THEORY OF COMPUTING, Volume 12 (890), 2016, pp. 1–61
www.theoryofcomputing.org

SPECIAL ISSUE: APPROX-RANDOM 2013

Private Learning and Sanitization: Pure vs. Approximate Differential Privacy

Amos Beimel* Kobbi Nissim† Uri Stemmer‡

Received October 2, 2013; Revised November 9, 2014; Published May 27, 2016

Abstract: We compare the sample complexity of private learning [Kasiviswanathan et al. 2008] and sanitization [Blum et al. 2008] under *pure* ε-differential privacy [Dwork et al. TCC 2006] and *approximate* (ε, δ)-differential privacy [Dwork et al. 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.

We define a family of optimization problems, which we call *Quasi-Concave Promise Problems*, that generalizes some of the tasks we consider. We observe that a quasi-concave promise problem can be privately approximated using a solution to a smaller instance of a quasi-concave promise problem. This allows us to construct an efficient recursive algorithm to solve such problems privately. Specifically, we construct private learners for

An extended abstract of this paper appeared in the Proceedings of the 17th International Workshop on Randomization and Computation, 2013 [5].

*Supported by a grant from the Israeli Science and Technology ministry and by an Israel Science Foundation grant 938/09.

†Work done while the second author was a Visiting Scholar at the Harvard Center for Research on Computation and Society, supported by NSF grant CNS-1237235. Partially supported by an Israel Science Foundation grant 2761/12.

‡Supported by the Ministry of Science and Technology (Israel), by the Check Point Institute for Information Security, and by the IBM PhD Fellowship Awards Program. Work done in part while the third author was visiting Harvard University supported by NSF grant CNS-1237235 and a gift from Google, Inc.

ACM Classification: K.4.1, I.2.6, F.2.0

AMS Classification: 68Q32, 68Q25, 68W20

Key words and phrases: differential privacy, sample complexity, private learning, sanitization

© 2016 Amos Beimel, Kobbi Nissim, and Uri Stemmer
Licensed under a Creative Commons Attribution License (CC-BY)
DOI: 10.4086/toc.2016.v012a001

point functions, threshold functions, and axis-aligned rectangles in high dimension. Similarly, we construct sanitizers for point functions and threshold functions.

We also examine the sample complexity of *label-private* learners, a relaxation of private learning where the learner is required to only protect the privacy of the labels in the sample. We show that the VC dimension completely characterizes the sample complexity of such learners, that is, the sample complexity of learning with label privacy is equal (up to constants) to learning without privacy.

1 Introduction

Learning is 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 private learning [22] and a related task—sanitization [7]—while preserving differential privacy [13]. We show striking differences between the required sample complexity for these tasks under ε-differential privacy [13] (also called *pure* differential privacy) and its variant (ε,δ)-differential privacy [11] (also called *approximate* differential privacy).

Differential privacy. Differential privacy protects the privacy of individuals by requiring that the information of an individual does not significantly affect the output. More formally, 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 set of A, the probability of this event is close up to a multiplicative factor of $e^{\varepsilon} \approx 1+\varepsilon$ whether A is applied on one database or on the other. *Approximate Differential Privacy* is a relaxation of pure differential privacy where the guarantee needs to be satisfied only for events whose probability is at least $\approx \delta$. We show that even a negligible $\delta > 0$ can have a significant effect on the sample complexity of private learning and sanitization.

Private learning. Private learning was introduced in [22] as a combination of Valiant's PAC learning model [29] and differential privacy. For now, we can think of a private learner as a differentially private algorithm that operates on a set of classified random examples, and outputs a hypothesis that misclassifies fresh examples with probability at most (say) $1/10$. The work on private learning has mainly focused on pure privacy. On the one hand, Blum et al. [6] and Kasiviswanathan et al. [22] have showed, via generic constructions, that every finite concept class C can be learned privately, using sample complexity proportional to $\mathrm{poly}(\log|C|)$ (often efficiently). On the other hand, a significant difference was shown between the sample complexity of *traditional* (non-private) learners (crystallized in terms of $\mathrm{VC}(C)$ and smaller than $\log|C|$ in many interesting cases) and private learners. 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 and to zero elsewhere). 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 [3]. 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 $O(1)$ [3]. This, however, comes with a price, as it was shown in [3] 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 complete characterization for the sample complexity of pure-private learners was recently given in [4], 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 $\Theta(\mathrm{RepDim}(C))$. Following that, Feldman and Xiao [17] showed an equivalence between the representation dimension of a concept C and the randomized one-way communication complexity of the evaluation problem for concepts from C. Using this equivalence they separated the sample complexity of pure-private learners from that of non-private ones. For example, they showed a lower bound of $\Omega(d)$ on the sample complexity of every pure-private (proper or improper) learner for the class $\texttt{THRESH}_d$ of threshold functions over the interval $[0,2^d-1]$. This is a strong separation from the non-private sample complexity, which is $O(1)$ (as the VC dimension of this class is constant).

We show that the sample complexity of proper learning with *approximate* differential privacy can be significantly lower than that satisfying *pure* differential privacy. Our starting point for this work is an observation that with *approximate* (ε,δ)-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 $\delta = 2^{-o(d)}$.

Sanitization. The notion of differentially private sanitization was introduced in the work of Blum et al. [7]. A sanitizer for a class of predicates C is a differentially private mechanism translating an input database S to an output database $\hat{S}$ such that $\hat{S}$ (approximately) agrees with S on the fraction of the entries satisfying φ for all $\varphi \in C$, where every predicate $\varphi \in C$ is a function from X to $\{0,1\}$. Blum et al. gave a generic construction of pure differentially private sanitizers exhibiting sample complexity $O(\mathrm{VC}(C)\log|X|)$. Lower bounds partially supporting this sample complexity were given by [25, 3, 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. We note that the construction of sanitizers is not generally computationally feasible [14, 28, 27].

1.1 Our contributions

To simplify the exposition, we omit in this section dependency on all variables except for d, corresponding to the representation length of domain elements.

Tools. A recent instantiation of the Propose-Test-Release (PTR) framework [12] by Smith and Thakurta [26] 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 thresholds and axis-aligned rectangles, and sanitization for point functions and thresholds:

- **Choosing Mechanism:** Given a *low-sensitivity* quality function, one can use the Exponential Mechanism [24] to choose an approximately maximizing solution. This requires, in general, a

database of size logarithmic in the number of possible solutions. We identify a subfamily 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 quasi-concave promise problems:** We define a family of optimization problems, which we call *Quasi-Concave Promise Problems*. The possible solutions are ordered, and *quasi-concavity* means that if two solutions $f \leq h$ have quality of at least $\mathfrak{X}$, then any solution $f \leq g \leq h$ also has quality of at least $\mathfrak{X}$. The optimization goal is, when there exists a solution with a promised quality of (at least) r, to find a solution with quality $\approx r$. We observe that a quasi-concave promise problem can be privately approximated using a solution to a smaller instance of a quasi-concave promise problem. This allows us to construct an efficient recursive algorithm solving such problems privately. We show that the task of learning $\texttt{THRESH}_d$ is, in fact, a quasi-concave promise problem, and it can be privately solved using our algorithm with sample size roughly $2^{O(\log^* d)}$. Sanitization for $\texttt{THRESH}_d$ does not exactly fit the model of quasi-concave promise problems but can still be solved by iteratively defining and solving a small number of quasi-concave promise problems.

Implications for private learning and sanitization. We give new private *proper*-learning algorithms for the classes $\texttt{POINT}_d$ and $\texttt{THRESH}_d$. We also construct a new private proper-learner for (a discrete version of) the class of all axis-aligned rectangles over n dimensions. Our algorithms exhibit sample complexity that is significantly lower than bounds given in prior work, separating pure and approximate private learning. Similarly, we construct sanitizers for $\texttt{POINT}_d$ and $\texttt{THRESH}_d$, again with sample complexity that is significantly lower than bounds given in prior work, separating sanitization in the pure and approximate privacy cases. Our algorithms are time-efficient.

