Smooth Sensitivity and Sampling in Private Data Analysis*

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

Sofya Raskhodnikova
Dept. of Computer Science and Engineering
Pennsylvania State University
sofya@cse.psu.edu

Adam Smith
Dept. of Computer Science and Engineering
Pennsylvania State University
asmith@cse.psu.edu

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-based 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-based 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.

*Part of the work 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.

†Research partially supported by the Israel Science Foundation (grant No. 860/06), and by the Frankel Center for Computer Science.

STOC'07, June 11–13, 2007, San Diego, California, USA.

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

General Terms: Algorithms, Security, Theory.

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

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 [16, 1, 6, 13, 17]). 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 [7, 12, 11, 3, 4, 5, 8] presents precise mathematical definitions of data privacy, that give meaningful guarantees in the presence of a strong, realistic adversary. To provide protocols satisfying the definitions, these works 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. Blum et al. [3] and Dwork et al. [10] 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-Based Noise. We introduce a new generic framework for private data analysis. Our framework allows one to release functions f of the data with instance-based 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-based 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, from leaking information.

Our framework generalizes the results of [3, 10]. 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 can release $f(x)$ with less noise than in [3, 10]. 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 show how to compute or approximate the smooth sensitivity and apply our framework for a variety of such functions. One example is the median, for which we give very efficient algorithms for the smooth sensitivity. The second example is the cost of the minimum spanning tree of a graph where individuals correspond to edges. Another example, the number of triangles in a graph, will appear in the full version.

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 [10] 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. All missing proofs and details, as well as additional examples, are deferred to the full version.

1.1 Privacy Definition

As in [12, 7, 11, 3, 10, 8], we consider a client-server setup, where clients' queries are answered by a trusted database access mechanism at the server. 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 sends back $f(x)$ after perturbing the value somewhat, for example by adding random noise. Our goal is to minimize the noise (and thus enable the user to learn f as accurately as possible) while preserving the privacy of individual contributors.

We use the privacy definition from [10, 9], which limits the *incremental* information a user might learn in addition to what he knew before seeing the released statistics. For a particular query f and a database x, the randomized access mechanism $\mathcal{A}$ defines a distribution on the outputs, denoted by $\mathcal{A}(x)$.

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., $d(x, y) = 1$. A *negligible function* $\delta()$ is a positive function that is asymptotically smaller than any inverse polynomial: $\delta(n) = 1/n^{\omega(1)}$.

A mechanism $\mathcal{A}$ is private if neighbor databases induce nearby distributions on the outputs. Our formulation follows [9].

DEFINITION 1.1 (INDISTINGUISHABILITY [10, 9]). *Let $\delta = \delta(n)$ be a negligible function of n. A randomized algorithm $\mathcal{A}$ is (ϵ, δ)-indistinguishable 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 $\delta = 0$, we say the algorithm is ϵ-indistinguishable.

Definition 1.1 states that no individual has a pronounced effect on the statistics published by the server, in the sense that the output distribution is almost the same whether the individual supplies his actual data or something irrelevant.

Indistinguishability 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 protocol consisting of t rounds, each of which is individually ϵ-indistinguishable, is itself $t\epsilon$-indistinguishable. We therefore focus on constructing 1-round protocols.

1.2 Calibrating Noise to Sensitivity

Recall that most works in private data analysis [7, 11, 3, 10] 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 $A(x) = f(x) + Y$ for some random

variable Y. 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 only for ease of presentation as the following analysis may be generalized to any metric space.

DEFINITION 1.2 (GLOBAL SENSITIVITY [10]). *For $f : D^n \to \mathbb{R}^d$, the global sensitivity of f is*

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

CLAIM 1.3 ([10]). *For all $f : D^n \to \mathbb{R}^d$, the database access mechanism $\mathcal{A}_f(x) = f(x)+(Y_1, \ldots, Y_d)$, where the Z_i are drawn i.i.d. from $\mathrm{Lap}(GS_f/\epsilon)$, is ϵ-indistinguishable*[1].

