bound on $LS_{f_{\text{cc}}}$ efficiently.

We circumvent this difficulty by relying on a different intuition: in well-clustered instances, random samples from the data should have approximately the same centers as the original data set, and this can be verified efficiently. This section describes a framework for releasing functions which fit this intuition. The application to f_{cc} appears in Section 5.

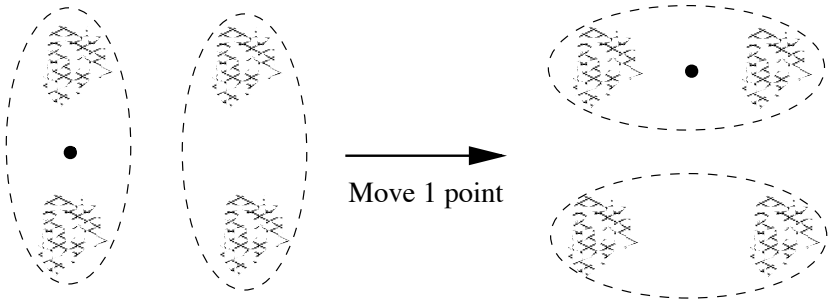

Figure 2: A sensitive 2-SED instance

4.2 Basic Framework

In this section we define the sample and aggregate framework and state the main theorem on its performance. In what follows, unless specified otherwise, $\mathcal{M}$ denotes a metric space with distance function $d_{\mathcal{M}}(\cdot, \cdot)$ and diameter Λ.

Suppose that f is defined on databases of all sizes, so that it makes sense to apply f to a random sample of the data points. Further, suppose that for a particular input $x \in D^n$, the function value $f(x)$ can be approximated well by evaluating f on a random sample of $o(n)$ points from the database. We prove below that in such cases one can release the value $f(x)$ with relatively small expected error.

Sample and aggregate works by replacing f with a related function $\bar{f}$ for which smooth sensitivity is low and efficiently computable. The first step in the framework can be thought of as randomly partitioning the

database x into m small databases, where m is a parameter in the construction. To simplify the analysis, we construct the small databases by taking independent random samples from x instead of actually partitioning x. Let $U_1, ..., U_m$ be random subsets of size n/m independently selected from $\{1, \ldots, n\}$. Each subset is obtained by taking uniform random samples without replacement. Let $x|_U$ denote the subset of x with indices in U. We evaluate f on m small databases $x|_{U_1}, \ldots, x|_{U_m}$ to obtain values $z_1, ..., z_m$ in the output space of f. Finally, we apply a carefully chosen *aggregation* function g, called the *center of attention* and defined in Section 4.4, to the values $z_1, ..., z_m$. The output of this computation, $\bar{f}(x) = g\left(f(x|_{U_1}), \ldots, f(x|_{U_m})\right)$, is released using the smooth sensitivity framework (Figure 3).

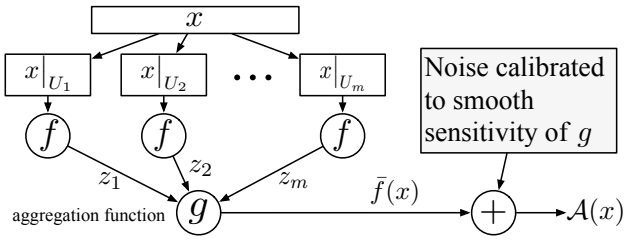

Figure 3: The Sample-Aggregate Framework

We will bound the smooth sensitivity of the function $\bar{f}$ at x by the smooth sensitivity of the aggregation function g at $z = (z_1, ..., z_m)$. The idea is that changing a single point in the original database x will change very few small databases, and hence very few evaluations $z_1, \ldots, z_m$. This requires a slight generalization of local sensitivity and related notions to handle more than one change to the input.

The bulk of the work in the analysis of the sample and aggregate framework is in carefully choosing the aggregation function so that (a) if most of the z_i's are close to some point, then $\bar{f}(x)$ is close to that point, and we can efficiently compute a smooth, fairly tight upper bound on the local sensitivity of $\bar{f}$.

Sample and aggregate outputs accurate answers on databases x on which $f(x)$ is approximated well by evaluating f on random samples from the database. The following definition quantifies what we mean by a good approximation.

Definition 4.1. *A function $f : D^* \to \mathcal{M}$ is approximated to within accuracy r* on the input x *using samples of size n' if* $\Pr_{\substack{U \subset [n], \\ |U| = n'}} \left[d_{\mathcal{M}}\Big(f(x|_U),\ f(x)\Big) \leq r \right] \geq \frac{3}{4}$.

For example, for the clustering application, this definition says that with probability $3/4$, the cluster centers of a small random sample from the database are within the Wasserstein distance r from the centers of the whole database.

Sample and aggregate is applicable for a function with any output space metric $\mathcal{M}$ for which there is an admissible noise process (see Definition 2.5). The performance of the framework depends on the parameters of the noise process. We summarize the performance of sample and aggregate for the case of $\mathcal{M} = L_1$ in Theorem 4.2. The corresponding statements for L_2 and various versions of the Wasserstein distance have more parameters, but are similar. Further below, we develop machinery for general metrics that, combined with results on adding noise from Section 2, yields a proof of Theorem 4.2 and its analogues for other metric spaces.

Theorem 4.2 (Main)**.** *Let $f : D^* \to \mathbb{R}^d$ be an efficiently computable function with range of diameter Λ and L_1 metric on the output space. Set $\epsilon > \frac{2d}{\sqrt{m}}$ and $m = \omega(\log^2 n)$. The sample-aggregate mechanism $\mathcal{A}$ is an ϵ-differentially private efficient mechanism. Moreover, if f is approximated within accuracy r on*

the database $x = (x_1, \ldots, x_n)$ *using samples of size* $\frac{n}{m}$*, then each coordinate of the random variable* $\mathcal{A}(x) - f(x)$ *has expected magnitude* $O\left(\frac{r}{\epsilon}\right) + \frac{\Lambda}{\epsilon} e^{-\Omega(\frac{\epsilon\sqrt{m}}{d})}$.