Sanitization vs. private learning. Gupta et al. [18] have given reductions in both directions between agnostic learning of a concept class C, and the sanitization task for the same class C. The learners and sanitizers they consider are limited to access their data via statistical queries [23] (such algorithm can be easily transformed to satisfy differential privacy [6]). In Section 5 we show a similar reduction from the task of privately learning a concept class C to the sanitization task of C, where the sanitizer's access to the database is unrestricted. This allows us to exploit lower bounds on the sample complexity of private learners and show an explicit class of predicates C over a domain X for which every private sanitizer requires databases of size $\Omega(\mathrm{VC}(C)\log|X|)$. A similar lower bound was shown by Hardt and Rothblum [19], achieving tighter results in terms of the approximation parameter. Their work proves 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 [9] and references therein), where the learner is required to only protect the privacy of the labels in the sample. Chaudhuri et al. [9] have proved 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, that is, the sample complexity of learning with label privacy is equal (up to constants) to learning without privacy.

1.2 Open questions

This work raises two kinds of research directions. First, this work presents (time and sample efficient) private learners and sanitizers for relatively simple concept classes. It would be natural to try and construct private learners and sanitizers for more complex concept classes. In particular, constructing a (time and sample efficient) private learner for hyperplanes would be very interesting to the community.

Another very interesting research direction is to try and understand the sample complexity of approximate-private learners. Currently, no lower bounds are known on the sample complexity of such learners. On the other hand, no generic construction for such learners is known to improve the sample complexity achieved by the generic construction of Kasiviswanathan et al. [22] for pure-private learners. Characterizing the sample complexity of approximate-private learners is a very interesting open question.

1.3 Other related work

Most related to our work is the work on private learning and its sample complexity [22, 6, 3, 9] and the early work on sanitization [7]. Another related work is the work of De [10], who proved a separation between *pure* ε-differential privacy and *approximate* (ε,δ)-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 [21] separated pure from approximate differential privacy for $\delta = n^{-O(1)}$.

Another interesting gap between pure and approximate differential privacy is the following. Blum et al. [7] have given a generic construction of pure-private sanitizers, in which the sample complexity grows as $1/\alpha^3$ (where α is the approximation parameter). Following that, Hardt and Rothblum [20] showed that with approximate-privacy, the sample complexity can be reduce to grow as $1/\alpha^2$. Currently, it is unknown whether this gap is essential.

2 Preliminaries

Notations. We use $O_\gamma(f(t))$ as a shorthand for $O(h(\gamma)\cdot f(t))$ for some non-negative function h. In informal discussions, we sometimes use $\widetilde{O}(f(t))$ instead of $O(f(t)\cdot \mathrm{polylog}(f(t)))$. For example,

$$2^{\log^*(d)}\cdot\log^*(d) = \widetilde{O}\left(2^{\log^*(d)}\right).$$

We use X to denote an arbitrary domain, and X_d for the domain $\{0,1\}^d$. We use X^m (and respectively X_d^m) for the cartesian m^{th} power of X, i. e., $X^m = (X)^m$, and use $X^* = \bigcup_{m=0}^{\infty} X^m$.

Given a distribution $\mathcal{D}$ over a domain X, we denote

$$\mathcal{D}(j) \triangleq \Pr_{x\sim\mathcal{D}}[x=j] \quad \text{for } j\in X, \text{ and} \quad \mathcal{D}(J) \triangleq \Pr_{x\sim\mathcal{D}}[x\in J] \quad \text{for } J\subseteq X\,.$$

2.1 Differential privacy

Differential privacy aims at protecting information of individuals. We consider a database, where each entry contains information pertaining to an individual. An algorithm operating on databases is said to preserve differential privacy if a change of a single record of the database does not significantly change the output distribution of the algorithm. Intuitively, this means that whatever is learned about an individual could also be learned with her data arbitrarily modified (or without her data). Formally:

Definition 2.1. 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.

Definition 2.2 (Differential Privacy [13, 11])**.** A randomized algorithm A is (ε, δ)-differentially private if for all neighboring databases $S_1, S_2 \in X^m$, and for all sets $\mathcal{F}$ of outputs,

$$\Pr[A(S_1) \in \mathcal{F}] \leq \exp(\varepsilon) \cdot \Pr[A(S_2) \in \mathcal{F}] + \delta. \tag{2.1}$$

The probability is taken over the random coins of A. 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 $\delta > 0$, in which case δ is typically a negligible function of the database size m.

We will later present algorithms that access their input database using (several) differentially private mechanisms. We will use the following composition theorems.

Theorem 2.3 ([11])**.** *If A_1 and A_2 satisfy $(\varepsilon_1, \delta_1)$ and $(\varepsilon_2, \delta_2)$ differential privacy, respectively, then their concatenation $A(S) = \langle A_1(S), A_2(S) \rangle$ satisfies $(\varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)$-differential privacy.*

Moreover, a similar theorem holds for the adaptive case, where a mechanism interacts with k *adaptively chosen* differentially private mechanisms.

Theorem 2.4 ([11, 12])**.** *A mechanism that permits k adaptive interactions with mechanisms that preserves (ε, δ)-differential privacy (and does not access the database otherwise) ensures $(k\varepsilon, k\delta)$-differential privacy.*

Note that the privacy guarantee in the theorem deteriorates linearly with the number of interactions. By bounding the *expected* privacy loss in each interaction (as opposed to worst-case), Dwork et al. [15] showed the following stronger composition theorem, where privacy deteriorates (roughly) as $\sqrt{k}\varepsilon + k\varepsilon^2$ (rather than $k\varepsilon$).

Theorem 2.5 ([15], restated)**.** *Let $0 < \varepsilon, \delta' \leq 1$, and let $\delta \in [0,1]$. A mechanism that permits k adaptive interactions with mechanisms that preserves (ε, δ)-differential privacy (and does not access the database otherwise) ensures $(\varepsilon', k\delta + \delta')$-differential privacy, for $\varepsilon' = \sqrt{2k\ln(1/\delta')} \cdot \varepsilon + 2k\varepsilon^2$.*

2.2 Preliminaries from learning theory

2.2.1 The PAC model

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* C over X is a set of concepts (predicates) 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 $c \in C$. The learning algorithm is successful when it outputs a hypothesis h that approximates the target concept over samples from $\mathcal{D}$. More formally:

Definition 2.6. The *generalization error* of a hypothesis $h : X \to \{0,1\}$ is defined as

$$\text{error}_{\mathcal{D}}(c,h) = \Pr_{x\sim\mathcal{D}}[h(x) \neq c(x)] .$$

If $\text{error}_{\mathcal{D}}(c,h) \leq \alpha$ we say that h is α-*good* for c and $\mathcal{D}$.

Definition 2.7 (PAC Learning [29])**.** Algorithm A is an (α,β,m)-*PAC learner* for a concept class C over X using hypothesis class H if for all concepts $c \in C$, 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 each x_i is drawn i.i.d. from $\mathcal{D}$, algorithm A outputs a hypothesis $h \in H$ satisfying

$$\Pr[\text{error}_{\mathcal{D}}(c,h) \leq \alpha] \geq 1-\beta .$$

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 $H \subseteq C$ then A is called a *proper* PAC learner; otherwise, it is called an *improper* PAC learner.

Definition 2.8. For a labeled sample $S = (x_i, y_i)_{i=1}^m$, the *empirical error* of h is

$$\text{error}_S(h) = \frac{1}{m}|\{i : h(x_i) \neq y_i\}| .$$

2.2.2 The Vapnik–Chervonenkis Dimension

The Vapnik-Chervonenkis (VC) Dimension is a combinatorial measure of concept classes, which characterizes the sample size of PAC learners.

Definition 2.9 ([30])**.** Let C be a concept class over a domain X, and let $B = \{b_1,\ldots,b_\ell\} \subseteq X$. The set of all dichotomies (behaviors) on B that are realized by C is

$$\Pi_C(B) = \Big\{(c(b_1),\ldots,c(b_\ell)) : c \in C\Big\} .$$

Observe that $\Pi_C(B)$ is a subset of $\{0,1\}^\ell$ (as $c \in C$ maps into $\{0,1\}$). The set of dichotomies $\Pi_C(B)$ can be viewed as the "projection" of C on B.

Definition 2.10 ([30])**.** A set $B \subseteq X$ is *shattered* by C if $\Pi_C(B) = \{0,1\}^\ell$ (where $\ell = |B|$).

That is, B is shattered by C if C realizes all possible dichotomies over B.

Definition 2.11 (VC-Dimension [30])**.** The *VC-Dimension* of a concept class C (over a domain X), denoted as $\text{VC}(C)$, is the cardinality of the largest set $B \subseteq X$ shattered by C. If arbitrarily large finite sets can be shattered by C, then $\text{VC}(C) = \infty$.