Claim 1.3 yields two generic approaches to constructing database access mechanisms for functions f. The first approach [10] is to show that GS_f is low, and hence $f(x)$ can be released with noise $Z \sim \mathrm{Lap}(GS_f/\epsilon)$. The second approach [3] is to express f in terms of functions $g_1, g_2, \ldots$ with low global sensitivity, and analyze how noisy answers to $g_1, g_2, \ldots$ interfere with the computation of f. [2]

These approaches are productive for many functions, such as sum queries [7, 11]; Principle Component Analysis, the Perceptron algorithm, k-means, learning ID3 decision trees, statistical learning [3]; histograms, Singular Value Decomposition, distance to a property, and functions that can be estimated on all x using small random samples from x [10].

1.3 Smooth Sensitivity

In the global sensitivity framework of [10] 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 1.4 (LOCAL SENSITIVITY). *For $f : D^n \to \mathbb{R}^d$ and $x \in D^n$, the local sensitivity of f at x is*

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

Observe that the global sensitivity from Definition 1.2 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 [19]. The current work is the first to use it in the context of private data analysis.

Suppose we release $f_{med}(x)$ with noise magnitude proportional to $LS_{f_{med}}(x)$, instead of $GS_{f_{med}}$. That would allow us to add significantly less noise for typical inputs. However, the resulting scheme is too naïve, and would not satisfy Definition 1.1 as the the noise magnitude itself reveals information about the database (see Section 2.2).

Our goal is hence to add instance-based 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 to 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. [10, 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 [10] 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.

[1] The probability density function of the Laplace distribution $\mathrm{Lap}(\lambda)$ is $h(y) = \frac{1}{2\lambda} e^{-|y|/\lambda}$. It has zero mean, and standard deviation $\sqrt{2}\lambda$.

[2] For simplicity we omit from our discussion the noise magnitude dependency on number of queries. See [7, 11, 3].

Our mechanism releases accurate answers on interesting 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. [15]. This definition implies that all near-optimal clusterings of x induce similar partitions of the points of x. [15] 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. [3] 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. [3] 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-BASED 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, [10] gave ϵ-indistinguishable protocols where the noise magnitude is proportional to global sensitivity (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 Bounds 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 [10].

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 whereas the probability of receiving a non-zero answer when $x_1 = \cdots = x_m = 0, x_{m+1} = \cdots = x_n = \Lambda$ is non-negligible. Thus, the protocol is not (ϵ, δ)-indistinguishable for any negligible δ. $\diamondsuit$

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 ϵ-smooth if $GS_{\ln(S(\cdot))} \leq \epsilon$.

DEFINITION 2.1 (A SMOOTH BOUND). *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:*

$$\forall x \in D^n : \quad S(x) \geq LS_f(x) \ ;$$
$$\forall x, y \in D^n, d(x,y) = 1 : \quad S(x) \leq e^{\beta} S(y) \ .$$

An example of a function that satisfies Definition 2.1 is the smooth sensitivity of f:

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

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

The smooth sensitivity $S^*_{f,\beta}$ is the *smallest* function to satisfy Definition 2.1:

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.*

Note that the constant function $S(x) = GS_f$ also meets the requirements of Definition 2.1, though in general it is a very conservative upper bound on LS_f.

2.2 Calibrating Noise to Smooth Bounds

We now show that adding noise proportional to a smooth upper bound on the local sensitivity yields a private output perturbation mechanism. We add noise proportional to $S_f(x)/\alpha$, 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 taking values in $\mathbb{R}^d$, the smoothing parameter β is $\propto \epsilon/d$ or $\propto \epsilon/\sqrt{d}$, depending on the exact choice of the noise distribution.

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

DEFINITION 2.4 (ADMISSIBLE NOISE DISTRIBUTION). *A probability distribution h on $\mathbb{R}^d$ is* (α, β)-admissible *if, for $\alpha = \alpha(\epsilon, \delta), \beta = \beta(\epsilon, \delta)$, the following two conditions hold for all $\|\Delta\| \leq \alpha$ and $|\lambda| \leq \beta$, and for all subsets $\mathcal{S} \subseteq \mathbb{R}^d$:*

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

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

The definition requires the noise distribution to not change much under translation (sliding) and scaling (dilation). A distribution satisfying the two properties can be used to add noise proportional to $S(x)$:

LEMMA 2.5. *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 the database access mechanism $\mathcal{A}(x) = f(x) + \frac{S(x)}{\alpha} \cdot Z$ is (ϵ, δ)-indistinguishable.*

On 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.