This result captures the intuition that if a function f can be approximated by sampling *on a particular input* x, then we can release $f(x)$ with small additive error. In the special case where ϵ is constant, we get the following:

Corollary 4.3. *Suppose* ϵ *is constant. If* f *is approximated within accuracy* r *on input* x *using samples of size* $o\left(\frac{n}{d^2 \log^2 n}\right)$*, then* $\mathcal{A}$ *releases* $f(x)$ *with expected error* $O(r) + \Lambda \cdot negl\left(\frac{n}{d}\right)$ *in each coordinate.*

The notation $negl()$ above denotes a *negligible function*, that is a positive function $g(x)$ that is asymptotically smaller than any inverse polynomial as x tends to infinity: $g(x) = 1/x^{\omega(1)}$.

In many natural settings where privacy is important, the database itself consists of a random sample taken from some underlying population. In that case, one can think of $f(x)$ as an approximation to some statistics about the population on a sample of size n. Corollary 4.3 states that privacy imposes only a slight degradation of the quality of the results: the server answers queries with accuracy corresponding so a sample of size $\tilde{\Omega}(n/d^2)$ while ensuring privacy.

Remark 1. A variant of Theorem 4.2 still holds if in Definition 4.1, the values $f(x|_U)$ lie near some specific value c, not necessarily $f(x)$ (that is, assume that $d_{\mathcal{M}}\Big(f(x|_U),\ c\Big) \leq r$ with probability at least 3/4). In that case, the analogous statement is that $\mathcal{A}(x) - c$ has expected magnitude $O\left(\frac{r}{\epsilon}\right) + \frac{\Lambda}{\epsilon} e^{-\Omega(\frac{\epsilon\sqrt{m}}{d})}$ in each coordinate.

This generality will be useful for both applications of sample and aggregate discussed in Section 5. For example, for k-SED clustering, the function f evaluated on each sample $x|_U$ will be a polynomial-time algorithm which outputs a set of cluster centers of near-optimal cost. In contrast, the value c is the set of *optimal* cluster centers, $c = f_{\text{cc}}(x)$, which is NP-hard to compute.

4.3 Good Aggregations for General Metric Spaces

Good Aggregations. Before defining a valid aggregation, we generalize local sensitivity and related notions to handle several changes to the input. It is not hard to prove that in the sample and aggregate framework, if we independently select m small databases of size n/m (each chosen uniformly without replacement), then with probability at least $1 - 2^{-\sqrt{m}+\log n}$, no point appears in more than $\sqrt{m}$ small databases. Hence, with high probability, each point in x affects at most $\sqrt{m}$ of the inputs to g. This observation leads us to a slight generalization of local sensitivity and smooth bounds, where we consider how changing up to s points in the input affects the output. For the application to sample and aggregate, we will set s to $\sqrt{m}$.

Definition 4.4. *For* $g : D^m \to \mathcal{M}$ *and* $z \in D^m$*, the local sensitivity of* g *at* x *with step size* s *is*

$$LS_g^{(s)}(z) = \max_{z' : d(z,z') \leq s} d_{\mathcal{M}}\left(g(z),\ g(z')\right) .$$

For $\beta > 0$*, a function* $S : D^m \to \mathbb{R}^+$ *is a* β*-smooth upper bound on the sensitivity of* g *with step size* s *if*

$$\forall z \in D^m : \quad S(z) \geq LS_g^{(s)}(z) ;$$
$$\forall z, z' \in D^m ,\ d(z, z') \leq s \ : \quad S(z) \leq e^{\beta} S(z') .$$

An aggregation function g comes with a corresponding smooth upper bound S on its sensitivity with step size s. When most of the input to g is contained in a small ball, (a) the output of g should be close to the ball and (b) S should be small. Let $\mathcal{B}(c, r)$ denote a ball around c of radius r.

Definition 4.5 (Good Aggregation)**.** *In a metric space $\mathcal{M}$ with diameter Λ, an (m, β, s)-aggregation is a pair of functions, an aggregation function $g : \mathcal{M}^m \to \mathcal{M}$ and a sensitivity function $S : \mathcal{M}^m \to \mathbb{R}^+$, such that*

1. *S is a β-smooth upper bound on $LS_g^{(s)}$.*
2. *If at least $\frac{2m}{3}$ entries in z are in some ball $\mathcal{B}(c, r)$ then*
   (a) *$g(z) \in \mathcal{B}(c, O(r))$;*
   (b) *$S(z) = O(r) + \Lambda \cdot e^{-\Omega(\beta m/s)}$.*

If g and S are computable in time $poly(m, 1/\beta)$, the aggregation is efficient.

Intuitively, the existence of a good aggregation implies that given a collection of points, most of which are contained in a small ball, it is possible to return a representative point that is not far from the ball, while preserving privacy. This ensures that the sample and aggregate mechanism returns a value $\bar{f}(x)$ close to $f(x)$ when at least $2/3$ of the values $z_i = f(x|_{U_i})$ lie close to $f(x)$. Condition (2b) ensures that not too much noise is added to $\bar{f}(x)$.

Example 2. When f takes values in $[0, \Lambda] \subseteq \mathbb{R}$, the median is an efficient good aggregation in the sense of Definition 4.5. To see why, note that if $2/3$ of the points in z are in an interval $\mathcal{B}$ of length $2r$, then the median is also contained in this interval. Condition (2a) is hence satisfied as the median is within distance r from the center of $\mathcal{B}$.

$S(z) = \max_{k=0,\ldots,n/6} LS_g^{(s(k+1))}(z) \cdot e^{-\beta k}$ is a β-smooth bound on $LS_g^{(s)}$ for any function g. When g is the median, $LS_{med}^{(s)}$ is efficiently computable, using formulas similar to those in Section 3.1, and so S is also efficiently computable.

For Condition (2b), note that if 2/3 of the points lie in $\mathcal{B}$, the term $LS_{med}^{(s(k+1))}(z)$ is at most $2r$ for $k = 0, ..., \frac{m-1}{6s}$ (if fewer than $m/6$ points get moved, the median is constrained to remain in $\mathcal{B}$). For larger k, we bound $LS_{med}^{(s(k+1))}(z)$ from above by the diameter Λ. Thus, $S(z) \leq 2r + \Lambda \cdot e^{-\frac{\beta m}{6s}}$ when $2/3$ of the inputs lie in $\mathcal{B}$. $\diamondsuit$

In general metric spaces, one can define a median to be a point minimizing the sum of the distances to all points in a dataset. This satisfies condition (2a) in any space. However, this median may be difficult to compute (it is NP-hard in, for example, permutation spaces [2]). Moreover, we must find a smooth bound on its local sensitivity. We propose an efficient aggregation that meets Definition 4.5 for *arbitrary metric spaces* (as long as computing pairwise distances is feasible). Our aggregator, the *center-of-attention*, is related to the median and is defined and discussed in Section 4.4.