Observe that as $\Pi_C(B) \leq |C|$ a set B can be shattered only if $|B| \leq \log|C|$ and hence $\text{VC}(C) \leq \log|C|$.

2.2.3 VC bounds

Classical results in computational learning theory state that a sample of size $\theta(\mathrm{VC}(C))$ is both necessary and sufficient for the PAC learning of a concept class C. The following two theorems give upper and lower bounds on the sample complexity.

Theorem 2.12 ([16])**.** *Any algorithm for PAC learning a concept class C must have sample complexity $\Omega(\mathrm{VC}(C)/\alpha)$, where α is the approximation parameter.*

Theorem 2.13 (VC-Dimension Generalization Bound [30, 8])**.** *Let C and $\mathcal{D}$ be a concept class and a distribution over a domain X. Let $\alpha, \beta > 0$, and*

$$m \geq \frac{8}{\alpha}\left(\mathrm{VC}(C)\ln\left(\frac{16}{\alpha}\right)+\ln\left(\frac{2}{\beta}\right)\right).$$

Fix a concept $c \in C$, and suppose that we draw a sample $S = (x_i, y_i)_{i=1}^m$, where x_i are drawn i.i.d. from $\mathcal{D}$ and $y_i = c(x_i)$. Then,

$$\Pr[\exists h \in C \text{ s.t. } \mathrm{error}_{\mathcal{D}}(h,c) > \alpha \wedge \mathrm{error}_S(h) = 0] \leq \beta.$$

So, for any concept class C, any algorithm that takes a sample of $m = \Omega_{\alpha,\beta}(\mathrm{VC}(C))$ labeled examples and produces as output a concept $h \in C$ that agrees with the sample is a PAC learner for C. Such an algorithm is a PAC learner for C using C (that is, both the target concept and the returned hypotheses are taken from the same concept class C), and, therefore, there always exist a hypothesis $h \in C$ with $\mathrm{error}_S(h) = 0$ (e. g., the target concept itself).

The next theorem handles (in particular) the agnostic case, in which a learning algorithm for a concept class C uses a hypotheses class $H \neq C$, and given a sample S (labeled by some $c \in C$), a hypothesis h with $\mathrm{error}_S(h) = 0$ might not exist in H.

Theorem 2.14 (VC-Dimension Agnostic Generalization Bound [2, 1])**.** *Let $\mathcal{D}$ and H be a distribution and a concept class over a domain X, and let $f : X \to \{0,1\}$ be some concept, not necessarily in H. For a sample $S = (x_i, f(x_i))_{i=1}^m$ where*

$$m \geq \frac{50\,\mathrm{VC}(H)}{\alpha^2}\ln\left(\frac{1}{\alpha\beta}\right)$$

and $\{x_i\}$ are drawn i.i.d. from $\mathcal{D}$, we have

$$\Pr\left[\forall\, h \in H:\ \left|\mathrm{error}_{\mathcal{D}}(h,f) - \mathrm{error}_S(h)\right| \leq \alpha\right] \geq 1-\beta.$$

Notice that in the agnostic case the sample complexity is proportional to $1/\alpha^2$, as opposed to $1/\alpha$ when learning a class C using C.

2.3 Private learning

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 2.15 (Private PAC Learning [22])**.** Let A be an algorithm that gets an input $S = \{z_1, \ldots, z_m\}$. Algorithm A is an $(\alpha, \beta, \varepsilon, \delta, m)$-*PPAC learner* for a concept class C over X using hypothesis class H if

PRIVACY. Algorithm A is (ε, δ)-differentially private (as in Definition 2.2);

UTILITY. Algorithm A is an (α, β, m)-*PAC learner* for C using H (as in Definition 2.7).

When $\delta = 0$ (pure privacy) we omit it from the list of parameters.

Note that the utility requirement in the definition is an average-case requirement, as the learner is only required to do well on typical samples (i. e., samples drawn i.i.d. from a distribution $\mathcal{D}$ and correctly labeled by a target concept $c \in C$). In contrast, the privacy requirement is a worst-case requirement, and Inequality (2.1) must hold for every pair of neighboring databases (no matter how they were generated, even if they are not consistent with any concept in C).

2.4 Sanitization

Given a database $S = (x_1, \ldots, x_m)$ containing elements from some domain X, the goal of *sanitization mechanisms* is to output (while preserving differential privacy) another database $\hat{S}$ that is in some sense similar to S. This returned database $\hat{S}$ is called a *sanitized* database.

Let $c : X \to \{0,1\}$ be a concept. The counting query $Q_c : X^* \to [0,1]$ is

$$Q_c(S) = \frac{1}{|S|} \cdot \left| \{i : c(x_i) = 1\} \right|.$$

That is, $Q_c(S)$ is the fraction of the entries in S that satisfy the concept c. Given a database S, a sanitizer for a concept class C is required to output a sanitized database $\hat{S}$ s.t. $Q_c(S) \approx Q_c(\hat{S})$ for every $c \in C$. For computational reasons, sanitizers are sometimes allowed not to return an actual database, but rather a data structure capable of approximating $Q_c(S)$ for every $c \in C$.

Definition 2.16. Let C be a concept class and let S be a database. A function $\mathrm{Est} : C \to [0,1]$ is called α-*close* to S if $|Q_c(S) - \mathrm{Est}(c)| \leq \alpha$ for every $c \in C$. If, furthermore, Est is defined in terms of a database $\hat{S}$, i. e., $\mathrm{Est}(c) = Q_c(\hat{S})$, we say that $\hat{S}$ is α-close to S.

Definition 2.17 (Sanitization [7])**.** Let 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 $\mathrm{Est} : C \to [0,1]$. Algorithm A is an $(\alpha, \beta, \varepsilon, \delta, m)$-improper-sanitizer for predicates in the class C, if

1. A is (ε, δ)-differentially private;
2. For every input $S \in X^m$, we have $\Pr_A[\text{Est is } \alpha\text{-close to } S] \geq 1 - \beta$.

The probability is over the coin tosses of algorithm A. 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.

Remark 2.18. Note that without the privacy requirements sanitization is a trivial task as it is possible to simply output the input database S. Furthermore, ignoring computational complexity, an $(\alpha,\beta,\varepsilon,\delta,m)$-improper-sanitizer can always be transformed into a $(2\alpha,\beta,\varepsilon,\delta,m)$-sanitizer, by finding a database $\hat{S}$ of m entries that is α-close to Est. Such a database must exist except with probability β (as in particular S is α-close to Est), and is 2α-close to S (by the triangle inequality).

The following theorems state some of the known results on the sample complexity of pure-privacy sanitizers. We start with an upper bound on the necessary sample complexity.

Theorem 2.19 (Blum et al. [7])**.** *There exists a constant Γ such that for any class of predicates C over a domain X, and any parameters α,β,ε, there exists an $(\alpha,\beta,\varepsilon,m)$-sanitizer for C, provided that the size of the database, denoted m, is at least*

$$m \geq \Gamma\left(\frac{\log|X|\cdot \mathrm{VC}(C)\cdot\log(1/\alpha)}{\alpha^3\varepsilon}+\frac{\log(1/\beta)}{\varepsilon\alpha}\right).$$

The algorithm might not be efficient.

The theorem states that, in principle, data sanitization is possible. The input database may be required to be as big as the representation size of elements in X. The next theorem states a general lower bound (far from the upper bound given in Theorem 2.19) on the sample complexity of any concept class C. Better bounds are known for specific concept classes [19].

Theorem 2.20 (Blum et al. [7])**.** *Let C be a class of predicates, and let $m \leq \mathrm{VC}(C)/2$. For any $0<\beta<1$ bounded away from 1 by a constant, for any $\varepsilon \leq 1$, if A is an $(\alpha,\beta,\varepsilon,m)$-sanitizer for C, then $\alpha \geq 1/(4+16\varepsilon)$.*

Recall that a proper sanitizer operates on an input database $S \in X^m$, and outputs a sanitized database $\hat{S} \in X^*$. The following is a simple corollary of Theorem 2.14, stating that the size of $\hat{S}$ does not necessarily depend on the size of the input database S.