EXAMPLE 2. Let $h(z) \propto 1/(1 + |z|^\gamma)$ for $\gamma > 1$. These $h(x)$ are $(\epsilon/4\gamma, \epsilon/\gamma)$-admissible, and yield $\delta = 0$. This is a collection of heavy tail distributions, asymptotically decreasing $\propto 1/|z|^\gamma$. For $\gamma > 3$ they have well-defined expectations and variances. In dimension $d > 1$, one can use a product of these distributions; the result is $(\epsilon/4\gamma, \epsilon/d\gamma)$- admissible.

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 that are not heavy tail (such as Gaussian and Laplace), we take $\delta > 0$, and require the sliding and dilation properties to hold with high probability.

EXAMPLE 3. The Laplace distribution, $h(z) = \frac{1}{2} \cdot e^{-|z|}$, is (α, β)-admissible with $\alpha = \epsilon/2, \beta = \epsilon/2 \ln(1/\delta)$. In dimension $d > 1$, one can use the product of Laplace distributions, with $\beta = \Omega(\epsilon/\sqrt{d} \ln(1/\delta))$.

The Gaussian distribution, $h(z) = \frac{1}{2\pi} \cdot e^{-z^2/2}$, is (α, β)-admissible with $\alpha = \epsilon/\sqrt{\ln(1/\delta)}, \beta = \epsilon/2 \ln(1/\delta)$. In dimension $d > 1$, we get $\beta = \Omega(\epsilon/\sqrt{d \ln(1/\delta)})$. ♢

3. COMPUTING SMOOTH SENSITIVITY

In this section we show how to compute smooth sensitivity $S^*_{f,\epsilon}(x)$, proposed in Definition 2.2, for two specific functions, median and the cost of the minimum spanning tree.

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*

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

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

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

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 S^*. The next claim provides a general method for giving smooth upper bounds on local sensitivity.

CLAIM 3.2. *Let $\tilde{S}_{f,\epsilon}(x) = \max_{k=0,\ldots,n}(U_k(x) \cdot e^{-\epsilon k})$ where (1) $LS(x) \le U_0(x)$ and (2) for x, y such that $d(x, y) = 1$, $U_k(x) \le U_{k+1}(y)$. For a given value $k_0(n)$, let $\hat{S}_{f,\epsilon}(x) = \max(GS_f \cdot e^{-\epsilon k_0}, \max_{k=0,\ldots,k_0-1} e^{-\epsilon k} \cdot A^{(k)}(x))$. Then $\tilde{S}_{f,\epsilon}(x)$ and $\hat{S}_{f,\epsilon}(x)$ are ϵ-smooth upper bounds on local sensitivity.*

3.1 Median

Let f_{med} be as in Example 1 and assume the database elements are in nondecreasing order. Recall 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 $i \le 0$ and $x_i = \Lambda$ for $i > n$.

CLAIM 3.3. *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 $O(n^2)$.

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 $i \ge m$ are set to Λ. Thus,

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

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 $O(n^2)$. □

For example, consider the instance where $x_i = (\Lambda i)/n$ for all $i \in [n]$. In this case, $A^{(k)}(x) = (k+1)\Lambda/n$, and $e^{-\epsilon} A^{(k)}(x)$ is maximized when $k = 1/\epsilon$. Then $S^* \le \Lambda/(\epsilon n)$, and consequently, the magnitude of the noise we add is $\Lambda/(\epsilon^2 n)$. For comparison, the noise magnitude for f_{med} in the global sensitivity framework of [10] is Λ/ϵ, high enough to destroy all information.

Claim 3.2 leads to approximation algorithms for $S^*_{f_{med},\epsilon}$:

CLAIM 3.4. *There is a smooth upper bound on $LS_{f_{med}}$ that is a factor 2 approximation to $S^*_{f_{med},\epsilon}$ and is computable in time $O(n)$. There is smooth upper bound on $LS_{f_{med}}$ that approximates $S^*_{f_{med},\epsilon}$ within error $\frac{\Lambda}{\text{poly}(n)}$ and is computable in time $O(\frac{\log^2 n}{\epsilon^2})$.*

3.2 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. 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. Here we show how to compute the smooth sensitivity of f_{MST} in polynomial time.