4.4 The Center of Attention is a Good Aggregation

To prove Theorem 4.2, we propose an aggregation function computable in any metric space, called the *center of attention*. It depends only on pairwise distances between the points of the dataset. We start by

defining a simpler *unconstrained center of attention*, a good aggregation related to the center of attention, which might not be efficient in some metric spaces.

Let the input to the aggregation be a set $z \subseteq \mathcal{M}$ of m points in the metric space $\mathcal{M}$. For every point $c \in \mathcal{M}$ (not necessarily in z) define $r(c,t)$ to be its t-radius with respect to z, i.e., the distance from c to its t-th nearest neighbor in z (for $t > m$, set $r(c,t) = \Lambda$). Our aggregation is based on minimizing the t-radius for different values of t. Adding noise proportional to the t-radius was used very differently, but also in the context of data privacy, by Chawla et al. [7]. We are not aware of a direct connection between the two techniques.

4.4.1 A Good But Inefficient Aggregation

Our first aggregation is good for any metric space $\mathcal{M}$, but it might not be efficiently computable in some spaces.

Definition 4.6. *The unconstrained center of attention, $g_0(z)$, of the set $z \in \mathcal{M}^m$ is a point in $\mathcal{M}$ with the minimum t_0-radius, where $t_0 = (\frac{m+s}{2} + 1)$.*

In the metric spaces we consider, namely, finite spaces, L_p metrics, and variants of Wasserstein, the unconstrained center of attention always exists (by compactness), though it may not be unique. One can make g_0 a function by choosing an arbitrary minimizer for each z. Consider the ball of the minimum t_0-radius centered at $g_0(z)$. The number t_0 is chosen so that when s points are removed from z, a majority of the remaining points is contained inside the ball. This lets us bound $LS_{g_0}^{(s)}$.

Let $r^{(z)}(t)$ be the minimum t-radius of any point in $\mathcal{M}$, i.e., the minimum over $c \in \mathcal{M}$ of $r(c,t)$. We define the sensitivity function corresponding to g_0 as

$$S_0(z) \stackrel{\text{def}}{=} 2 \max_{k \geq 0} \left(r^{(z)}\left(t_0 + (k+1)s\right) e^{-\beta k} \right).$$

Ignoring efficiency, the pair (g_0, S_0) is a good aggregation.

Claim 4.7. *The pair (g_0, S_0) is an (m, β, s)-aggregation in any metric.*

Proof. First note that for any two sets z and z' differing in at most s points, every ball which contains $t+s$ points in z' must also contain t points in z. Hence,

$$r^{(z)}(t) \leq r^{(z')}(t+s). \tag{5}$$

Now consider $LS_{g_0}^{(s)}(z)$. Suppose the set z' differs from z in s points, and consider (a) the ball of radius $r^{(z')}(t_0)$ centered at $g_0(z')$ and (b) the ball of radius $r^{(z)}(t_0)$ centered at $g_0(z)$. Note that t_0 was chosen so that both balls contain a strict majority of the points in $z \cap z'$. Thus, they intersect in at least one point. The distance $d(g_0(z), g_0(z'))$ is at most $r^{(z)}(t_0) + r^{(z')}(t_0)$. By Eq. (5), this is bounded above by $r^{(z)}(t_0) + r^{(z)}(t_0 + s) \leq 2r^{(z)}(t_0 + s)$. This yields:

$$LS_{g_0}^{(s)}(z) \leq 2r^{(z)}(t_0 + s).$$

As right hand side above is the first term in the maximum defining $S_0(z)$, we get that S_0 bounds the local sensitivity correctly. The smoothness of S_0 follows from Eq. (5):

$$\begin{aligned} S_0(z) &\leq 2 \max_{k \geq 0} \left(r^{(z')}\left(t_0 + (k+2)s\right) e^{-\beta k} \right) \\ &= 2(e^{\beta}) \max_{k' \geq 1} \left(r^{(z')}\left(t_0 + (k'+1)s\right) e^{-\beta k'} \right) = e^{\beta} S_0(z'). \end{aligned}$$

It remains to prove that when the set z is concentrated in a ball of radius r, then $g_0(z)$ is close to the ball and $S_0(z)$ is close to $O(r)$ (properties 2a and 2b from Definition 4.5). Suppose that $\frac{2m}{3}$ of the inputs lie in $\mathcal{B}(c,r)$. Then radii $r^{(z)}(t)$ are at most r for all $t \leq \frac{2m}{3}$.

Property 2a: The ball of radius $r^{(z)}(t_0)$ which defines $g_0(z)$, must intersect with $\mathcal{B}(c,r)$ in at least one database point. The centers can be at distance at most $2r$, and so $g_0(z) \in \mathcal{B}(c,2r)$.

Property 2b: In the maximum defining $S_0(z)$, the first $\frac{m}{6s}$ terms are at most r, and the remaining ones are at most Λ. Therefore, $S_0(z) \leq 2\max\left(r\ ,\ \Lambda \cdot e^{-\frac{\beta m}{6s}}\right)$, which satisfies the requirements of a good aggregation. □

4.4.2 An Efficient Aggregation: the Center of Attention

To get an efficient aggregation, we "approximate" the unconstrained center of attention with the best point in the input z. Recall that the unconstrained center of attention of the set z is a point in $\mathcal{M}$ with the minimum t_0-radius.

Definition 4.8. *The center of attention, $g(z)$, of the set $z \in \mathcal{M}^m$ is a point in z with the minimum t_0-radius, where $t_0 = (\frac{m+s}{2} + 1)$.*

Recall that $r(c,t)$ is the t-radius of a point $c \in \mathcal{M}$. Let $r_1(t), r_2(t), \ldots, r_m(t)$ be the sorted $\{r(c,t)\}_{c \in z}$ (smallest to largest). We can compute the sorted lists for all $t \in [m]$ by computing all pairwise distances within z (this costs $\binom{m}{2}$ distance computations). It takes $O(m^2 \log m)$ time to generate the sorted list $r_1(t), r_2(t), \ldots, r_m(t)$.