Theorem 2.21. *Let C be a concept class. For any database S there exists a database $\hat{S}$ of size*

$$n = O\left(\frac{\mathrm{VC}(C)}{\alpha^2}\log\left(\frac{1}{\alpha}\right)\right) \quad \textit{such that} \quad \max_{h\in C}|Q_h(S)-Q_h(\hat{S})| \leq \alpha\,.$$

In particular, the theorem implies that an $(\alpha,\beta,\varepsilon,\delta,m)$-sanitizer A can always be transformed into a $(2\alpha,\beta,\varepsilon,\delta,m)$-sanitizer A' s.t. the sanitized databases returned by A' are always of fixed size

$$n = O\left(\frac{\mathrm{VC}(C)}{\alpha^2}\log\left(\frac{1}{\alpha}\right)\right).$$

This can be done by finding a database $\hat{S}$ of n entries that is α-close to the sanitized database returned by A. Using the triangle inequality, $\hat{S}$ is (w.h.p.) 2α-close to the input database.

2.5 Basic differentially-private mechanisms

2.5.1 The Laplace Mechanism

The most basic constructions of differentially private algorithms are via the Laplace Mechanism as follows.

Definition 2.22 (The Laplace Distribution)**.** A random variable has probability distribution $\mathrm{Lap}(b)$ if its probability density function is

$$f(x) = \frac{1}{2b}\exp\left(-\frac{|x|}{b}\right),$$

where $x \in \mathbb{R}$.

Definition 2.23 (Sensitivity)**.** The *sensitivity* of a function $f : X^m \to \mathbb{R}^n$ is the smallest k such that for every neighboring $D, D' \in X^m$, we have $\|f(D) - f(D')\|_1 \leq k$.

We use the term "sensitivity-k function" to mean a function of sensitivity $\leq k$.

Theorem 2.24 (The Laplace Mechanism [13])**.** *Let $f : X^m \to \mathbb{R}^n$ be a sensitivity-k function. The mechanism A that on input $D \in X^m$ adds independently generated noise with distribution $\mathrm{Lap}(k/\varepsilon)$ to each of the n output terms of $f(D)$ preserves ε-differential privacy. Moreover,*

$$\Pr\left[\exists i \; s.t. \; |A_i(D) - f_i(D)| > \Delta\right] \leq n \cdot \exp\left(-\frac{\varepsilon\Delta}{k}\right),$$

where $A_i(D)$ and $f_i(D)$ are the i^{th} coordinates of $A(D)$ and $f(D)$.

2.5.2 The Exponential Mechanism

We next describe the Exponential Mechanism of McSherry and Talwar [24]. Let X be a domain and H a set of solutions. Given a database $S \in X^*$, the Exponential Mechanism privately chooses a "good" solution h out of the possible set of solutions H. This "goodness" is quantified using a *quality function* that matches solutions to scores.

Definition 2.25 (Quality function)**.** A *quality function* is a function $q : X^* \times H \to \mathbb{R}$ that maps a database $S \in X^*$ and a solution $h \in H$ to a real number, identified as the score of the solution h w.r.t. the database S.

Given a quality function q and a database S, the goal is to chooses a solution h approximately maximizing $q(S,h)$. The Exponential Mechanism chooses a solution probabilistically, where the probability mass that is assigned to each solution h increases exponentially with its quality $q(S,h)$:

The Exponential Mechanism
Input: parameter ε, finite solution set H, database $S \in X^m$, and a sensitivity-1 quality function q.

1. Randomly choose $h \in H$ with probability $\dfrac{\exp(\varepsilon \cdot q(S,h)/2)}{\sum_{f \in H} \exp(\varepsilon \cdot q(S,f)/2)}$.
2. Output h.

Proposition 2.26 (Properties of the Exponential Mechanism)**.** *(i) The Exponential Mechanism is ε-differentially private. (ii) Let $\hat{e} \triangleq \max_{f\in H}\{q(S,f)\}$ and $\Delta > 0$. The Exponential Mechanism outputs a solution h such that $q(S,h) \leq (\hat{e} - \Delta m)$ with probability at most $|H| \cdot \exp(-\varepsilon\Delta m/2)$.*

Kasiviswanathan et al. [22] showed in 2008 that the Exponential Mechanism can be used as a generic private learner—when used with the quality function $q(S,h) = |\{i : h(x_i) = y_i\}|$, the probability that the Exponential Mechanism outputs a hypothesis h such that $\text{error}_S(h) > \min_{f\in H}\{\text{error}_S(f)\} + \Delta$ is at most $|H| \cdot \exp(-\varepsilon\Delta m/2)$. This results in a generic private proper-learner for every finite concept class C, with sample complexity $O_{\alpha,\beta,\varepsilon}(\log|C|)$.

2.5.3 Stability and privacy – $\mathcal{A}_{\text{dist}}$

We restate a simplified variant of algorithm $\mathcal{A}_{\text{dist}}$ by Smith and Thakurta [26], which is an instantiation of the Propose-Test-Release framework [12]. Let $q : X^* \times H \to \mathbb{N}$ be a sensitivity-1 quality function over a domain X and a set of solutions H. Given a database $S \in X^*$, the goal is to choose a solution $h \in H$ maximizing $q(S,h)$, under the assumption that the optimal solution h scores much better than any other solution in H.

Algorithm $\mathcal{A}_{\text{dist}}$
Input: parameters ε, δ, database $S \in X^*$, sensitivity-1 quality function q.

1. Let $h_1 \neq h_2$ be two highest score solutions in H, where $q(S,h_1) \geq q(S,h_2)$.
2. Let $\text{gap} = q(S,h_1) - q(S,h_2)$ and $\text{gap}^* = \text{gap} + \text{Lap}(1/\varepsilon)$.
3. If $\text{gap}^* < \frac{1}{\varepsilon}\log\left(\frac{1}{\delta}\right)$ then output $\perp$ and halt.
4. Output h_1.

Proposition 2.27 (Properties of $\mathcal{A}_{\text{dist}}$ [26])**.** *(i) Algorithm $\mathcal{A}_{\text{dist}}$ is (ε,δ)- differentially private. (ii) When given an input database S for which*

$$\text{gap} \geq \frac{1}{\varepsilon}\log\left(\frac{1}{\beta\delta}\right),$$

algorithm $\mathcal{A}_{\text{dist}}$ outputs h_1 maximizing $q(h,S)$ with probability at least $(1-\beta)$.

2.6 Concentration bounds

Let $X_1,\ldots,X_n$ be independent random variables where $\Pr[X_i = 1] = p$ and $\Pr[X_i = 0] = 1-p$ for some $0 < p < 1$. Clearly, $\mathbb{E}[\sum_i X_i] = pn$. The Chernoff bounds show that the sum is concentrated around this expected value:

$$\Pr\left[\sum\nolimits_i X_i > (1+\delta)pn\right] \leq \exp\left(-pn\delta^2/3\right) \qquad \text{for } \delta > 0\,,$$
$$\Pr\left[\sum\nolimits_i X_i < (1-\delta)pn\right] \leq \exp\left(-pn\delta^2/2\right) \qquad \text{for } 0 < \delta < 1\,.$$

3 Learning with approximate privacy

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

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

Definition 3.1. 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 et al. [3] 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 $O_{\alpha,\beta,\varepsilon}(1)$. An alternative private learner for this class was presented in [4].

As we will now see, algorithm $\mathcal{A}_{\text{dist}}$ (defined in Section 2.5) can be used as a *proper* (ε,δ)-private learner for $\texttt{POINT}_d$ with sample complexity $O_{\alpha,\beta,\varepsilon,\delta}(1)$. This is our first (and simplest) example separating the sample complexity of pure and approximate private proper-learners. Consider the following algorithm.

Algorithm LearnPoints
Input: parameters $\alpha,\beta,\varepsilon,\delta$, and a database $S \in (X_{d+1})^m$.

1. For every $x \in X_d$, define $q(S,x)$ as the number of appearances of $(x,1)$ in S.
2. Execute $\mathcal{A}_{\text{dist}}$ on S with the quality function q and parameters $\alpha/2,\beta/2,\varepsilon,\delta$.
3. If the output was j then return c_j.
4. Else, if the output was $\perp$ then return a random $c_i \in \texttt{POINT}_d$.

