Private Learning and Sanitization: Pure vs. Approximate Differential Privacy[⋆]

Amos Beimel[1], Kobbi Nissim[2], and Uri Stemmer[1]

[1] Dept. of Computer Science Ben-Gurion University
[2] Dept. of Computer Science Ben-Gurion University & Harvard University
`{beimel,kobbi,stemmer}@cs.bgu.ac.il`

Abstract. We compare the sample complexity of private learning and sanitization tasks under *pure* ϵ-differential privacy [Dwork, McSherry, Nissim, and Smith TCC 2006] and *approximate* (ϵ, δ)-differential privacy [Dwork, Kenthapadi, McSherry, Mironov, and Naor EUROCRYPT 2006]. We show that the sample complexity of these tasks under approximate differential privacy can be significantly lower than that under pure differential privacy.

Keywords: Differential Privacy, Private Learning, Sanitization.

1 Introduction

Learning and sanitization are often applied to collections of sensitive data of individuals and it is important to protect the privacy of these individuals. We examine the sample complexity of these tasks while preserving differential privacy [9]. Our main focus is on private learning [14] and sanitization [4] and we show starking differences between the required sample complexity for these tasks under ϵ-differential privacy [9] (also called *pure* differential privacy) and its variant (ϵ, δ)-differential privacy [7] (also called *approximate* differential privacy).

An algorithm A satisfies the requirement of *Pure Differential Privacy* if for every two databases that differ on exactly one entry, and for every event defined over the output of the algorithm, the probability of this event is close up to a multiplicative factor of $\approx 1 + \epsilon$ whether A is applied on one database or on the other. Informally, to satisfy the requirement of *Approximate Differential Privacy* the above guarantees needs to be satisfied only for events whose probability is at least $\approx \delta$. We show that even negligible $\delta > 0$ can have a significant effect on sample complexity of two fundamental tasks: private learning and sanitization.

[⋆] Research supported by the Israel Science Foundation (grants No. 938/09 and 2761/12) and by the Frankel Center for Computer Science at Ben-Gurion University. Work done while the second author was a Visiting Scholar at the Harvard Center for Research on Computation and Society (CRCS). Work partially done when the third author was visiting Harvard University supported in part by NSF grant CNS-1237235 and a gift from Google, Inc.

© Springer-Verlag Berlin Heidelberg 2013

Private Learning. Private learning was introduced in [14] as a combination of Valiant's PAC learning model [21] and differential privacy (applied to the examples used by the learner). The work on private learning has since mainly focused on pure privacy. On the one hand, this work showed, via generic constructions [3,14], that every finite concept $\mathcal{C}$ class can be learned privately, using sample complexity proportional to poly$(\log |\mathcal{C}|)$ (often efficiently). On the other hand, a significant difference was shown between the sample complexity of *traditional* (non-private) learners (crystallized in terms of $\mathrm{VC}(\mathcal{C})$ and smaller than $\log |\mathcal{C}|$ in many interesting cases) and private learners, when the latter are required to be *proper* (i.e., output a hypothesis in $\mathcal{C}$).

As an example, let $\texttt{POINT}_d$ be the class of point functions over the domain $\{0,1\}^d$ (these are the functions that evaluate to one on exactly one point of the domain). Consider the task of *properly* learning $\texttt{POINT}_d$ where, after consulting its sample, the learner outputs a hypothesis that is by itself in $\texttt{POINT}_d$. Non-privately, learning $\texttt{POINT}_d$ requires merely a constant number of examples (as $\mathrm{VC}(\texttt{POINT}_d) = 1$). Privately, $\Omega(d)$ examples are required [1]. Curiously, the picture changes when the private learner is allowed to output a hypothesis not in $\texttt{POINT}_d$ (such learners are called *improper*), as the sample complexity can be reduced to $O(1)$ [1]. This, however, comes with a price, as it was shown [1] that such learners must return hypotheses that evaluate to one on exponentially many points in $\{0,1\}^d$ and, hence, are very far from all functions in $\texttt{POINT}_d$. A similar lower bound of $\Omega(d)$ samples is known also for properly and privately leaning the class $\texttt{INTERVAL}_d$ of threshold functions over the interval $[0, 2^d - 1]$ (no corresponding sample-efficient improper private learner is known, that is, the best previously known private learning algorithm (proper or improper) for $\texttt{INTERVAL}_d$ has sample complexity $O(d)$). A complete characterization for the sample complexity of pure private learners was recently given in [2], in terms of a new dimension – the *Representation Dimension*, that is, given a class C the number of samples needed and sufficient for privately learning C is $\mathrm{RepDim}(C)$.

We show that the sample complexity of proper private learning with approximate differential privacy can be significantly lower than that with pure differential privacy. Our starting point for this work is an observation that with *approximate* $(\epsilon, \delta{>}0)$-differential privacy, sample complexity of $O(\log(1/\delta))$ suffices for learning points *properly*. This gives a separation between pure and approximate proper private learning for $\delta = 2^{-o(d)}$. Anecdotally, combining this with a result from [1] gives a learning task that is not computationally feasible under pure differential privacy and polynomial time computable under approximate differential privacy.

Sanitization. The notion of differentially private sanitization was introduced in the seminal work of Blum et al. [4]. A sanitizer for a class of predicates $\mathcal{C}$ is a differentially private mechanism translating an input database S to an output database $\hat{S}$ s.t. $S, \hat{S}$ agree (approximately) on the fraction of the entries in S satisfying φ for all $\varphi \in \mathcal{C}$. Blum et al. gave a generic construction of pure differentially private sanitizers exhibiting sample complexity $O(\mathrm{VC}(C) \log |X|)$, and lower bounds partially supporting this sample complexity were given by [17,1,13] (the construction is not generally feasible [10,20,19]). As with private learning, we show significant differences between the sample complexity required for sanitization of simple predicate classes under pure and approximate differential privacy.

1.1 Our Contributions

To simplify the exposition, we omit in this section dependency on all variables except for the complexity variable d corresponding, e.g., to domain size and (logarithm of) concept class size.