As with the unrestricted center of attention, $LS_g^{(s)} \leq 2r_1(t_0+s)$. Let $a < t_0$ be a parameter of the construction. We will later set $a \approx s/\beta$. In order to compute a smooth bound on the sensitivity, define $\rho(t) = \frac{1}{a}\sum_{i=1}^{a} r_i(t)$. This is an upper bound on $r_1(t)$, and smooth enough to be used as a measure of noise magnitude. Let

$$S(z) = 2\max_k \left(\rho(t_0 + (k+1)s) \cdot e^{-\beta k}\right).$$

Theorem 4.9. *Set $\beta > 2s/m$. In every metric space with efficiently computable pairwise distances, (g, S) is an* efficient *(m, β, s)-aggregation. Computing $g(z)$ and $S(z)$ requires $O(m^2)$ distance computations and $O(m^2 \log m)$ time.*

This theorem, combined with results on adding noise from Section 2, implies the main Theorem 4.2 and its analogues for other metric spaces.

Proof of Theorem 4.9. We noted above that g, S are computable efficiently from the pairwise distances of the point set. Consider two point sets z, z' that differ on s points. Define $r'_i(t)$ and $\rho'(t)$ analogously, but for the set z'.

The following claim is an analogue of (5) in the analysis of the inefficient aggregation. It implies that $\frac{S(z)}{S(z')} \leq e^{\beta}(1 + \frac{s}{a})$. Taking $a = \lceil s/\beta \rceil$ shows that S is $\approx 2\beta$-smooth. We may choose a that sarge since $\beta > 2s/m$ by the hypotheses of the theorem. Also, we may scale β appropriately to get the result we need.

Claim 4.10. *For all $a < t$:* $\quad \rho(t) \leq (1 + \frac{s}{a}) \cdot \rho'(t+s)$

Proof. To prove the claim, we will compare the two sorted lists $r_1(t), ..., r_i(t), ..., r_m(t)$ and $r'_1(t+s), ..., r'_j(t+s), ..., r'_m(t+s)$. We can define a partial matching between them based on the *unchanged* points shared by z and z': if a point c appears in both databases with indices i and j respectively, then define $i = \pi(j)$

and match the corresponding radii $r_{\pi(j)}(t)$ and $r'_j(t+s)$. For every pair of matched entries, we have $r'_j(t+s) \geq r_{\pi(j)}(t)$.

Exactly s entries in each list will be left unmatched (corresponding to the point which was changed). Let I and J be the set of s unmatched indices in each of the two lists. We consider two cases:

Case 1: Suppose that there are no unmatched elements in the first a terms of the second list $r'_1(t+s), ..., r'_m(t+s)$, that is $J \cap [a] = \emptyset$. Then we can compute a lower bound on the sum $\rho'(t+s)$ in terms of a sum of some a elements in the first list, which is itself an upper bound on $\rho(t)$:

$$\rho'(t+s) = \tfrac{1}{a}\sum_{j=1}^{a} r'_j(t+s) \geq \tfrac{1}{a}\sum_{j=1}^{a} r_{\pi(j)}(t) \geq \tfrac{1}{a}\sum_{i=1}^{a} r_i(t) = \rho(t)$$

Case 2: In general, we have to break the sum $\rho'(t)$ into two pieces, according to whether the indices are in or out of J. We can bound the first sum directly:

$$\tfrac{1}{a}\sum_{j\in[a]\setminus J} r'_j(t+s) \geq \tfrac{1}{a}\sum_{j\in[a]\setminus J} r_{\pi(j)}(t) \geq \tfrac{1}{a}\sum_{i=1}^{|[a]\setminus J|} r_i(t)$$

Thus we are using the terms in $[a] \setminus J$ to bound the first $|[a] \setminus J|$ terms of $\rho(t)$.

It remains to find bounds for the remaining terms. We claim that for all $i \leq a$,

$$r_i(t) \leq 2r'_1(t+s).$$

To see this, note that there is ball of radius $r'_1(t+s)$ containing $t+s$ points from z', and hence at least t points from z. Each of these points must have t-radius at most $2r'_1(t+s)$, by the triangle inequality. Since $i \leq a < t$, we can bound $r_i(t) \leq 2r'_1(t+s)$.

To complete the proof, it is convenient to bound $\rho(t)$ above (instead of bounding $\rho'(t+s)$ below). Again, we break $\rho(t)$ into two pieces, which corresponds to breaking $\rho'(t)$ according to indices in or out of J.

$$\begin{aligned}\rho(t) &= \frac{1}{a}\sum_{i\in[a]} r_i(t) = \frac{1}{a}\sum_{i=1}^{|[a]\setminus J|} r_i(t) + \frac{1}{a}\sum_{i=|[a]\setminus J|+1}^{a} r_i(t) \\ &\leq \frac{1}{a}\sum_{j\in[a]\setminus J} r'_j(t+s) + \frac{|[a]\cap J|}{a}(2r'_1(t+s))\end{aligned}$$

Because $r'_1(t+s)$ is the smallest term in the average defining $\rho'(t+s)$, we can regroup the terms in last sum to obtain

$$\rho(t) \leq \rho'(t+s) + \frac{|[a]\cap J|}{a} r'_1(t+s).$$

Finally, since $[a] \cap J$ has at most s elements, and $r'_1(t+s) \leq \rho'(t+s)$, this last inequality yields the desired bound: $\rho(t) \leq (1+\frac{s}{a})\rho'(t+s)$. This proves Claim 4.10. □

The previous claim gave us that $S(z)$ is a smooth upper bound on local sensitivity of g. To complete the proof that (g, S) form a good aggregation it remains to verify that g, S behave properly when the $\frac{2m}{3}$ points in z lie in a ball $\mathcal{B}(c, r)$. This is is very similar to the analysis of g_0, S_0 above: When $i < t < \frac{2m}{3}$, the ith t-radius must be small: $r_i(t) \leq 2r$.

Property 2a: The balls $\mathcal{B}(c, r)$ and $\mathcal{B}(g(z), 2r)$ each contain more than $m/2$ points, and so they intersect. Thus $g(z) \in \mathcal{B}(c, 3r)$.

Property 2b: There are $\frac{3m}{6s}$ terms in the maximum defining $S(z)$ that are at most r, since $\rho(t) \leq r_a(t) \leq r_t(t) \leq r$ for $t \leq \frac{2m}{3}$. Each of the other terms is at most Λ, and hence:

$$S(z) \leq 2 \max \left\{ 2r,\ \Lambda \cdot e^{-\beta(\frac{m}{6}-2)} \right\}$$

assuming the underlying space has diameter at most Λ. This completes the proof of Theorem 4.9. □

Remark 2. One can show that a variation on S can be used as a smooth upper bound on the sensitivity of the *median* in general metric spaces, and also to bound the sensitivity of the standard 2-approximation to the median, given by taking the point in the data set with the minimum sum of distances to other points. However, when used to prove Theorem 4.9 these functions are more complicated to analyze and yield worse constants.