Lemma 3.2. *Let $\alpha,\beta,\varepsilon,\delta$ be s.t. $1/(\alpha\beta) \le 2^d$. Algorithm LearnPoints is an efficient $(\alpha,\beta,\varepsilon,\delta)$-PPAC proper learner for $\texttt{POINT}_d$ using a sample of*

$$m = O\left(\frac{1}{\alpha\varepsilon}\ln\left(\frac{1}{\beta\delta}\right)\right)$$

labeled examples.

For intuition, consider a target concept c_j and an underlying 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 $\mathcal{A}_{\text{dist}}$ to identify j.

Proof. As algorithm $\mathcal{A}_{\text{dist}}$ is (ε,δ)-differentially private, all that remains is the utility analysis. Fix a target concept $c_\ell \in \texttt{POINT}_d$ and a distribution $\mathcal{D}$ on X_d. In a typical sample S, the only point that can appear with the label 1 is ℓ, and algorithm $\mathcal{A}_{\text{dist}}$ has two possible outputs: $\ell,\perp$.

If $\mathcal{D}(\ell) > \alpha$ then (using the Chernoff bound), with probability at least $(1-\exp(-\alpha m/8))$, the labeled example $(\ell,1)$ appears in S at least $r = \alpha m/2$ times. Note that $q(S,\ell) \geq r$, and every $i \neq \ell$ has quality $q(S,i) = 0$. For

$$m \geq \frac{8}{\alpha\varepsilon} \ln\left(\frac{4}{\beta\delta}\right),$$

by Proposition 2.27, this gap is big enough s.t. algorithm $\mathcal{A}_{\text{dist}}$ outputs ℓ with probability at least $(1-\beta/2)$. Therefore, when $\mathcal{D}(\ell) > \alpha$, the probability of A outputting an α-good solution is at least $(1-\exp(-\alpha m/8))(1-\beta/2)$, which is at least $(1-\beta)$ for $m \geq (8/\alpha)\ln(2/\beta)$.

If, on the other hand, $\mathcal{D}(\ell) \leq \alpha$, then algorithm A will fail to output an α-good solution only if $\mathcal{A}_{\text{dist}}$ outputs $\perp$, and algorithm A chooses a hypothesis c_i s.t. $i \neq \ell$ and $\mathcal{D}(i) > \alpha$. But there could be at most $1/\alpha$ such points, and the probability of A failing is at most $1/(\alpha 2^d)$. Assuming $2^d \geq 1/(\alpha\beta)$, this probability is at most β. □

Remark 3.3. Recall that Algorithm LearnPoints outputs a random $c_i \in \texttt{POINT}_d$ whenever $\mathcal{A}_{\text{dist}}$ outputs $\perp$. In order for this random c_i to be good (w.h.p.) we needed 2^d (i. e., the number of possible concepts) to be at least $1/(\alpha\beta)$. This requirement could be avoided by outputting the all-zero hypothesis $c_0 \equiv 0$ whenever $\mathcal{A}_{\text{dist}}$ outputs $\perp$. However, this approach results in a *proper* learner only if we add the all-zero concept to $\texttt{POINT}_d$.

3.2 Towards a proper (ε,δ)-PPAC learner for $\texttt{THRESH}_d$

Definition 3.4. 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 $\texttt{THRESH}_d = \{c_j\}_{0 \leq j \leq 2^d}$.

Note that $\text{VC}(\texttt{THRESH}_d) = 1$, and, therefore, there exists a *proper* non-private learner for $\texttt{THRESH}_d$ with sample complexity $O_{\alpha,\beta}(1)$. As $|\texttt{THRESH}_d| = 2^d + 1$, one can use the generic construction of Kasiviswanathan et al. [22] and get a proper ε-private learner for this class with sample complexity $O_{\alpha,\beta,\varepsilon}(d)$. Feldman and Xiao [17] showed that this is in fact optimal, and every ε-private learner for this class (proper or improper) must have sample complexity $\Omega(d)$.

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 $\texttt{THRESH}_d$. 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 $\texttt{THRESH}_d$ with sample complexity (roughly) $2^{O(\log^*(d))}$. 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 $h \in \texttt{THRESH}_d$ 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 $c_\ell \in \texttt{THRESH}_d$. We would now like to choose a hypothesis $h \in \texttt{THRESH}_d$ with small empirical error on S, and we would like to do so while accessing the sample S only through differentially private tools.

We 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 $\leq |G|/k$ 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. Note that a k-good interval is, in particular, an ℓ-good interval for every $\ell \geq k$. Figure 1 illustrates such an interval G, where the dotted line represents the (unknown) target concept, and the bold dots correspond to sample points.

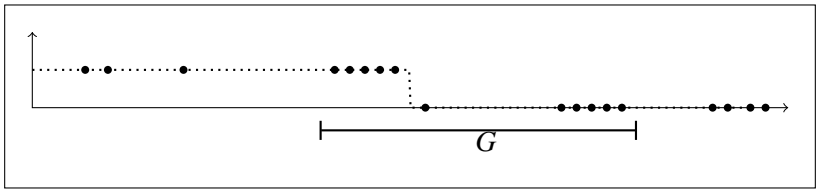

Figure 1: An illustration of a 4-good interval G.

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 Figure 2, where G is divided into four equal intervals g_1, g_2, g_3, g_4, and five hypotheses $h_1, \ldots, h_5$ are defined s.t. the points where they switch from one to zero are located at the edges of g_1, g_2, g_3, g_4.

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 $|G|/4$ 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. For this argument to go through we need "not too many" to be smaller than αm (say $(3/4)\alpha m$), where α is our approximation parameter and m is the sample size.

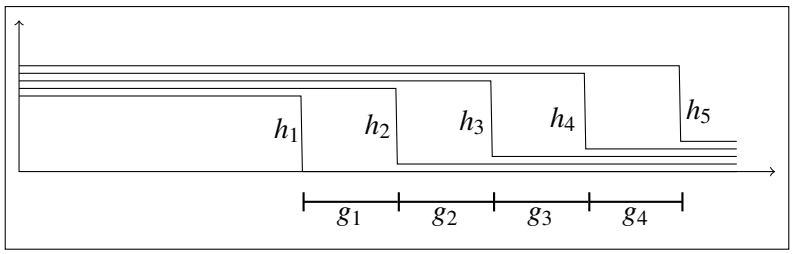

Figure 2: Extracting a small set of hypotheses from a good interval.

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 4-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 is 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 Figure 3.

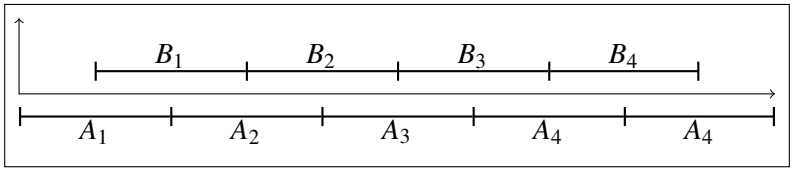

Figure 3: Dividing the axis X_d into intervals of length $2J$.

As the promised 2-good interval G is of length J, at least one of the these intervals contains G. We next explain how to privately choose such interval. 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. Therefore, $q(A_2)$ is large, 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 1 and we can use algorithm $\mathcal{A}_{\text{dist}}$ to privately identify A_2. It suffices, e. g., that $q(A_2) \geq (1/4)\alpha m$, and we can, therefore, set our "a lot" bound to be $(1/4)\alpha m$. 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 2-good interval G of that length, then 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. More specifically, for every length $0 \leq J \leq 2^d$, define

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

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

Note that $Q(\cdot)$ is a monotonically non-decreasing function, and that $Q(0) = 0$ (as in a correctly labeled sample a point cannot appear both with the label 1 and with the label 0). Recall our assumption that the sample S is "diverse" (contains many points labeled as 1 and many points labeled as 0), and, therefore, $Q(2^d)$ is large. Hence, there exists a J s.t. $Q(J)$ is "big enough" (say at least $(1/4)\alpha m$) while $Q(J-1)$ is "small enough" (say at most $(3/4)\alpha m$). That is, a J s.t. (1) there exists an interval of length J containing lots of ones and lots of zeros; and (2), every interval of length $< J$ cannot contain too many

ones and too many zeros simultaneously. Such a J can easily be (privately) obtained using a (noisy) binary search. However, as there are d noisy comparisons, this solution requires a sample of size $d^{O(1)}$ 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 is reduced to $\log^{\Omega(1)}(d)$. Again, a (noisy) binary search on $0 \leq j \leq d$ can (privately) yield an appropriate length $J = 2^j$ s.t. $Q(2^j)$ is "big enough," while $Q(2^{j-1})$ is "small enough." Such a $J = 2^j$ is, indeed, a length of a 2-good interval. Too see this, note that as $Q(2^j)$ is "big enough," there exists an interval of length 2^j containing lots of ones and lots of zeros. Moreover, as $Q(2^{j-1})$ is "small enough," every interval of length $2^{j-1} = (1/2)2^j$ cannot contain too many ones and too many zeros simultaneously.