Tools. A recent instantiation of the Propose-Test-Release framework [8] by Smith and Thakurta [18] results, almost immediately, with a proper learner for points, exhibiting $O(1)$ sample complexity while preserving approximate differential privacy. This simple technique does not suffice for our other constructions of learners and sanitizers, and we, hence, introduce new tools for coping with proper private learning of intervals, sanitization for point functions, and sanitization for intervals:

- **Choosing mechanism:** Given a *low-sensitivity* quality function, one can use the exponential mechanism [16] to choose an approximate solution. This method requires, in general, a database of size logarithmic in the number of possible solutions. We identify a sub family of low-sensitivity functions, called *bounded-growth* functions, for which it is possible to significantly reduce the necessary database size when using the exponential mechanism.
- **Recursive algorithm for concave promise problems:** We define a family of optimization problems, which we call *Concave Promise Problems.* The possible solutions are ordered, and *concavity* means that if two solutions $f \leq h$ have quality $\geq \mathcal{X}$, then any solution $f \leq g \leq h$ also has quality $\geq \mathcal{X}$. The optimization goal is, when there exists a solution with a promised quality $\geq r$, to find a solution with quality $\approx r$. We observe that a concave promise problem can be privately approximated using a solution to a smaller instance of a concave promise problem. This allows us to construct an efficient recursive algorithm solving such problems privately. We show that the task of learning $\texttt{INTERVAL}_d$ is, in fact, a concave promise problem, and can be privately solved using our algorithm with sample size roughly $2^{\log^* d}$. Sanitization for $\texttt{INTERVAL}_d$ does not exactly fit the model of concave promise problems but can still be solved by iteratively defining and solving a small number of concave promise problems.

Implications for Private Learning and Sanitization. We give new *proper* private learning algorithms for the classes $\texttt{POINT}_d$ and $\texttt{INTERVAL}_d$. Both algorithms exhibit sample complexity that is significantly lower than bounds given in prior work, separating pure and approximate proper private learning. Similarly, we construct sanitizers for these two classes, again with sample complexity that is significantly lower than bounds given in prior work, separating sanitization in the pure and approximate privacy cases.

Santization vs. Private Learning. In [11], a reduction is given in both directions between agnostic learning of a concept class C, and the sanitization task for the same class C. They consider learners and sanitizers with limited access to the data, using statistical queries [15] (a non-private SQ learner could be transformed into a private learner, as was shown in [14]). In Section 5 we show a different (and natural) reduction from the task of privately learning a concept class C to the sanitization task of a slightly different concept class C', where the sanitizer's access to the database is unrestricted. We then exploit lower bounds on the sample complexity of private learners and show a class of predicates $\mathcal{C}$ over a domain X for which every private sanitizer requires databases of size $\Omega(\mathrm{VC}(\mathcal{C}) \log |X|)$. A similar lower bound was already shown by Hardt and Rothblum [13], achieving better results in terms of the approximation parameter. Their work ensures the existence of such a concept class, but does not give an explicit one.

Label Privacy. In Section 6 we examine private learning under a relaxation of differential privacy called *label privacy* (see [5] and references therein) where the learner is required to only protect the privacy of the labels in the sample. Chaudhuri et al. [5] gave lower bounds for label-private learners in terms of the doubling dimension of the target concept class. We show that the VC dimension completely characterizes the sample complexity of such learners.

1.2 Related Work

Mostly related to our work is the work on private learning and its sample complexity [14,3,1,5] and the early work on sanitization [4] mentioned above. Another related work is the work of De [6] who proved a separation between *pure* ϵ-differential privacy and *approximate* $(\epsilon, \delta{>}0)$-differential privacy. Specifically, he demonstrated that there exists a query where it is sufficient to add noise $O(\sqrt{n \log(1/\delta)})$ when $\delta > 0$ and $\Omega(n)$ noise is required when $\delta = 0$. Earlier work by Hardt and Talwar separated pure from approximate differential privacy for $\delta = n^{-O(1)}$ [12].

2 Preliminaries

Notation. We use $O_\gamma(g(d))$ as a shorthand for $O(h(\gamma) \cdot g(d))$ for some non-negative function h. We use X to denote an arbitrary domain, and X_d for the domain $\{0,1\}^d$. Databases $S_1 \in X^m$ and $S_2 \in X^m$ over a domain X are called *neighboring* if they differ in exactly one entry.

2.1 Differential Privacy

Definition 1 ([9,7]). *A randomized algorithm A is (ϵ, δ)-differentially private if for all neighboring databases S_1, S_2, and for all sets $\mathcal{F}$ of outputs,* $\Pr_A[A(S_1) \in \mathcal{F}] \leq \exp(\epsilon) \cdot \Pr_A[A(S_2) \in \mathcal{F}] + \delta$. *When $\delta = 0$ we omit it and say that A preserves ϵ-differential privacy.*

We use the term *pure* differential privacy when $\delta = 0$ and the term *approximate* differential privacy when $\delta > 0$.

2.2 PAC Learning and Private PAC Learning

A concept $c : X \to \{0,1\}$ is a predicate that labels *examples* taken from the domain X by either 0 or 1. A *concept class* $\mathcal{C}$ over X is a set of concepts mapping X to $\{0,1\}$. A learning algorithm is given examples sampled according to an unknown probability distribution $\mathcal{D}$ over X, and labeled according to an unknown *target* concept $c \in \mathcal{C}$. The goal of the learning algorithm is to output a hypothesis h that approximates c well over samples from $\mathcal{D}$.

Definition 2. *The* generalization error *of a hypothesis* $h : X \to \{0,1\}$ *is defined as* $\mathrm{error}_{\mathcal{D}}(c,h) = \Pr_{x\sim\mathcal{D}}[h(x) \neq c(x)]$. *If* $\mathrm{error}_{\mathcal{D}}(c,h) \leq \alpha$ *we say that* h *is* α-good *for* c *and* $\mathcal{D}$.

Definition 3 (PAC Learning [21]). *Algorithm* A *is an* (α,β,m)-PAC learner *for a concept class* $\mathcal{C}$ *over* X *using hypothesis class* $\mathcal{H}$ *if for all concepts* $c \in \mathcal{C}$, *all distributions* $\mathcal{D}$ *on* X, *given an input of* m *samples* $S = (z_1,\ldots,z_m)$, *where* $z_i = (x_i, c(x_i))$ *and* x_i *are drawn i.i.d. from* $\mathcal{D}$, *algorithm* A *outputs a hypothesis* $h \in \mathcal{H}$ *satisfying* $\Pr[\mathrm{error}_{\mathcal{D}}(c,h) \leq \alpha] \geq 1-\beta$. *The probability is taken over the random choice of the examples in* S *according to* $\mathcal{D}$ *and the coin tosses of the learner* A. *If* $\mathcal{H} \subseteq \mathcal{C}$ *then* A *is called a* proper *PAC learner; otherwise, it is called an* improper *PAC learner.*

Definition 4. *For a labeled sample* $S = (x_i, y_i)_{i=1}^m$, *the* empirical error *of* h *w.r.t.* S *is* $\mathrm{error}_S(h) = \frac{1}{m}|\{i : h(x_i) \neq y_i\}|$.

In private learning, we would like to accomplish the same goal as in non-private learning, while protecting the privacy of the input database.