4.5 Using a Good Aggregation: Proof of Theorem 4.2

Suppose we are given f, f' as in the theorem. We can instantiate the sample and aggregate framework as follows, assuming $m \geq \log^2 n$ and n sufficiently large ($n > 35$):

1. Select m random subsets of size n/m, independently and uniformly at random.
2. If any index $i \in [n]$ appears in more than $\sqrt{m}$ of the sets, then reject the sets and repeat Step 1. By Lemma 4.11 below, this step is executed at most $1/(1 - \frac{2n}{(\sqrt{m})!})$ times in expectation. Because $\sqrt{m}! \approx n^{O(\ln \ln n)}$, this expected time is polynomial in n (and at most 2 for n great than 35).
3. Let $z = (f'(x|_{U_1}), ..., f'(x|_{U_m})) \in D^m$. Output $g(z)$ with noise scaled according to $S_{\beta,g}(z)$, where g, S are the aggregation of Theorem 4.9 computed with respect to the L_1 metric.

To prove Theorem 4.2, we must show that this algorithm is a valid privacy mechanism. Each collection of sets $U_1, ..., U_m$ defines a function $\tilde{f}(x) = g\left(f'(x|_{U_1}), ..., f'(x|_{U_m})\right)$. S is a smooth upper bound on the local sensitivity of $\tilde{f}$ since it is a smooth bound with step size s for g: each change in x corresponds to changing at most s points in the input to g. In other words, $S(z)$, viewed as a function of x, satisfies Definition 2.1.

We now turn to bounding the noise added to $f(x)$. Under the conditions of Theorem 4.2, each of the computed samples within distance r of $f(x)$ with probability at least $\frac{3}{4}$. Suppose the sets $U_1, ..., U_m$ were chosen without the added check in Step 2. By a Chernoff bound, with probability $1 - \exp(\Omega m)$, at least $\frac{2}{3}$ of the elements in z would be contained in the ball of radius r centered at $f(x)$. The added conditioning due to Step 2 can change this probability by at most $2n/\sqrt{m}! = n/e^{-\Omega(\sqrt{m})}$, so overall the probability that $\frac{2}{3}$ of the elements lie in a ball of radius r about $f(x)$ is $1 - n/e^{-\Omega(\sqrt{m})}$.

Conditioned on that event, we can apply the properties of the g, S, with $s = \sqrt{m}$. They ensure that we add noise of standard deviation $O(\frac{r}{\epsilon}) + \frac{\Lambda}{\epsilon} e^{-\Omega(\beta\sqrt{m})}$ (where $\beta = \Theta(\epsilon/d)$) in each coordinate, to a center $g(z)$ that is within distance $O(r)$ of $f(x)$. This bias — the difference between $f(x)$ and $g(z)$ — is at most $O(r/\epsilon)$ for $\epsilon < 1$. The expected deviation in each coordinate from $f(x)$ is thus on the order of $\frac{r}{\epsilon} + \frac{\Lambda}{\epsilon} e^{-\Omega(\epsilon\sqrt{m/d})}$, as desired. This completes the proof of Theorem 4.2.

Lemma 4.11. *Suppose $U_1, ..., U_m \subset [n]$ are chosen uniformly and independently from among sets of size n/m. Then the probability that all points $i \in [n]$ are contained in fewer than $s \geq 1$ sets is at least $1 - 2n/s!$.*

Proof. For any given point $i \in [n]$, the number of sets containing it is distributed as $Bin(m, \frac{1}{m})$. For $s \leq m$, the probability that i is included in s or more sets is $\sum_{j \geq s} \binom{m}{s}(1-\frac{1}{m})^{n-j}(\frac{1}{m})^j \leq \sum_{j \geq s} 1/(j!)$. For $s \geq 1$, this sequence decreases by a factor of more than 2 at each term, so the sum is bounded by $2/s!$. The claim follows by a union bound over all i. □

5 Applying the Sample-Aggregate Framework

We illustrate the sample and aggregate framework via two closely related applications: k-SED clustering (also called k-means), discussed in Section 4.1, and learning mixtures of spherical Gaussians. In the case of clustering, we show that instances which are well clustered according to a criterion of [25] behave well with respect to sampling. In the case of learning, we show that polynomially-many i.i.d. samples allow one to release the mixture parameters accurately as long as the components of the mixture do not overlap too heavily.

5.1 Clustering Well-Separated Datasets

Recall that $f_{\text{cc}}(x)$, defined in Section 4.1, denotes the k-SED cluster centers, where x_i are points from a subset of $\mathbb{R}^\ell$ with diameter Λ. Recall that $GS_{f_{\text{cc}}}$ is high and $LS_{f_{\text{cc}}}$ appears difficult to compute or even approximate (though proving that the computation is hard is an open problem). We circumvent these difficulties by using the sample and aggregate framework to release relatively accurate answers on "well-clustered" instances. Ostrovsky et al. [25] introduced a measure of quality of k-SED instances called *separation*. They prove that if x is well-separated, then all near-optimal k-SED clusterings of x induce similar partitions of the points of x. They use this to show that well-separated data sets are amenable to heuristics based on Lloyd's algorithm. We prove that sample and aggregate performs well under similar conditions. In this abstract, we outline the main ideas.

Definition 5.1 (Separation, [25])**.** *Given a dataset x, let $\Delta_k^2(x)$ be the cost of the optimal k-SED clustering of x. We say x is* ϕ^2-separated *if $\Delta_k^2 \leq \phi^2 \Delta_{k-1}^2$.*

An immediate concern is that ϕ^2-separation discusses the cost of the *optimal* solution to the clustering problem. It is NP-hard, in general, to find the optimal clustering of a data set but there exist efficient $O(1)$ approximation algorithms (e.g., [23]). Let f be an A-approximation algorithm that outputs cluster centers of near optimal cost, for some constant A. The sample and aggregate mechanism cannot evaluate f_{cc} efficiently, but f is a reasonable query to the server. Recall that to compare two sets of cluster centers we use the Wasserstein distance, d_{W}, defined in Section 4.1. The following lemma states that for a well-separated instance x, distance $d_{\mathsf{W}}\Big(f(x|_U),\ f_{\text{cc}}(x)\Big)$ is small with probability $\frac{3}{4}$ over the choice of a random subset U of size n/m.