The main idea in the analysis is to expresses 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. 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) is in 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. Let $\ell(S) = |\{(i,j) \mid i \in S, j \in V/S\}|$ denote the number of edges crossing the cut.

LEMMA 3.5. *The local sensitivity of f_{MST} at distance k is*
$A^{(k)}_{f_{MST}}(G) = \max_{S \subset V} A^{(k)}_{f_{\min}}\left(w_1(S), w_2(S), \ldots, w_{\ell(S)}(S)\right).$

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 3.5 and exchange the order of the maximums to get

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

LEMMA 3.6. *The smooth sensitivity of f_{MST} can be computed in time polynomial in the number of edges.*

Our algorithm for computing $S^*_{f_{\text{MST}},\epsilon}(G)$ is based on repeated calls to min-cut on graphs related to G. Here we only present the main idea on computing $\max_S w_k(S)$, the first term in the expression for $A^{(k)}_{f_{\text{MST}}}(G)$ in Equation 1.

To compute $\max_S w_k(S)$, we use a procedure that computes the minimum cut of an undirected unweighted graph. Let $E_w = \{e \in E \mid w(e) \leq w\}$. Let G_w be the graph (V, E_w) with *no weights*. Let $c_w = \text{cost}(\text{min-cut}(G_w))$, i.e., the number of edges in the minimum cut of G_w. Let $0 \leq wt_1 < wt_2 < \ldots < wt_t \leq \Lambda$ be the sorted list of distinct weight values in G. Then $t \leq |E| < n^2$. We can do a binary search on the list of weights to find i such that $c_{wt_i} \geq k$ and $c_{wt_{i-1}} < k$. Then $\max_{S \subset V} w_k(S) = wt_i$. To summarize, $max_{S \subset V} w_k(S)$ can be computed with $O(\log t)$ min-cut computations on G_{wt_i}'s.

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

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* [20]. 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:

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

It is easy to verify that this defines a metric on unordered sets of points in $\mathbb{R}^\ell$. Computing distances is efficient (one can reduce it to computing a maximum matching in a bipartite graph).

We do not know how to add noise with respect to the Wasserstein distance. For the purposes of adding noise, we view $\mathcal{M}$ as L_2^d, $d = \ell k$. This works because L_2 distance is an upper bound on the Wasserstein distance. To summarize, we compute sensitivity with respect to Wasserstein, but add noise with respect to L_2.

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. 1, 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.

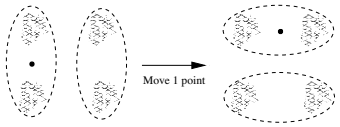

Figure 1: A sensitive 2-SED instance

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 2).

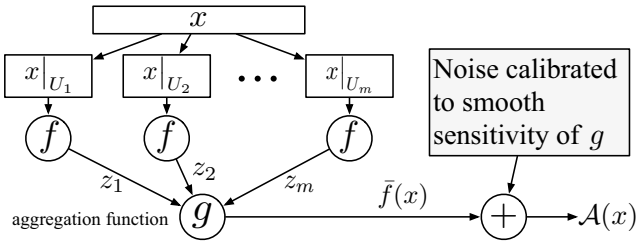

Figure 2: The Sample-Aggregate Framework

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 2.4). 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, and are deferred to the full version of the paper. Further below, we develop machinery for general metrics that, combined with results on adding noise from Section 2.2, 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-indistinguishable 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.*

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 Aggregation 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*

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

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

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

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 4. 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, $S(z) \le 2r + \Lambda \cdot e^{-\frac{\beta m}{6s}}$ 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

To prove Theorem 4.9, 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. [4]. We are not aware of a direct connection between the two techniques.