Definition 5 (Private PAC Learning [14]). *Let* A *be an algorithm that gets an input* $S = \{z_1,\ldots,z_m\}$. *Algorithm* A *is an* $(\alpha,\beta,\epsilon,\delta,m)$-PPAC learner *for a concept class* $\mathcal{C}$ *over* X *using hypothesis class* $\mathcal{H}$ *if (i) Algorithm* A *is* (ϵ,δ)-*differentially private; and (ii) Algorithm* A *is an* (α,β,m)-PAC learner *for* $\mathcal{C}$ *using* $\mathcal{H}$. *When* $\delta = 0$ *(pure privacy) we omit it from the list of parameters.*

2.3 Private Data Release

Given a concept $c : X \to \{0,1\}$, the counting query $Q_c : X^* \to [0,1]$ is defined as $Q_c(S) = \frac{1}{|S|} \cdot |\{x_i \in S : c(x_i) = 1\}|$. That is, $Q_c(S)$ is the fraction of entries in S that satisfy the concept c. Given a database S, a sanitizer for a concept class C is required to approximate $Q_c(S)$ for every $c \in C$.

Definition 6 (Sanitization [4]). *Let* $\mathcal{C}$ *be a class of concepts mapping* X *to* $\{0,1\}$. *Let* A *be an algorithm that on an input database* $S \in X^*$ *outputs a description of a function* $\mathrm{est} : \mathcal{C} \to [0,1]$. *Algorithm* A *is an* $(\alpha,\beta,\epsilon,\delta,m)$-*improper-sanitizer for predicates in the class* $\mathcal{C}$, *if (i)* A *is* (ϵ,δ)-*differentially private; and, (ii) For every input* $S \in X^m$, $\Pr_A\left[\forall c \in C \;\; \left|Q_c(S) - \mathrm{est}(c)\right| \leq \alpha\right] \geq 1-\beta$.

If on an input database S algorithm A outputs another database $\hat{S} \in X^*$, and $\mathrm{est}(\cdot)$ *is defined as* $\mathrm{est}(c) = Q_c(\hat{S})$*, then algorithm A is called a* proper-sanitizer *(or simply a* sanitizer*). As before, when $\delta = 0$ (pure privacy) we omit it from the set of parameters. A database S and a function* est *(or two databases $S, \hat{S}$) are called α-*close *if $|Q_c(S) - \mathrm{est}(c)| \leq \alpha$ for every $c \in C$.*

Ignoring computational complexity, an $(\alpha, \beta, \epsilon, \delta, m)$-improper-sanitizer can always be transformed into a $(2\alpha, \beta, \epsilon, \delta, m)$-sanitizer, by finding a database $\hat{S}$ of m entries that is α-close to the returned estimation est (such a database exists, i.e., S itself).

2.4 Basic Differentially Private Mechanisms

The Laplace Mechanism. The most basic construction of differentially private algorithm is via the Laplace mechanism as follows.

Definition 7 (The Laplace Distribution). *A random variable has probability distribution* $\mathrm{Lap}(b)$ *if its probability density function is* $f(x) = \frac{1}{2b}\exp(-\frac{|x|}{b})$.

Definition 8 (Sensitivity). *A function $f : X^m \to \mathbb{R}$ has* sensitivity *k if for every neighboring $D, D' \in X^m$, it holds that $|f(D) - f(D')| \leq k$.*

Theorem 1 (The Laplacian Mechanism [9]). *Let $f : X^m \to \mathbb{R}$ be a sensitivity k function. The mechanism A which on input $D \in X^m$ outputs $A(D) = f(D) + \mathrm{Lap}(\frac{k}{\epsilon})$ preserves ϵ-differential privacy. Moreover,* $\Pr[|A(D) - f(D)| > \Delta] = \exp\left(-\frac{\epsilon\Delta}{k}\right)$.

The Exponential Mechanism. Let X be a domain and $\mathcal{H}$ a set of solutions. Given a quality function $q : X^* \times \mathcal{H} \to \mathbb{N}$, and a database $S \in X^*$, the goal is to chooses a solution $h \in \mathcal{H}$ approximately maximizing $q(S, h)$. The exponential mechanism, by McSherry and Talwar [16], chooses a solution $h \in H$ with probability proportional to $\exp(\epsilon \cdot q(S, h)/2)$.

Proposition 1 (Properties of the Exponential Mechanism). *(i) The exponential mechanism is ϵ-differentially private. (ii) Let* $\hat{e} \triangleq \max_{f \in \mathcal{H}}\{q(S, f)\}$. *The exponential mechanism outputs a solution h such that $q(S, h) \leq (\hat{e} - \Delta m)$ with probability at most $|\mathcal{H}| \cdot \exp(-\epsilon\Delta m/2)$.*

Stability and Privacy. We restate a simplified variant of algorithm $\mathcal{A}_{dist}$ by Smith and Thakurta [18], which is an instantiation of the PTR framework [8]. Let $q : X^* \times F \to \mathbb{N}$ be a sensitivity-1 quality function over a domain X and a set of solutions F. Given a database $S \in X^*$, the goal is to chooses a solution $f \in F$ maximizing $q(S, f)$, under the assumption that the optimal solution f scores much better than any other solution in F.

```
Algorithm $\mathcal{A}_{dist}$
Input: parameters $\epsilon, \delta$, database $S \in X^*$, sensitivity-1 quality function $q$.
 1. Let $f_1 \neq f_2$ be two highest score solutions in $F$, where $q(f_1, S) \geq q(f_2, S)$.
 2. Let $\text{gap} = q(f_1, S) - q(f_2, S)$ and $\text{gap}^* = \text{gap} + \text{Lap}(\frac{4}{\epsilon})$.
 3. If $\text{gap}^* < \frac{4}{\epsilon}\ln(\frac{1}{\delta}) + 2$ then output $\perp_1$ and halt.
 4. If $\text{gap} = 0$ then output $\perp_2$, otherwise output $f_1$.
```

Proposition 2 (Properties of $\mathcal{A}_{dist}$ [18]). *(i) Algorithm $\mathcal{A}_{dist}$ is (ϵ, δ)- differentially private. (ii) When given an input database S for which* $\text{gap} \geq \frac{4}{\epsilon}\ln(\frac{2}{\beta\delta})$*, algorithm $\mathcal{A}_{dist}$ outputs f_1 maximizing $q(f, S)$ with probability at least* $(1 - \beta)$.

3 Learning with Approximate Privacy

We present proper (ϵ, δ)-private learners for two simple concept classes, $\texttt{POINT}_d$ and $\texttt{INTERVAL}_d$, demonstrating separations between pure and approximate private proper learning.