Remark 3.5. 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 $Q(2^j) \geq (\alpha/4)m$ and $Q(2^{j-1}) \leq (3\alpha/4)m$, where α is our approximation parameter, and m is the sample size.

To summarize, using a binary search we find a length $J = 2^j$ such that there exists a 2-good interval of length J. Then, using $\mathcal{A}_{\text{dist}}$, we find a 4-good interval. Finally, we partition this interval to 4 intervals, and using the Exponential Mechanism we choose a starting point or end point of one of these intervals as our the threshold.

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 $\texttt{THRESH}_d$.

3.3 Privately approximating quasi-concave promise problems

We next define the notions that enable our recursive algorithm.

Definition 3.6. A function $Q(\cdot)$ is *quasi-concave* if $Q(\ell) \geq \min\{Q(i), Q(j)\}$ for every $i \leq \ell \leq j$.

Definition 3.7 (Quasi-concave promise problems)**.** A *quasi-concave promise problem* consists of an ordered set 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 quasi-concave and if there exists a 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.

Example 3.8. Consider a sample $S = (x_i, y_i)_{i=1}^m$, labeled by some target function $c_j \in \texttt{THRESH}_d$. The goal of choosing a hypothesis with small empirical error can be viewed as a quasi-concave promise problem as follows. Set the range of possible solutions to $[0,2^d]$, the approximation parameter to α and the quality promise to m. Define $Q(S,k) = |\{i : c_k(x_i) = y_i\}|$; i. e., $Q(S,k)$ is the number of points in S correctly classified by $c_k \in \texttt{THRESH}_d$. Note that the target concept c_j satisfies $Q(S,j) = m$. Our task is to find a hypothesis $h_k \in \texttt{THRESH}_d$ s.t. $\text{error}_S(h_k) \leq \alpha$, which is equivalent to finding $k \in [0,2^d]$ s.t. $Q(S,k) \geq (1-\alpha)m$.

To see that $Q(S,\cdot)$ is quasi-concave, let $u \le v \le w$ be s.t. $Q(S,u), Q(S,w) \ge \lambda$. Consider j, the index of the target concept, and assume w.l.o.g. that $j \le v$ (the other case is symmetric). That is, $j \le v \le w$. Note that c_v errs only on points in between j and v, and c_w errs on all these points. That is, $\text{error}_S(c_v) \le \text{error}_S(c_w)$, and, therefore, $Q(S,v) \ge \lambda$. See Figure 4 for an illustration.

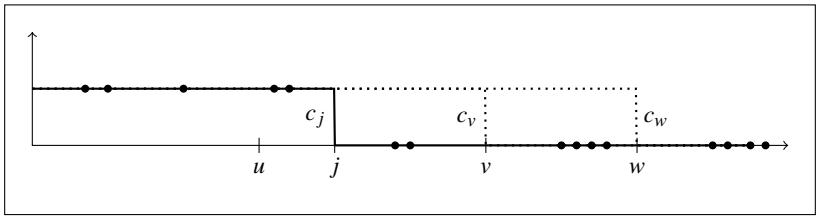

Figure 4: An illustration for Example 3.8. Here c_j is the target concept and the bold dots correspond to sample points. Note that c_w errs on every point on which c_v errs.

Remark 3.9. Note that if the sample S in Example 3.8 is *not* consistent with any $c \in \texttt{THRESH}_d$, then there is no j s.t. $Q(S,j) = m$, and the quality promise is void. Moreover, in such a case $Q(S,\cdot)$ might not be quasi-concave.

We are interested in solving quasi-concave promise problems while preserving differential privacy. As motivated by Remark 3.9, privacy must be preserved even when $Q(S,\cdot)$ is not quasi-concave or $Q(S,p) < r$ for all $p \in [0,T]$. Our algorithm RecConcave is presented in Figure 5 (see inline comments for some of the underlying intuition).

We start the analysis of Algorithm RecConcave by bounding the number of recursive calls.

Notation. Given an integer n, let $\log^{\lceil * \rceil}(n)$ denote the number of times that the function $\lceil \log(x) \rceil$ must be iteratively applied before the result is less than or equal to 1, i. e., $\log^{\lceil * \rceil}(n) = 1 + \log^{\lceil * \rceil} \lceil \log(n) \rceil$ if $n > 1$ and zero otherwise. Observe that $\log^{\lceil * \rceil}(n) = \log^*(n)$.[1]

Observation 3.10. On a range $[0,T]$ there could be at most $\log^{\lceil * \rceil}(T) = \log^*(T)$ recursive calls throughout the execution of RecConcave.

Before proceeding to the privacy analysis, we make the following simple observation.

Observation 3.11. Let $\{f_1, f_2, \ldots, f_N\}$ be a set of sensitivity-1 functions mapping X^* to $\mathbb{R}$. Then $f_{max}(S) = \max_i\{f_i(S)\}$ and $f_{min}(S) = \min_i\{f_i(S)\}$ are sensitivity-1 functions.

We now proceed with the privacy analysis of algorithm RecConcave.

Lemma 3.12. *When executed on a sensitivity-1 quality function Q, parameters ε, δ, and a bound on the recursion depth N, algorithm RecConcave preserves $(3N\varepsilon, 3N\delta)$-differential privacy.*

[1] Clearly $\log^{\lceil * \rceil}(n) \ge \log^*(n)$. Let ℓ be the smallest number of the form $2^{2^{\cdot^{\cdot^2}}}$ s.t. $\ell \ge n$. We have that $\log^*(\ell) = \log^*(n)$, and that $\log^{\lceil * \rceil}(\ell) = \log^*(\ell)$ (as all of the numbers in the iterative process of $\log^{\lceil * \rceil}(\ell)$ will be integers). As $\log^{\lceil * \rceil}(\cdot)$ is monotonically non-decreasing we get $\log^{\lceil * \rceil}(n) \le \log^{\lceil * \rceil}(\ell) = \log^*(\ell) = \log^*(n)$.

Algorithm RecConcave
Inputs: Range $[0,T]$, quality function Q, quality promise r, parameters $\alpha,\varepsilon,\delta$, and a database S.
Optional Input: a bound $N \geq 1$ on the number of recursive calls (set $N=\infty$ otherwise).

1. If $[(T \leq 32)$ or $(N=1)]$, then use the Exponential Mechanism with the quality function Q and the parameter ε to choose and return an index $j \in [0,\ldots,T]$. Otherwise set $N = N-1$.
2. Let T' be the smallest power of 2 s.t. $T' \geq T$, and define $Q(S,i) = \min\{0, Q(S,T)\}$ 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 (1) there exists an interval $I \subseteq [0,T']$ of length 2^j s.t. $Q(S,i) \geq x$ for all $i \in I$; and (2) in every interval $I \subseteq [0,T']$ of length 2^j there exists a point $i \in I$ s.t. $Q(S,i) \leq x$. Note that $L(S,j+1) \leq L(S,j)$. See Figure 6 for an illustration.
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 $Q(S,i) \ll r$. See Figure 6.
5. Let $R = \dfrac{\alpha}{2} r$.
   % 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 $q(S,j) \geq \frac{\alpha}{2} r = \mathrm{R}$.
6. Execute RecConcave recursively on the range $\{0,\ldots,\log(T')\}$, the quality function $q(\cdot,\cdot)$, the promise R, an approximation parameter $1/4$, and ε,δ,N. Denote the returned value by k, and let $K = 2^k$.
   % If the call to RecConcave was successful, then k is s.t. $q(S,k) \geq (1-\frac{1}{4})R = \frac{3\alpha}{8} r$. That is, $L(S,k) \geq (1-\frac{5\alpha}{8})r$ and $L(S,k+1) \leq (1-\frac{3\alpha}{8})r$. Note in the top level call the approximation parameter α is arbitrary (given as input), and that in all of the lower level calls the approximation parameter is fixed at $\frac{1}{4}$.
7. Divide $[0,T']$ into the following intervals of length $8K$ (the last ones might be trimmed):
$$A_1 = [0, 8K-1],\ A_2 = [8K, 16K-1],\ A_3 = [16K, 24K-1], \ldots$$
$$B_1 = [4K, 12K-1],\ B_2 = [12K, 20K-1],\ B_3 = [20K, 28K-1], \ldots$$
   % We show that in at least one of those two partitions (say the $\{A_i\}$), there exists a good interval A_g s.t. $Q(S,i) = r$ for some $i \in A_g$, and $Q(S,i) \leq (1-\frac{3\alpha}{8})r$ for all $i \in \{0,\ldots,T\} \setminus A_g$.