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. 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

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

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,

$$r^{(z)}(t) \le r^{(z')}(t + s) . \tag{2}$$

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. (2), this is bounded above by $r^{(z)}(t_0) + r^{(z)}(t_0 + s) \le 2r^{(z)}(t_0 + s)$. This yields:

$$LS^{(s)}_{g_0}(z) \le 2r^{(z)}(t_0 + s).$$

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. (2):

$$S_0(z) \le 2 \max_{k \ge 0} \left(r^{(z')} \left(t_0 + (k+2)s\right) e^{-\beta k} \right)$$
$$= 2(e^{\beta}) \max_{k' \ge 1} \left(r^{(z')} \left(t_0 + (k'+1)s\right) e^{-\beta k'} \right) = e^{\beta} S_0(z').$$

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 $t \le \frac{2m}{3}$.

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, $S_0(z) \le 2 \max \left(r \ , \ \Lambda \cdot e^{-\frac{\beta m}{6s}} \right)$, which satisfies the requirements of a good aggregation. □

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, $LS^{(s)}_g \le 2r_1(t_0 + s)$. 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 2.2, implies the main Theorem 4.2 and its analogues for other metric spaces.

5. APPLYING SAMPLE-AGGREGATE

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 [15] 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. [15] 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, [15]). *Given a dataset x, let $\Delta^2_k(x)$ be the cost of the optimal k-SED clustering of x. We say x is* ϕ^2-separated *if $\Delta^2_k \le \phi^2 \Delta^2_{k-1}$.*

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., [14]). 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 $d_{\mathsf{W}}\left(f(x|_U),\ f_{\text{cc}}(x)\right)$ 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 $n' > C_2(\frac{\Lambda^2}{\Delta^2_{k-1}})^2 A^2 k\ell \ln\left(\frac{\ell \Lambda^2}{\Delta^2_{k-1}}\right)$ and $\phi^2 < \frac{C_1}{A+1}$, then*

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

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.

5.2 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 [18], whose properties are summarized here:

LEMMA 5.3. *A modification of the Vempala-Wang clustering algorithm [18], 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 $n' \geq \ell^3 k \operatorname{polylog}(\ell)$.*

When $n' = \omega\left(\frac{\sqrt{\ell}\log(1/\delta)}{\epsilon} \log(\frac{nk}{\sigma})\right)$, the sample and aggregate mechanism is (ϵ, δ)-indistinguishable; 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). Details are deferred to the full version.

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. A significant part of this research was performed while the authors were visiting UCLA's IPAM. We thank Rafi Ostrovsky and the IPAM staff for making our stay there pleasant and productive.

6. 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] N. Ailon, M. Charikar, and A. Newman. Aggregating inconsistent information: ranking and clustering. In *STOC 2005*, pages 684–693, 2005.

[3] A. Blum, C. Dwork, F. McSherry, and K. Nissim. Practical privacy: The SuLQ framework. In *PODS*, 2005.

[4] S. Chawla, C. Dwork, F. McSherry, A. Smith, and H. Wee. Toward privacy in public databases. In *Theory of Cryptography Conference (TCC)*, pages 363–385, 2005.

[5] S. Chawla, C. Dwork, F. McSherry, and K. Talwar. On the utility of privacy-preserving histograms. In *21st Conference on Uncertainty in Artificial Intelligence (UAI)*, 2005.

[6] C. Clifton, M. Kantarcioglu, J. Vaidya, X. Lin, and M. Y. Zhu. Tools for privacy preserving data mining. *SIGKDD Explorations*, 4(2):28–34, 2002.

[7] I. Dinur and K. Nissim. Revealing information while preserving privacy. In *PODS*, pages 202–210, 2003.

[8] C. Dwork. Differential privacy. In *ICALP*, pages 1–12, 2006.

[9] C. Dwork, K. Kenthapadi, F. McSherry, I. Mironov, and M. Naor. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, pages 486–503, 2006.

[10] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, pages 265–284, 2006.

[11] C. Dwork and K. Nissim. Privacy-preserving datamining on vertically partitioned databases. In *CRYPTO*, pages 528–544, 2004.

[12] A. V. Evfimievski, J. Gehrke, and R. Srikant. Limiting privacy breaches in privacy preserving data mining. In *PODS*, pages 211–222, 2003.

[13] J. 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.

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

[15] 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.

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

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

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

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

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