3.1 (ϵ, δ)-PPAC Learner for $\texttt{POINT}_d$

For $j \in X_d$ let $c_j : X_d \to \{0, 1\}$ be defined as $c_j(x) = 1$ if $x = j$ and $c_j(x) = 0$ otherwise. Define the concept class $\texttt{POINT}_d = \{c_j\}_{j \in X_d}$. Note that the VC dimension of $\texttt{POINT}_d$ is 1, and, therefore, there exists a *proper* non-private learner for $\texttt{POINT}_d$ with sample complexity $O_{\alpha,\beta}(1)$. Beimel at el. [1] proved that every *proper* ϵ-private learner for $\texttt{POINT}_d$ must have sample complexity $\Omega(d) = \Omega(\log |\texttt{POINT}_d|)$. They also showed that there exists an *improper* ϵ-private learner for this class, with sample complexity $O_{\alpha,\beta,\epsilon}(1)$.

As we will now see, algorithm $\mathcal{A}_{dist}$ (defined in Section 2.4) can be used as a *proper* (ϵ, δ)-private learner for $\texttt{POINT}_d$ with sample complexity $O_{\alpha,\beta,\epsilon,\delta}(1)$. This is our first (and simplest) example separating the sample complexity of pure and approximate private learners. Consider the following algorithm.

```
Input: parameters $\alpha, \beta, \epsilon, \delta$, and a database $S \in X_{d+1}^m$.
 1. For every $x$, define $q(S, x)$ as the number of appearances of $(x, 1)$ in $S$.
 2. Execute $\mathcal{A}_{dist}$ on $S$ with the quality function $q$ and parameters $\frac{\alpha}{2}, \frac{\beta}{2}, \epsilon, \delta$.
 3. If the output was $j$ then return $c_j$.
 4. Else, if the output was $\perp_1$ or $\perp_2$ then return a random $c_i \in \texttt{POINT}_d$.
```

Lemma 1. *Let $\alpha, \beta, \epsilon, \delta$ be s.t. $\frac{1}{\alpha\beta} \leq 2^d$. The above algorithm is an efficient $(\alpha, \beta, \epsilon, \delta)$-PPAC proper learner for $\texttt{POINT}_d$ using a sample of $m = O\left(\frac{1}{\alpha\epsilon}\ln(\frac{1}{\beta\delta})\right)$ labeled examples.*

The proof is omitted from this extended abstract. For intuition, consider a target concept c_j and an underling distribution $\mathcal{D}$. Whenever $\mathcal{D}(j)$ is noticeable, a typical sample S contains many copies of the point j labeled as 1. As every other point $i \neq j$ will be labeled as 0, we expect $q(S, j)$ to be significantly higher than any other $q(S, i)$, and we can use algorithm $\mathcal{A}_{dist}$ to identify j.

3.2 Towards a Proper (ϵ, δ)-PPAC Learner for $\texttt{INTERVAL}_d$

For $0 \leq j \leq 2^d$ let $c_j : X_d \to \{0, 1\}$ be defined as $c_j(x) = 1$ if $x < j$ and $c_j(x) = 0$ otherwise. Define the concept class $\texttt{INTERVAL}_d = \{c_j\}_{0 \leq j \leq 2^d}$. Note that $\text{VC}(\texttt{INTERVAL}_d) = 1$, and, therefore, there exists a *proper* non-private learner for $\texttt{INTERVAL}_d$ with sample complexity $O_{\alpha,\beta}(1)$. As $|\,\texttt{INTERVAL}_d\,| = 2^d + 1$, one can use the generic construction of Kasiviswanathan et al. [14] and get a proper ϵ-private learner for this class with sample complexity $O(d)$. Beimel et al. [1] showed that this is in fact optimal, and every proper ϵ-private learner for this class must have sample complexity $\Omega(d)$. It is unknown whether there exists an improper ϵ-private learner for $\texttt{INTERVAL}_d$ with sample complexity $o(d)$.

Our learner for $\texttt{POINT}_d$ relied on a strong "stability" property of the problem: Given a labeled sample, either a random concept is (w.h.p.) a good output, or, there is exactly one consistent concept in the class, and every other concept has large empirical error. This, however, is not the case when dealing with $\texttt{INTERVAL}_d$. In particular, many hypotheses can have low empirical error, and changing a single entry of a sample S can significantly affect the set of hypotheses consistent with it.

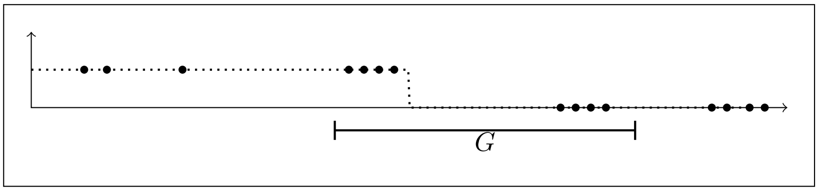
In Section 3.3, we present a proper (ϵ, δ)-private learner for $\texttt{INTERVAL}_d$ exhibiting sample complexity $\widetilde{O}_{\alpha,\beta,\epsilon,\delta}(16^{\log^*(d)})$. We use this section for motivating the construction. We start with two simplifying assumptions. First, when given a labeled sample S, we aim at choosing a hypothesis $h \in \texttt{INTERVAL}_d$ approximately minimizing the empirical error (rather than the generalization error). Second, we assume that we are given a "diverse" sample S that contains many points labeled as 1 and many points labeled as 0. Those two assumptions (and any other informalities made hereafter) will be removed in Section 3.3.

Assume we are given as input a sample $S = (x_i, y_i)_{i=1}^m$ labeled by some unknown $c_\ell \in \texttt{INTERVAL}_d$. We would now like to choose a hypothesis $h \in \texttt{INTERVAL}_d$ with small empirical error on S, and we would like to do so while accessing the sample S only through differentially private tools.

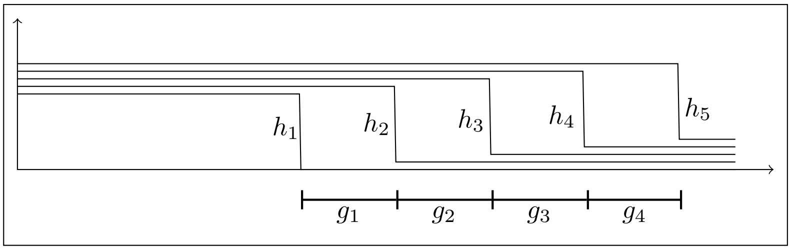
We will refer to points labeled as 1 in S as *ones*, and to points labeled as 0 as *zeros*. Imagine for a moment that we already have a differentially private algorithm that given S outputs an interval $G \subseteq X_d$ with the following two properties:

1. The interval G contains "a lot" of ones, *and* "a lot" of zeros in S.
2. Every interval $I \subseteq X_d$ of length $\leq \frac{|G|}{k}$ does not contain, simultaneously, "too many" ones and "too many" zeros in S, where k is some constant.