Lemma 5.2. *For some constants C_1, C_2, if $x \in (\mathbb{R}^\ell)^n$ is ϕ^2-separated, where $n' > C_2(\frac{\Lambda^2}{\Delta_{k-1}^2})^2 A^2 k\ell \ln\left(\frac{\ell\Lambda^2}{\Delta_{k-1}^2}\right)$ and $\phi^2 < \frac{C_1}{A+1}$, then*

$$\Pr_{\substack{U \subset [n],\\ |U|=n'}}\left[d_{\mathcal{M}}\Big(\hat{f}_{cc}(x|_U),\ f_{cc}(x)\Big) \leq 60\Lambda\phi^2\sqrt{k}\right] \geq \tfrac{3}{4}\ .$$

The variant of Theorem 4.2 corresponding to Wasserstein distance, together with this lemma, implies that the sample and aggregate mechanism will release a clustering of x with additive noise of magnitude $O(\Lambda\phi^2\sqrt{k}/\epsilon)$ in each coordinate. We conjecture that the actual noise is much lower, but proving the conjecture seems to require a much more refined understanding of the clustering problem.

5.1.1 Proving Lemma 5.2

The intuition behind the proof is that a random sample of the data set will have similar optimal clustering — up to small additive error, as the original data set, since sampling approximates the cost of each candidate clustering well. This was proved in [24], although they consider the related k-median problem.

Lemma 5.3 (Mishra et al. [24])**.** *With probability at least $\frac{3}{4}$ over random samples U of size at least $O((\frac{\Lambda^4 dk}{\gamma^2}\ln(\frac{d\Lambda^2}{\gamma}))$, for all sets of candidate centers c, we have $\mathrm{cost}(c) = \mathrm{cost}_{x|_U}(c) \pm \gamma$.*

Suppose that all sufficiently good clusterings of a data set x are close to each other. When we apply sample and aggregate, if n/m is sufficiently large, then most of the samples will have a near-optimal clustering of x as their best clustering, and the mechanism will output a near-optimal clustering. Intuitively, the existence of an essentially unique optimum clustering is what makes k-means-style clustering problems meaningful (if there are really many more clusters, as in Fig. 2, then we should use a different value of k). Following this intuition, the sample and aggregate framework should perform well in many situations. We formalize this intuition in one particular situation: when the data set satisfies a combinatorial separation condition [25], under which heuristics perform well.

The analysis below assumes f' works with probability 1, but one can easily extend the result to deal with probabilistic guarantees. Note that even though we were given only an A-approximation algorithm, the final guarantee relates the output to an optimal clustering. This is due to the separation condition; it guarantees, in particular, that even approximate solutions are close to the optimum.

We can also use this style of reasoning to show directly that any good approximation algorithm for k-SED has low smooth sensitivity on well-separated inputs. This should not be surprising, since low local sensitivity is in some sense necessary for adding little noise. However, the analysis of the sampling framework yields a qualitatively stronger result: one can apply it some large class of inputs, which include well-separated ones, but one need not first test whether the input is well-separated. In contrast, a direct bound on S^* based on good separation puts the burden on the privacy mechanism to test for good separation, and also to find a way to degrade quality gracefully as inputs become less well separated. This may well be computationally harder. More importantly, such a mechanism could only work for well-separated inputs; other classes where sampling does well (such as those drawn i.i.d. from mixtures with relatively low separation) would be ignored.

Proof of Lemma 5.2. Let $a = \frac{1-401\phi^2}{400}$. We show that (a) any clustering with cost less than $a\Delta^2_{k-1}$ is close to the optimal clustering in the Wasserstein distance, and (b) show that with high probability, under the conditions of the theorem, the output of $f'(x|_U)$ has cost less than $a\Delta^2_{k-1}$.

We start with (b). Under the conditions of the theorem, with probability at least $3/4$ a random sample approximates the cost of every clustering within additive error $\gamma \approx \Delta^2_{k-1}/(A+1)\sqrt{C_2}$ (by Lemma 5.3). If $\hat{c}$ is the set of centers output by f', then $\mathrm{cost}_{x|_U}(\hat{c}) \leq A\,\mathrm{cost}_{x|_U}(c^*_x)$, since c^*_x costs as much or more than the cost of the optimal clustering for $x|_U$. On the other hand, $\mathrm{cost}_{x|_U}(c^*_x) \leq \mathrm{cost}_x(c^*_x) + \gamma$, and so

$$\mathrm{cost}_x(\hat{c}) \leq \mathrm{cost}_{x_U}(\hat{c}) + \gamma \leq A(\mathrm{cost}_x(c^*_x) + \gamma) + \gamma \leq A\phi^2\Delta^2_{k-1} + (A+1)\gamma$$

Since $\gamma \approx \Delta^2_{k-1}/(A+1)\sqrt{C_2}$, and $\phi^2 A < \phi^2$, we can choose the constant C_1 small enough and C_2 large enough so that $\mathrm{cost}_x(\hat{c}) \leq a\Delta^2_{k-1}$, as needed for part (b).

We now turn to showing that clusterings with low cost are close to the optimal. Every set of centers $\hat{c}_1, ..., \hat{c}_k$ defines a partition of the data set (or index set) based on the Voronoi cells of the centers in $\mathbb{R}^d$. We write $R(\hat{c}_i)$ to denote the corresponding partition of $[n]$ (this depends implicitly on all of $\hat{c}_1, ..., \hat{c}_k$). Let $R^*(i)$ denote the partition induced by the optimal cluster centers c^*x. We use the following result:

Lemma 5.4 ([25], Thm 5.1)**.** *If x is ϕ^2-separated and $cost_x(\hat{c}) \leq a\Delta_{k-1}^2$, then there is a permutation $\pi : [k] \to [k]$ such that for each i, the symmetric difference $R(\hat{c}_i) \triangle R^*(\pi(i))$ has size at most $28\phi^2 R^*(i)$.*

This lemma sets up a matching between centers in $\hat{c}$ and in c^*. Consider the distance $\|\hat{c}_i - c_i^*\|$. Since each is the average of a set of vectors of length at most Λ, we can bound the difference by $28\Lambda\phi^2(1 + \frac{1}{1-28\phi^2})$. Given our setting of ϕ^2, this is at most $60\Lambda\phi^2$. Finally, since each of the k centers may have this distance from its optimal counterpart, the Wasserstein distance previously defined is at most $60\Lambda\phi^2\sqrt{k}$. This completes the proof of Lemma 5.2.