Figure 5: Algorithm RecConcave.

Algorithm RecConcave (continued)

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 $\mathcal{A}_{\text{dist}}$ with parameters ε, δ 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 $\mathcal{A}_{\text{dist}}$, w.h.p. at least one of the returned A and B is good.
10. Use the Exponential Mechanism with the quality function $Q(\cdot,\cdot)$ and parameter ε to choose and return an index $j \in (A \cup B)$.
    % We show that a constant fraction of the solutions in $(A \cup B)$ have high qualities, and, hence, the Exponential Mechanism needs only a constant sample complexity in order to achieve good utility guarantees.

Figure 5: Algorithm RecConcave, continued.

Proof. Note that since Q is a sensitivity-1 function, all of the quality functions defined throughout the execution of RecConcave are of sensitivity 1 (see Observation 3.11). In each recursive call algorithm RecConcave invokes at most three differentially private mechanisms—once with the Exponential Mechanism (on Step 1 or on Step 11), and at most twice with algorithm $\mathcal{A}_{\text{dist}}$ (on Step 9). As there are at most N recursive calls, we conclude that throughout the entire execution algorithm RecConcave invokes most $3N$ mechanisms, each (ε,δ)-differentially private. Hence, using Theorem 2.4, algorithm RecConcave is $(3N\varepsilon, 3N\delta)$-differentially private. □

We now turn to proving the correctness of algorithm RecConcave. As the proof is by induction (on the number of recursive calls), we need to show that each of the recursive calls to RecConcave is made with appropriate inputs. We first claim that the function $q(S,\cdot)$ constructed in Step 4 is quasi-concave. Note that for this claim we do not need to assume that $Q(S,\cdot)$ is quasi-concave.

Claim 3.13. *Let $Q : X^* \times [0,T] \to \mathbb{R}$ be a quality function, and let the functions $L(\cdot,\cdot)$ and $q(\cdot,\cdot)$ be as in steps 3, 4 of algorithm RecConcave. Then $q(S,\cdot)$ is quasi-concave for every $S \in X^*$.*

Proof. Fix $S \in X^*$. First observe that the function

$$L(S,j) = \max_{\substack{[a,b] \subseteq [0,T'] \\ b-a+1=2^j}} \left(\min_{i \in [a,b]} \Big(Q(S,i) \Big) \right)$$

is monotonically non-increasing (as a function of j). To see this, note that if $L(S,j) = \mathcal{X}$, then there exists an interval of length 2^j in which every point has quality at least $\mathcal{X}$. In particular, there exists such an interval of length $(1/2)2^j$, and $L(S,j-1) \geq \mathcal{X}$.

Now, let $i \leq \ell \leq j$ be s.t. $q(S,i), q(S,j) \geq x$. We get that $L(S,\ell) - (1-\alpha)r \geq L(S,j) - (1-\alpha)r \geq x$, and that $r - L(S,\ell+1) \geq r - L(S,i+1) \geq x$. Therefore, $q(S,\ell) \geq x$, and $q(S,\cdot)$ is quasi-concave. □

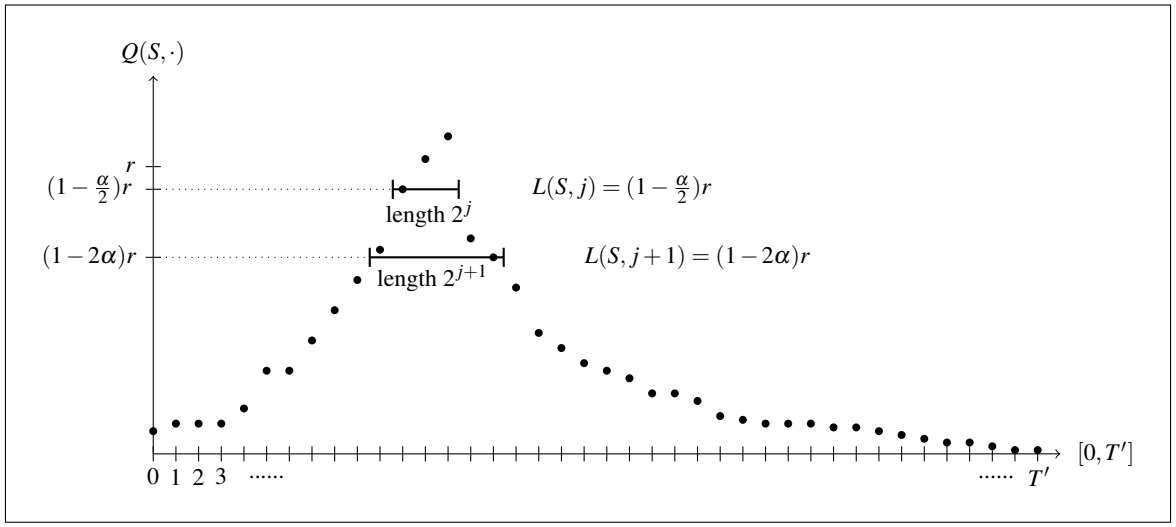

Figure 6: A demonstration for the functions L and q from Steps 3,4 of RecConcave. In the illustration, every interval of length 2^j contains at least one point with quality at most $(1-\alpha/2)r$, and there exists an interval of length 2^j containing only points with quality at least $(1-\alpha/2)r$. Hence, $L(S,j)=(1-\alpha/2)r$. Similarly, $L(S,j+1)=(1-2\alpha)r$. Therefore, for this j we have that $q(S,j)=\min\{L(S,j)-(1-\alpha)r, r-L(S,j+1)\}=(\alpha/2)r$. The reason for defining $q(\cdot,\cdot)$ is the following. We were interested in identifying a j with an appropriate lower bound on $L(S,j)$ and with an appropriate upper bound on $L(S,j+1)$. That is, in order to decide whether a given j is a good, we need to check both $L(S,j)$ and $L(S,j+1)$. After defining $q(S,\cdot)$, we can simply look for a j with a high $q(S,j)$. A high $q(S,j)$ implies upper and lower bounds (respectively) on $L(S,j), L(S,j+1)$.

Notation. We use $\log^{\lceil N\rceil}(\cdot)$ to denote the outcome of N iterative applications of the function $\lceil\log(\cdot)\rceil$, i. e.,

$$\log^{\lceil N\rceil}(n)=\underbrace{\lceil\log\lceil\log\lceil\cdots\lceil\log}_{N\text{ times}}(n)\rceil\cdots\rceil\rceil\rceil\,.$$

Observe that

$$\log^{\lceil N\rceil}(n)\le 2+\underbrace{\log\log\cdots\log}_{N\text{ times}}(n)$$

for every $N\le\log^*(n)$.[2]

[2]For example

$$\begin{aligned}\lceil\log\lceil\log\lceil\log(n)\rceil\rceil\rceil &\le \lceil\log\lceil\log(2+\log(n))\rceil\rceil\le\lceil\log\lceil\log(2\log(n))\rceil\rceil=\lceil\log\lceil 1+\log\log(n)\rceil\rceil\\ &\le\lceil\log(2+\log\log(n))\rceil\le\lceil\log(2\log\log(n))\rceil=\lceil 1+\log\log\log(n)\rceil\le 2+\log\log\log(n)\,.\end{aligned}$$

Lemma 3.14. *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 quasi-concave. Let $\alpha \leq 1/2$ and let $\beta, \varepsilon, \delta, r, N$ be s.t.*

$$\max_{i\in[0,T]} \{Q(S,i)\} \geq r \geq 8^N \cdot \frac{4}{\alpha\varepsilon} \left\{ \log\left(\frac{32}{\beta\delta}\right) + \log^{\lceil N \rceil}(T) \right\}.$$

When executed on $S, [0,T], r, \alpha, \varepsilon, \delta, N$, algorithm RecConcave fails to outputs an index j s.t. $Q(S,j) \geq (1-\alpha)r$ with probability at most $2\beta N$.