Such an interval will be referred to as a k-*good* interval. The figure below illustrates such an interval G, where the dotted line represents the (unknown) target concept, and the bold dots correspond to sample points.

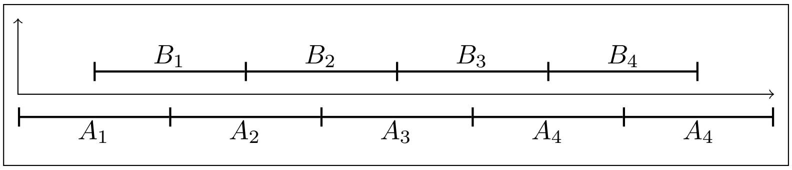


Given such a 4-good interval G, we can (without using the sample S) define a set H of five hypotheses, s.t. at least one of them has small empirical error. To see this, consider the figure below, where G is divided into four equal intervals g_1, g_2, g_3, g_4, and 5 hypotheses $h_1, \ldots, h_5$ are defined s.t. the points where they switch from one to zero are uniformly spread inside G. Now, as the interval G contains both ones and zeros, it must be that the target concept c_ℓ switches from 1 to 0 inside G. Assume without loss of generality that this switch occurs inside g_2. Note that g_2 is of length $\frac{|G|}{4}$ and, therefore, either does not contain too many ones, and h_2 is "close" to the target concept, or does not contain too many zeros, and h_3 is "close" to the target concept.



After defining such a set H, we could use the exponential mechanism to choose a hypothesis $h \in H$ with small empirical error on S. As the size of H is constant, this requires only a constant number of samples. To conclude, finding a k-good interval G (while preserving privacy) is sufficient for choosing a good hypothesis. We next explain how to find such an interval.

Assume, for now, that we have a differentially private algorithm that given a sample S, returns an *interval length* J s.t. there exists a 2-good interval $G \subseteq X_d$ of length $|G| = J$. This length J could be used to find an explicit 4-good interval as follows. Divide X_d into intervals $\{A_i\}$ of length $2J$, and into intervals $\{B_i\}$ of length $2J$ right shifted by J as in the figure below.



As the promised 2-good interval G is of length J, at least one of the above intervals contains G. If, e.g., $G \subseteq A_2$ then A_2 contains both a lot of zeros and a lot of ones. The target concept must switch inside A_2, and, therefore, every other $A_i \neq A_2$ cannot contain both zeros and ones. For every interval A_i, define its quality $q(A_i)$ to be the minimum between the number of zeros in A_i and the number of ones in A_i. We have, therefore, that $q(A_2)$ is big, while $q(A_i) = 0$ for every $A_i \neq A_2$. That is, A_2 scores much better than any other A_i under this quality function q. The sensitivity of $q()$ is one and we can use algorithm $\mathcal{A}_{dist}$ to privately identify A_2. Recall that $G \subseteq A_2$ is a 2-good interval, and that $|A_2| = 2|G|$. The identified A_2 is, therefore, a 4-good interval.

To conclude, if we could indeed find (while preserving privacy) a length J s.t. there exists a k-good interval G of that length – our task would be completed.

Computing the Interval Length *J*. At first attempt, one might consider preforming a binary search for such a length $0 \leq J \leq 2^d$, in which every comparison will be made using the Laplace mechanism. This will indeed preserve privacy. However, as there are d noisy comparisons, this solution will require a sample of size $d^{\Omega(1)}$ in order to achieve reasonable utility guarantees.

As a second attempt, one might consider preforming a binary search, not on $0 \leq J \leq 2^d$, but rather on the power j of an interval of length 2^j. That is, preforming a search for a power $0 \leq j \leq d$ for which there exists a 2-good interval of length 2^j. Here there are only $\log(d)$ noisy comparisons, and the sample size will be reduced to $\log^{\Omega(1)}(d)$. More specifically, for every power $0 \leq j \leq d$, define

$$Q(j) = \max_{\substack{[a,b]\subseteq X_d \\ b-a=2^j}} \left\{ \min \left\{ \begin{array}{c} \text{number of} \\ \text{zeros in } [a,b] \end{array}, \begin{array}{c} \text{number of} \\ \text{ones in } [a,b] \end{array} \right\} \right\}.$$

If, e.g., we have that $Q(j) = 100$ for some j, then *there exists* an interval $[a,b] \subseteq X_d$ of length 2^j that contains at least 100 ones and at least 100 zeros. Moreover, *every* interval of length $\leq 2^j$ either contains at most 100 ones, or, contains at most 100 zeros.

A binary search on $0 \leq j \leq d$ can (privately) yield an appropriate length $J = 2^j$ s.t. $Q(j)$ is "big enough" (and so, there exists an interval of length 2^j containing lots of ones and lots of zeros), while $Q(j-1)$ is "small enough" (and so, every interval of length $\frac{1}{2}2^j$ can not contain too many ones and too many zeros simultaneously).

Remark 1. A binary search as above would have to operate on noisy values of $Q(\cdot)$ (as otherwise differential privacy cannot be obtained). For this reason we set the bounds for "big enough" and "small enough" to overlap. Namely, we search for a value j such that $Q(j) \geq (1-\frac{3\alpha}{4})m$ and $Q(j-1) \leq (1-\frac{\alpha}{4})m$, where α is our approximation parameter, and m is the sample size.

We will apply recursion to reduce the costs of computing $J = 2^j$ to $2^{O(\log^*(d))}$. The tool performing the recursion would be formalized and analyzed in the next section. This tool will later be used in our construction of a proper (ϵ, δ)-private learner for $\texttt{INTERVAL}_d$.

3.3 Privately Approximating Concave Promise Problems

Definition 9. *A function $Q(\cdot)$ is* concave *if $Q(i), Q(j) \geq x$ implies $Q(\ell) \geq x$ for every $i \leq \ell \leq j$.*

Definition 10 (Concave Promise Problem). *A* Concave Promise Problem *consists of an interval of possible solutions $[0, T] = \{0, 1, \ldots, T\}$, a database $S \in X^m$, a sensitivity-1 quality function $Q : X^* \times [0, T] \to \mathbb{R}$, an approximation parameter α, and another parameter r (called a* quality promise*).*

If $Q(S, \cdot)$ is concave and if there exists an solution $p \in [0, T]$ for which $Q(S, p) \geq r$ then a good output for the problem is a solution $k \in [0, T]$ satisfying $Q(S, k) \geq (1 - \alpha)r$. The outcome is not restricted otherwise.