Note that the proof of Lemma 5.4 in [25] proceeds by bounding the lengths of the vectors $\|\hat{c}_i - c_i^*\|$. It is likely that one could improve the constants in this proof by combining the two arguments. As it is, there are several possible settings of the constants C_1, C_2 that come out of the proof: one example is $C_1 = \frac{1}{401}$ and $C_2 \approx 10^{12}$; another possibility is $C_1 \approx 10^6$ and $C_2 \approx 10^6$. □

Remark 3. Meila [22] defined a different "well-clusteredness" condition, based on the difference between the semi-definite programming relaxation of k-SED and the integral optimum. That work also shows a stability condtion similar to Lemma 5.4. It is not clear to us how to compare the approaches of [22,25]. Yet another condition was proposed by Dasgupta and Schulman [10], based on a goodness of fit test for Gaussian mixtures. To our knowledge, however, no stability results have been proved based on that condition.

5.2 Learning Mixtures of Gaussians

Consider estimating and releasing the parameters of a uniform mixture of spherical Gaussian distributions. We say a density function h over $\mathbb{R}^\ell$ is a mixture of k spherical Gaussians if we can express it as a convex combination $h(x) = \sum_{i=1}^k \frac{1}{k} h_0(x - \mu_i)$ where h_0 is the density function of a spherical Gaussian distribution with variance σ^2 in each coordinate. The centers μ_i are the parameters of h (assuming σ^2 is known).

Given n i.i.d. samples $x_1 ..., x_n$ drawn according to h, our goal is to estimate and release the μ_i's as accurately as possible while protecting the privacy of $x = (x_1, ..., x_n)$. This is related to, but different from, the k-SED clustering problem (the optimal k-SED cluster centers form the maximum likelihood estimate for the μ_i's, but the optimal centers are hard to compute exactly and it is not clear that centers with nearly optimal cost will yield a good estimate of the μ_i's).

We apply the sample and aggregate framework. Since the points of x are drawn i.i.d., each of the subsets $x|_{U_1}, ..., x|_{U_m}$ will also consist of i.i.d. samples. Thus, it is sufficient to show that given n/m i.i.d. samples from h, we can compute a set of centers that is "close" to the real centers. As above, we use the Wasserstein distance (Sec. 4.1) to measure the distance between different sets of centers. To get a good estimate of the centers from each of the samples, we use a slight modification of the learning algorithm of Vempala and Wang [28], whose properties are summarized here:

Lemma 5.5 ([28])**.** *There exists an efficient algorithm which, given n' samples from a mixture of k spherical Gaussians, identifies the correct* partition *of the data points with high probability as long as the distance between any pair of centers is $\omega\left(\sigma k^{1/4} \log^{1/2}(\ell/w_{\min})\right)$, where $w_{\min}$ is the smallest weight in the mixture, and $n' = \frac{\ell^3}{w_{\min}} \operatorname{polylog}(\ell)$.*

For larger n' this lemma may not hold (since the probability of one point going far astray and landing in the wrong cluster eventually becomes quite high) but one can still get a guarantee on the how far the estimated centers of the Gaussians will be from the real ones. (The idea is to running the algorithm on different subsets of the data and combine them to get an estimate of the set of centers.)

Lemma 5.6. *A modification of the Vempala-Wang clustering algorithm [28], given n' samples from a mixture of k spherical Gaussians, outputs a set of centers within Wasserstein distance $O(\sigma k\sqrt{\frac{\ell}{n'}})$ of the real centers, with probability $1-o(1)$ as long as the distance between any pair of centers is $\omega\left(\sigma k^{1/4}\log^{1/2}(\ell k)\right)$ and $n' \geq \ell^3 k\,\mathrm{polylog}(\ell)$.*

When $n' = \omega\left(\frac{\sqrt{\ell}\log(1/\delta)}{\epsilon}\log(\frac{nk}{\sigma})\right)$, the sample and aggregate mechanism is (ϵ,δ)-differentially private; it releases a set of centers $\{\hat{\mu}_i\}$ where the expected error $\|\hat{\mu}_i - \mu_i\|$ in each of the estimates is $O\left(\frac{\sigma\ell^{3/2}k\log(1/\delta)}{\epsilon\sqrt{n'}}\right)$. For large n (polynomial in k and ℓ), we get an estimate h' of the mixture distribution, which converges to h (that is, the KL-divergence between the distributions tends to 0).

Acknowledgements

We are grateful to Cynthia Dwork, Piotr Indyk, Frank McSherry, Nina Mishra, Moni Naor, Gil Segev, and Enav Weinreb for discussions about various aspects of this work. We thank Sergey Orshanksiy for the $O(n\log n)$-time algorithm for computing the smooth sensitivity of the median. A significant part of this research was performed while the authors were visiting IPAM at UCLA. We thank Rafi Ostrovsky and the IPAM staff for making our stay there pleasant and productive.

References

[1] N. R. Adam and J. C. Wortmann. Security-control methods for statistical databases: a comparative study. *ACM Computing Surveys*, 25(4), 1989.

[2] Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: ranking and clustering. In *STOC 2005*, pages 684–693, 2005.

[3] Boaz Barak, Kamalika Chaudhuri, Cynthia Dwork, Satyen Kale, Frank McSherry, and Kunal Talwar. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In Leonid Libkin, editor, *PODS*, pages 273–282. ACM, 2007.

[4] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: The SuLQ framework. In *PODS*, 2005.

[5] Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. In *Symposium on the Theory of Computing (STOC)*, 2008.

[6] Kamalika Chaudhuri and Nina Mishra. When random sampling preserves privacy. In Cynthia Dwork, editor, *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 198–213. Springer, 2006.

[7] Shuchi Chawla, Cynthia Dwork, Frank McSherry, Adam Smith, and Hoeteck Wee. Toward privacy in public databases. In Joe Kilian, editor, *Theory of Cryptography Conference (TCC)*, volume 3378 of *Lecture Notes in Computer Science*, pages 363–385. Springer, 2005.

[8] Shuchi Chawla, Cynthia Dwork, Frank McSherry, and Kunal Talwar. On the utility of privacy-preserving histograms. In *21st Conference on Uncertainty in Artificial Intelligence (UAI)*, 2005.