We are interested in solving concave promise problems while preserving differential privacy (privacy must be preserved even when $Q(S, \cdot)$ is not concave or $Q(S, p) < r$ for all $p \in [0, T]$). Our algorithm *Rec* is presented in Fig. 1 (see inline comments for some of the underlying intuition).

Lemma 2. *When executed on a range $[0, T]$, a sensitivity-1 quality function Q, and parameters ϵ, δ, algorithm Rec preserves $(\epsilon', 4\log^*(T)\delta)$-differential privacy, where $\epsilon' = \sqrt{6\log^*(T)\ln(\frac{1}{\log^*(T)\delta})} \cdot \epsilon + 6\log^*(T) \cdot \epsilon^2$.*

Lemma 3. *Let $Q : X^* \times [0, T] \to \mathbb{R}$ be a sensitivity-1 quality function, and let $S \in X^*$ be a database s.t. $Q(S, \cdot)$ is concave. Let $\alpha \leq \frac{1}{2}$ and let $\beta, \epsilon, \delta, r$ be s.t. $L(S, 0) \triangleq \max_i\{Q(S, i)\} \geq r \geq \left(\frac{2}{\alpha}\right)^{\log^*(T)} \frac{16}{\alpha\epsilon} \ln\left(\frac{32}{\beta\delta}\right)$. When executed on $S, [0, T], r, \alpha, \epsilon, \delta$, algorithm Rec fails to outputs an index j s.t. $Q(S, j) \geq (1-\alpha)r$ with probability at most $2\beta\log^*(T)$.*

Remark 2. The computational efficiency of algorithm *Rec* depends on the quality function $Q(\cdot, \cdot)$. Note, however, that it suffices to efficiently implement the top level call (i.e., without the recursion). This is because an iteration of algorithm *Rec*, operating on a range $[0, T]$, can easily be implemented in time poly(T), and the range given as input to recursive calls is logarithmic in the size of the initial range.

Algorithm *Rec* can be used as a proper $(\alpha, \beta, \epsilon, \delta, m)$-private learner for $\texttt{INTERVAL}_d$. The details of this straight forward application are omitted from this extended abstract.

Theorem 2. *There exists an efficient proper $(\alpha, \beta, \epsilon, \delta, m)$-PPAC learner for $\texttt{INTERVAL}_d$, where $m = O_{\alpha,\beta,\epsilon}\left(16^{\log^*(d)}\sqrt{\log^*(d)\log(\frac{1}{\delta})}\log\left(\frac{1}{\delta}\log^*(d)\right)\right)$.*

4 Sanitization with Approximate Privacy

Beimel et al. [1] showed that every pure ϵ-private sanitizer for $\texttt{POINT}_d$, must operate on databases of $\Omega(d)$ elements. In this section we state the existence of an

```
Algorithm Rec
Inputs: range $[0,T]$, quality function $Q$, quality promise $r$, parameters $\alpha,\epsilon,\delta$, and a sample $S$.

1. If $T \leq 32$, then use the exponential mechanism with the quality function $Q$ and the parameter $\epsilon$ to choose and return an index $j \in [0,\dots,T]$.
2. Let $T'$ be the smallest power of 2 s.t. $T' \geq T$, and define $Q(S,i) = 0$ for $T < i \leq T'$.
3. For $0 \leq j \leq \log(T')$ let $L(S,j) = \max_{\substack{[a,b]\subseteq[0,T'] \\ b-a+1=2^j}} \left( \min_{i\in[a,b]} \Big( Q(S,i) \Big) \right)$. For $j = \log(T')+1$ let $L(S,j) = \min\{0, L(S,\log(T'))\}$.
   % If $L(S,j) = x$ then in every interval $I \subseteq [0,T']$ of length $2^j$ there exists a point $i \in I$ s.t. $Q(S,i) \leq x$. Moreover, there exists an interval $I \subseteq [0,T']$ of length $2^j$ s.t. $Q(S,i) \geq x$ for all $i \in I$. Note that, as $L(S,j+1)$ maximizes the minimum over intervals bigger than $I$, it must be bounded by $x$.
4. Define the function $q(S,j) = \min\Big( L(S,j) - (1-\alpha)r, r - L(S,j+1) \Big)$ where $0 \leq j \leq \log(T')$.
   % If $q(S,j)$ is high for some $j$, then there exists an interval $I = [a, a+2^j-1]$ s.t. every $i \in I$ has a quality $Q(S,i) >> (1-\alpha)r$, and for every interval $I' = [a', a'+2^{j+1}-1]$ there exists $i' \in I'$ with quality $Q(S,i) << r$.
5. Let $R = \frac{\alpha}{2}r$.
   % $R$ is the promise parameter for the recursive call. Note that for the maximal $j$ with $L(S,j) \geq (1-\frac{\alpha}{2})r$ we get $q(S,j) \geq \frac{\alpha}{2}r = \text{R}$.
6. Execute Rec recursively on the range $\{0,\dots,\log(T')\}$, the quality function $q(\cdot,\cdot)$, the promise $R$, and $\alpha,\epsilon,\delta$. Denote the returned value by $k$, and let $K = 2^k$.
   % Assuming the recursive call was successful, $k$ is s.t. $q(S,k) \geq (1-\alpha)R = (1-\alpha)\frac{\alpha}{2}r$. That is, $L(S,k) \geq (1-\frac{\alpha}{2}-\frac{\alpha^2}{2})r$ and $L(S,k+1) \leq (1-\frac{\alpha}{2}+\frac{\alpha^2}{2})r$.
7. Divide $[0,T']$ into the following intervals of length $8K$ (the last ones might be trimmed):
   $A_1 = [0, 8K-1],\ A_2 = [8K, 16K-1],\ A_3 = [16K, 24K-1], \dots$
   $B_1 = [4K, 12K-1],\ B_2 = [12K, 20K-1],\ B_3 = [20K, 28K-1], \dots$
   % We show that in at least one of those two partitions (say the $\{A_i\}$'s), there exists a "good" interval $A_g$ s.t. $Q(S,i) = r$ for some $i \in A_g$, and $Q(S,i) \leq (1-\frac{\alpha}{2}+\frac{\alpha^2}{2})r$ for all $i \in \{0,\dots,T\} \setminus A_g$.
8. For every such interval $I \in \{A_i\} \cup \{B_i\}$ let $u(S,I) = \max_{i\in I} \Big( Q(S,i) \Big)$.
9. Use algorithm $\mathcal{A}_{dist}$ with parameters $\epsilon,\delta$ and the quality function $u(\cdot,\cdot)$, once to choose an interval $A \in \{A_i\}$, and once more to choose an interval $B \in \{B_i\}$.
   % By the properties of $\mathcal{A}_{dist}$, w.h.p. at least one of the returned $A$ and $B$ is "good".
10. Denote $A = [a,b]$ and $B = [c,d]$, and define $H = \{a + i\frac{K}{2} : 0 \leq i \leq 15\} \cup \{b + i\frac{K}{2} : 0 \leq i \leq 15\}$.
   % We show that $H$ contains an index with high quality.
11. Use the exponential mechanism with the quality function $Q(\cdot,\cdot)$ and parameter $\epsilon$ to choose and return an index $j \in H$.
```

Fig. 1. Algorithm *Rec*

(ϵ, δ)-private sanitizer for $\texttt{POINT}_d$ with sample complexity $O_{\alpha,\beta,\epsilon,\delta}(1)$. This separates the database size necessary for $(\epsilon, 0)$-private sanitizers from the database size sufficient for (ϵ, δ)-private sanitizers.