[9] Chris Clifton, Murat Kantarcioglu, Jaideep Vaidya, Xiaodong Lin, and Michael Y. Zhu. Tools for privacy preserving data mining. *SIGKDD Explorations*, 4(2):28–34, 2002.

[10] Sanjoy Dasgupta and Leonard J. Schulman. A two-round variant of EM for gaussian mixtures. In Craig Boutilier and Moisés Goldszmidt, editors, *UAI*, pages 152–159. Morgan Kaufmann, 2000.

[11] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *PODS*, pages 202–210, 2003.

[12] Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2006.

[13] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, pages 486–503, 2006.

[14] Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In Michael Mitzenmacher, editor, *STOC*, pages 371–380. ACM, 2009.

[15] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2006.

[16] Cynthia Dwork and Kobbi Nissim. Privacy-preserving datamining on vertically partitioned databases. In *CRYPTO*, pages 528–544, 2004.

[17] Alexandre V. Evfimievski, Johannes Gehrke, and Ramakrishnan Srikant. Limiting privacy breaches in privacy preserving data mining. In *PODS*, pages 211–222, 2003.

[18] Johannes Gehrke. Models and methods for privacy-preserving data publishing and analysis (tutorial slides). In *Twelfth Annual SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD 2006)*, 2006.

[19] J. Kleinberg and E. Tardos. *Algorithm Design*. Addison-Wesley, 2005.

[20] Frank McSherry and Ilya Mironov. Differentially private recommender systems: Building privacy into the netflix prize contenders. In John F. Elder IV, Françoise Fogelman-Soulié, Peter A. Flach, and Mohammed Javeed Zaki, editors, *KDD*, pages 627–636. ACM, 2009.

[21] Frank McSherry and Kunal Talwar. Differential privacy in mechanism design. In A. Sinclair, editor, *IEEE Symposium on the Foundations of Computer Science (FOCS)*, October 2007.

[22] Marina Meila. The uniqueness of a good optimum for k-means. In William W. Cohen and Andrew Moore, editors, *ICML*, pages 625–632. ACM, 2006.

[23] Ramgopal R. Mettu and C. Greg Plaxton. Optimal time bounds for approximate clustering. *Machine Learning*, 56(1-3):35–60, 2004.

[24] Nina Mishra, Daniel Oblinger, and Leonard Pitt. Sublinear time approximate clustering. In *SODA*, pages 439–447, 2001.

[25] R. Ostrovsky, Y. Rabani, L. Schulman, and C. Swamy. The effectiveness of Lloyd-type methods for the k-means problem. In *47th IEEE Symposium on the Foundations of Computer Science (FOCS)*, 2006.

[26] A.B. Slavkovic. *Statistical Disclosure Limitation Beyond the Margins: Characterization of Joint Distributions for Contingency Tables.* Ph.D. Thesis, Department of Statistics, Carnegie Mellon University, 2004.

[27] Latanya Sweeney. Privacy-enhanced linking. *SIGKDD Explorations*, 7(2):72–75, 2005.

[28] Santosh Vempala and Grant Wang. A spectral algorithm for learning mixture models. *J. Comput. Syst. Sci.*, 68(4):841–860, 2004.

[29] Van H. Vu. On the concentration of multi-variate polynomials with small expectation. *Random Structures and Algorithms*, 16(4):344–363, 2000.

[30] L. N. Wasserstein. Markov processes over denumerable products of spaces describing large systems of automata. *Probl. Inform. Transmission*, 5:47–52, 1969.

A Useful Concentration Bounds

We collect here some of the concentration bounds used in the paper. We include proofs for completenes.

Fact A.1. *a) If Y is a sum of d exponential random variables, each with expectation 1, then the probability that Y exceeds $y = d + rd$ is at most $\exp(-d(r - \ln(1 + r)))$. This is at most $\exp(-d(r - 1)/2) = \exp(-(y - 2d)/2)$.*

b) If If Y is a sum of d chi-square random variables, each with 1 degree of freedom, then the probability that Y exceeds $y = d + rd$ is at most $\exp(-d(r - \ln(1 + r))/2)$. This is at most $\exp(-d(r - 1)/4) = \exp(-(y - 2d)/4)$.

Proof of Fact A.1. a) The moment generating function of Y is $E(e^{tY}) = (1 - t)^{-d}$. For any $t \in (0, 1)$ and $y > 0$, the probability that Y exceeds y is at most $\frac{(1-t)^d}{e^{ty}}$ (by Markov's bound). Setting $t = 1 - y/d$, we get that the probability is at most $(\frac{y}{d})^d e^{-(y-d)}$. For $y = d(1 + r)$, this reduces to $\exp(-d(r - \ln(1 + r)))$.

b) The moment generating function of Y is $E(e^{tY}) = (1 - 2t)^{-d/2}$. For any $t \in (0, 1)$ and $y > 0$, the probability that Y exceeds y is at most $\frac{(1-2t)^{d/2}}{e^{ty}}$ (by Markov's bound). Setting $t = \frac{1}{2}(1 - \frac{d}{y})$, we get that the probability is at most $(\frac{y}{d})^{d/2} e^{-(y-d)/2}$. For $y = d(1 + r)$, this reduces to $\exp(-d(r - \ln(1 + r))/2)$.

For both parts (a) and (b), the final bound is obtained by noting that $r - \ln(1 + r)$ is always greater than $\frac{r-1}{2}$. □

Fact A.2. *If $Y \sim N(0, 1)$, then for $\delta' < 1/e$, we have $\Pr(Y \leq \sqrt{2\ln(1/\delta')})$ is at most δ'.*

Proof. For any $y > 0$, we have $\Pr(N \geq y) = \int_{x \geq y} \frac{1}{\sqrt{2\pi}} e^{-x^2} < \frac{1}{y} \int_{x \geq y} \frac{1}{\sqrt{2\pi}} x e^{-x^2}$. This latter integral can be evaluated exactly; we obtain $\Pr(N \geq y) = \frac{1}{y\sqrt{2\pi}} e^{-y^2/2}$. Now set $y = \sqrt{2\ln(1/\delta')}$. Note that $y > 1$, so can bound the probability above by $\frac{e^{-y^2/2}}{\sqrt{2\pi}} < \delta'/2$. Similarly, the probability that $N < -y$ is at most $\delta'/2$. The union of the two events has probability at most δ'. □