Recall that in our private PAC learner for $\texttt{POINT}_d$, given a typical labeled sample, there exists a unique concept in the class that stands out (we used algorithm $\mathcal{A}_{dist}$ to identify it). This is not the case in the context of sanitization, as a given database S may have many α-close sanitized databases $\hat{S}$. We will overcome this issue using the following private tool for approximating a restricted class of choosing problems.

4.1 The Choosing Mechanism

A function $q : X^* \times \mathcal{F} \to \mathbb{N}$ defines an *optimization problem* over the domain X and solution set $\mathcal{F}$: Given a dataset S over domain X choose $f \in \mathcal{F}$ which (approximately) maximizes $q(S, f)$. We are interested in a subset of these optimization problems, which we call *bounded-growth choice problems.* For this section we think of a database $S \subseteq X^*$ as a multiset.

Definition 11. *Given q and S define* $\text{opt}_q(S) = \max_{f\in\mathcal{F}}\{q(S, f)\}$. *A solution $f \in \mathcal{F}$ is called* α-good *for a database S if* $q(S, f) \geq \text{opt}_q(S) - \alpha|S|$.

Definition 12. *A scoring function* $q : X^* \times \mathcal{F} \to \mathbb{N}$ *is* k-bounded-growth *if:*
1. *$q(\emptyset, f) = 0$ for all $f \in \mathcal{F}$.*
2. *If $S_2 = S_1 \cup \{x\}$, then (i) $q(S_2, f) \geq q(S_1, f) \geq q(S_2, f) - 1$ for all $f \in \mathcal{F}$; and (ii) there are at most k solutions $f \in \mathcal{F}$ s.t. $q(S_1, f) < q(S_2, f)$.*

In words, the second requirement means that (i) Adding an element to the database could either have no effect on the score of a solution f, or can increase the score by exactly 1; and (ii) There could be at most k solutions whose scores are increased (by 1). Note that a k-bounded-growth scoring function is, in particular, a sensitivity-1 function as two neighboring S_1, S_2 must be of the form $D \cup \{x_1\}$ and $D \cup \{x_2\}$ respectively. Hence, $q(S_1, f) - q(S_2, f) \leq q(D, f) + 1 - q(D, f) = 1$ for every solution f.

The choosing mechanism below is a private algorithm for approximately solving bounded-growth choice problems. Step 1 of the algorithm checks whether a good solutions exist, as otherwise any solution is approximately optimal (and the mechanism returns $\perp$). Step 2 invokes the exponential mechanism, but with the *small* set $G(S)$ instead of $\mathcal{F}$. We get the following lemma.

Choosing Mechanism

Input: a database S of $m \geq \frac{16}{\alpha\epsilon} \cdot \max\left((\frac{\epsilon}{4} + \ln(\frac{1}{\delta})), \ln(\frac{16k}{\alpha\beta\epsilon})\right)$ elements, and parameters $\alpha, \beta, \epsilon, \delta$.

1. Set $\text{best}(S) = \max_{f\in\mathcal{F}} \{q(S, f)\} + \text{Lap}(\frac{4}{\epsilon})$. If $\text{best}(S) < \frac{\alpha m}{2}$ then halt and return $\perp$.
2. Let $G(S) = \{f \in \mathcal{F} : q(S, f) \geq 1\}$. Choose and return $f \in G(S)$ using the exponential mechanism with parameter $\frac{\epsilon}{2}$.

Lemma 4. *When q is a k-bounded-growth quality function, the choosing mechanism is (ϵ, δ)-differentially private. Moreover, given a database S the choosing mechanism outputs an α-good solution for S with probability at least $1 - \beta$.*

4.2 (ϵ, δ)-Private Sanitiazers

Using the above choosing mechanism, we construct a private sanitizer for $\texttt{POINT}_d$. We also construct a recursive sanitizer for $\texttt{INTERVAL}_d$, using both algorithm *Rec* and the choosing mechanism. Here we only state the results.

Theorem 3. *Fix $\alpha, \beta, \epsilon, \delta$. There exists an efficient $(\alpha, \beta, \epsilon, \delta, m)$-improper- sanitizer for $\texttt{POINT}_d$, where $m = O\left(\frac{1}{\alpha^{1.5}\epsilon}\sqrt{\ln(\frac{1}{\delta})\ln(\frac{1}{\alpha\beta\epsilon\delta})}\right)$.*

Theorem 4. *Fix $\alpha, \beta, \epsilon, \delta$. There exists an efficient $(\alpha, \beta, \epsilon, \delta, m)$-proper- sanitizer for $\texttt{INTERVAL}_d$, where $m = O\left(\frac{1}{\alpha^{2.5}\epsilon}8^{\log^*(d)}\sqrt{\log^*(d)}\log(\frac{1}{\alpha\delta})\log\left(\frac{\log^*(d)}{\alpha\beta\epsilon\delta}\right)\right)$.*

5 Sanitization and Proper Private PAC

Similar techniques are used for both data sanitization and private learning, suggesting relationships between the two. We now explore one such relationship in proving a lower bound on the sample complexity needed for sanitization (under pure differential privacy). In particular, we show a *reduction* from the task of private learning to the task of data sanitization, and then use a lower bound on private learners to derive a lower bound on data sanitization.

Notation. We will refer an element of X_{d+1} as $\boldsymbol{x}\circ y$, where $\boldsymbol{x} \in X_d$, and $y \in \{0, 1\}$.

5.1 Sanitization Implies Proper PPAC

For a given predicate c over X_d, we define the predicate c^{label} over X_{d+1} as

$$c^{label}(\boldsymbol{x} \circ y) = \begin{cases} 1, & c(\boldsymbol{x}) \neq y. \\ 0, & c(\boldsymbol{x}) = y. \end{cases}$$

Note that $c^{label}(\boldsymbol{x} \circ \sigma) = \sigma \oplus c(\boldsymbol{x})$ for $\sigma \in \{0, 1\}$. For a given class of predicates C over X_d, we define $C^{label} = \{c^{label} \ : \ c \in C\}$.

The next theorem states that for every concept class C, a sanitizer for C^{label} implies a private learner for C.

Theorem 5. *Let $\alpha, \epsilon \leq \frac{1}{8}$, and let C be a class of predicates. If there exists an $(\alpha, \beta, \epsilon, m)$-sanitizer A for C^{label}, then there exists a proper $((3\alpha + 2\beta), 2\beta, \epsilon, t)$-PPAC learner for C, where $t = \mathcal{O}_{\alpha,\beta}(m)$.*

Remark 3. Given an efficient proper-sanitizer for a class C, and assuming the existence of an efficient *non-private* learner for C, this reduction results in an efficient *private* learner for the class C.

Using the above reduction together with known lower bounds on the sample complexity of private learners, we get:

Theorem 6. *There exists an explicit concept class C over X_d such that every $(\alpha, \beta, \epsilon, m)$-sanitizer for C requires databases of size*

$$m = \Omega\left(\frac{1}{\alpha\epsilon}\mathrm{VC}(C) \cdot \log |X_d|\right).$$

6 Generic Label-Private Learner

The model of *label privacy* was defined as a relaxation of private learning, where privacy must only be preserved for the *labels* of the elements in the database, and not necessarily for the elements themselves. This is a reasonable privacy requirement when the elements are public and only their labels are private.

Consider a database $S = (x_i, y_i)_{i=1}^m$ containing labeled points from some domain X. We denote $S_x = (x_i)_{i=1}^m \in X^m$, and $S_y = (y_i)_{i=1}^m \in \{0,1\}^m$.

Definition 13 (Label-Private Learner [5]). *Let A be an algorithm that gets as input a database $S_x \in X^m$ and its labeling $S_y \in \{0,1\}^m$. Algorithm A is an* $(\alpha, \beta, \epsilon, m)$-Label Private PAC Learner *for a concept class $\mathcal{C}$ over X if*

Privacy. *$\forall S_x \in X^m$, algorithm $A(S_x, \cdot) = A_{S_x}(\cdot)$ is ϵ-differentially private (as in Definition 1);*

Utility. *Algorithm A is an* (α, β, m)-PAC learner *for $\mathcal{C}$ (as in Definition 3).*

Chaudhuri et al. [5] show lower bounds on the sample complexity of label-private learners for a class C in terms of its doubling dimension. As the next theorem states, the correct measure for characterizing the sample complexity of such learners is the VC dimension, and the sample complexity of label-private learners is actually of the same order as that of non-private learners (assuming α, β and ϵ are constants).

Theorem 7. *Let C be a concept class over a domain X. For every α, β, ϵ, there exists an $(\alpha, \beta, \epsilon, m)$-Label Private PAC learner for C, where $m = O_{\alpha,\beta,\epsilon}(VC(C))$. The learner might not be efficient.*

Acknowledgments. We thank Salil Vadhan and Jon Ullman for helpful discussions of ideas in this work.

References

1. Beimel, A., Kasiviswanathan, S.P., Nissim, K.: Bounds on the sample complexity for private learning and private data release. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 437–454. Springer, Heidelberg (2010)
2. Beimel, A., Nissim, K., Stemmer, U.: Characterizing the sample complexity of private learners. In: ITCS, pp. 97–110 (2013)

3. Blum, A., Dwork, C., McSherry, F., Nissim, K.: Practical privacy: The SuLQ framework. In: PODS, pp. 128–138. ACM (2005)
4. Blum, A., Ligett, K., Roth, A.: A learning theory approach to noninteractive database privacy. J. ACM 60(2), 12:1–12:25 (2013)
5. Chaudhuri, K., Hsu, D.: Sample complexity bounds for differentially private learning. In: COLT, vol. 19, pp. 155–186 (2011)
6. De, A.: Lower bounds in differential privacy. In: Cramer, R. (ed.) TCC 2012. LNCS, vol. 7194, pp. 321–338. Springer, Heidelberg (2012)
7. Dwork, C., Kenthapadi, K., McSherry, F., Mironov, I., Naor, M.: Our data, ourselves: Privacy via distributed noise generation. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 486–503. Springer, Heidelberg (2006)
8. Dwork, C., Lei, J.: Differential privacy and robust statistics. In: STOC 2009, pp. 371–380. ACM, New York (2009)
9. Dwork, C., McSherry, F., Nissim, K., Smith, A.: Calibrating noise to sensitivity in private data analysis. In: Halevi, S., Rabin, T. (eds.) TCC 2006. LNCS, vol. 3876, pp. 265–284. Springer, Heidelberg (2006)
10. Dwork, C., Naor, M., Reingold, O., Rothblum, G., Vadhan, S.: On the complexity of differentially private data release. In: STOC, pp. 381–390. ACM (2009)
11. Gupta, A., Hardt, M., Roth, A., Ullman, J.: Privately releasing conjunctions and the statistical query barrier. In: STOC, pp. 803–812. ACM, New York (2011)
12. Hardt, M., Talwar, K.: On the geometry of differential privacy. In: STOC,7 pp. 705–714 (2010)
13. Hardt, M.A.W.: A Study of Privacy and Fairness in Sensitive Data Analysis. PhD thesis, Princeton University (2011)
14. Kasiviswanathan, S.P., Lee, H.K., Nissim, K., Raskhodnikova, S., Smith, A.: What can we learn privately? In: FOCS, pp. 531–540. IEEE Computer Society (2008)
15. Kearns, M.J.: Efficient noise-tolerant learning from statistical queries. Journal of the ACM 45(6), 983–1006 (1998); Preliminary version in Proceedings of STOC 1993
16. McSherry, F., Talwar, K.: Mechanism design via differential privacy. In: FOCS, pp. 94–103. IEEE (2007)
17. Roth, A.: Differential privacy and the fat-shattering dimension of linear queries. In: Serna, M., Shaltiel, R., Jansen, K., Rolim, J. (eds.) APPROX and RANDOM 2010. LNCS, vol. 6302, pp. 683–695. Springer, Heidelberg (2010)
18. Smith, A., Thakurta, A.: Differentially private feature selection via stability arguments, and the robustness of the lasso. Manuscript (2012)
19. Ullman, J.: Answering $n^{2+o(1)}$ counting queries with differential privacy is hard. CoRR, abs/1207.6945 (2012)
20. Ullman, J., Vadhan, S.: PCPs and the hardness of generating private synthetic data. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 400–416. Springer, Heidelberg (2011)
21. Valiant, L.G.: A theory of the learnable. Communications of the ACM 27, 1134–1142 (1984)