Proof. The proof is by induction on the number of recursive calls, denoted as t. For $t = 1$ (i. e., $T \leq 32$ or $N = 1$), the Exponential Mechanism ensures that for

$$r \geq \frac{2}{\alpha\varepsilon} \log\left(\frac{T}{\beta}\right),$$

the probability of algorithm RecConcave failing to output a j s.t. $Q(S,j) \geq (1-\alpha)r$ is at most β.

Assume that the stated lemma holds whenever algorithm RecConcave performs at most $t-1$ recursive calls, and let $S, [0,T], r, \alpha, \varepsilon, \delta, N$ be inputs (satisfying the conditions of Lemma 3.14) on which algorithm RecConcave preforms t recursive calls. Consider the first call in the execution of RecConcave on those inputs, and denote by T' the smallest power of 2 s.t. $T' \geq T$. In order to apply the inductive assumption, we need to show that for the recursive call in step 6, all the conditions of Lemma 3.14 hold.

We first note that by Claim 3.13, the quality function $q(S,\cdot)$ defined of step 4 is quasi-concave. We next show that the recursive call is preformed with an appropriate quality promise $R = (\alpha/2)r$. The conditions of the lemma ensure that $L(S,0) \geq r$, and, by definition, we have that $L(S, \log(T')+1) \leq 0$. There exists therefore a $j \in [0, \log(T')]$ for which $L(S,j) \geq (1-\alpha/2)r$, and $L(S,j+1) < (1-\alpha/2)r$. Plugging these inequalities in the definition of $q(S,j)$ we get that $q(S,j) \geq (\alpha/2)r$. Therefore, there exists an index $j \in [0, \log(T')]$ with quality $q(S,j) \geq R$. Moreover, the recursive call of step 6 executes RecConcave on the range $[0, \log(T')] = [0, \lceil \log(T) \rceil]$ with $(N-1)$ as the bound on the recursion depth, with $\widetilde{\alpha} \triangleq 1/4$ as the approximation parameter, and with a quality promise R satisfying

$$\begin{aligned} R &= \frac{\alpha}{2} r \\ &\geq \frac{\alpha}{2} \cdot 8^N \cdot \frac{4}{\alpha\varepsilon} \left\{ \log\left(\frac{32}{\beta\delta}\right) + \log^{\lceil N \rceil}(T) \right\} \\ &= 8^{N-1} \cdot \frac{4}{\widetilde{\alpha}\varepsilon} \left\{ \log\left(\frac{32}{\beta\delta}\right) + \log^{\lceil N-1 \rceil} \lceil \log(T) \rceil \right\}. \end{aligned}$$

We next show that w.h.p. at least one of the two intervals A, B chosen on Step 9, contains a lot of points with high score. Denote the index returned by the recursive call of step 6 as k. By the inductive assumption, with probability at least $(1-2\beta(N-1))$, the index k is s.t. $q(S,k) \geq (1-1/4)R = (3\alpha/8)r$; we proceed with the analysis assuming that this event happened. By the definition of $q(S,k)$, this means that $L(S,k) \geq q(S,k) + (1-\alpha)r \geq (1-5\alpha/8)r$ and that $L(S,k+1) \leq r - q(S,k) \leq (1-3\alpha/8)r$. That is, there exists an interval G of length 2^k s.t. $\forall i \in G$ we have $Q(S,i) \geq (1-5\alpha/8)r$, and every interval of length $2 \cdot 2^k$ contains at least one point i s.t. $Q(S,i) \leq (1-3\alpha/8)r$.

As promised by the conditions of the lemma, there exists a point $p \in [0,T]$ with quality $Q(S,p) \geq r$. Consider the following two intervals: $P_1 = [p - 2 \cdot 2^k + 1, p]$ and $P_2 = [p, p + 2 \cdot 2^k - 1]$, and denote $P = P_1 \cup P_2$ (these two intervals might be trimmed if p is close to the edges of $[0,T]$). Assuming P_1, P_2 are not trimmed, they both are intervals of length $2 \cdot 2^k$, and, therefore, each of them contains a point i_1, i_2 respectively with quality $Q(S,i_1), Q(S,i_2) \leq (1 - 3\alpha/8)r$. Therefore, by the quasi-concavity of $Q(S,\cdot)$, every point $\ell \geq i_2$ and every point $\ell \leq i_1$ must have quality at most $Q(S,\ell) \leq (1 - 3\alpha/8)r$ (otherwise, by the quasi-concavity of $Q(S,\cdot)$, every point between ℓ and p must have quality strictly greater than $(1 - 3\alpha/8)r$, contradicting the quality bound on i_1, i_2). See Figure 7.

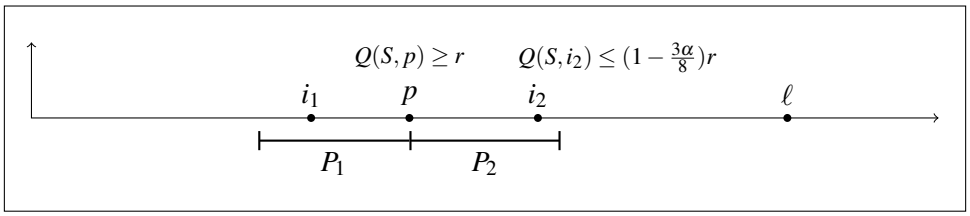

Figure 7: A point $\ell \notin P$ cannot have quality greater than $(1 - \frac{3\alpha}{8})r$.

Note that if P_1 (or P_2) is trimmed, then there are no points on the left of (or on the right of) P. So, the interval P contains the point p with quality $Q(S,p) \geq r$ and every point $i \in [0,T] \setminus P$ has quality of at most $(1 - 3\alpha/8)r$. Moreover, P is of length $4 \cdot 2^k - 1$. As the intervals of the partitions $\{A_i\}$ and $\{B_i\}$ are of length $8 \cdot 2^k$, and the $\{B_i\}$ are shifted by $4 \cdot 2^k$, there must exist an interval $C \in \{A_i\} \cup \{B_i\}$ s.t. $P \subseteq C$. Assume without loss of generality that $C \in \{A_i\}$.

Recall that the quality $u(S,\cdot)$ of an interval I is defined as the maximal quality $Q(S,i)$ of a point $i \in I$. Therefore, as $p \in C$, the quality of C is at least r. On the other hand, the quality of every $A_i \neq C$ is at most $(1 - 3\alpha/8)r$. That is, the interval C scores better (under u) than any other interval in $\{A_i\}$ by at least an additive factor of

$$\frac{3\alpha}{8} r \geq \frac{1}{\varepsilon} \log\left(\frac{1}{\beta\delta}\right).$$

By the properties of $\mathcal{A}_{\text{dist}}$, with probability at least $(1 - \beta)$, the chosen interval A in step 9 is s.t. $P \subseteq A$. We proceed with the analysis assuming that this is the case.

Consider again the interval P containing the point p, and recall that there exists an interval G of length 2^k containing only points with quality $Q(S,\cdot)$ of at least $(1 - 5\alpha/8)r$. Such an interval must be contained in P. Otherwise, by the quasi-concavity of $Q(S,\cdot)$, all the points between G and the point p must also have quality at least $(1 - 5\alpha/8)r$, and, in particular, P must indeed contain such an interval.

So, the chosen interval A in step 9 is of length $8 \cdot 2^k$, and it contains a subinterval of length 2^k in which every point has quality at least $(1 - 5\alpha/8)r$. That is, at least $1/16$ out of the points in $(A \cup B)$ has quality at least $(1 - 5\alpha/8)r$. Therefore, as

$$r \geq \frac{4}{\alpha\varepsilon} \log\left(\frac{16}{\beta}\right),$$

the Exponential Mechanism ensures that the probability of step 10 failing to return a point $h \in (A \cup B)$ with $Q(S,h) \geq (1 - \alpha)r$ is at most β.[3]

[3]As there are at least $(1/16)|A \cup B|$ solutions with quality at least $(1 - 5\alpha/8)r$, the probability that the Exponential

All in all, with probability at least $(1-2\beta(N-1)-2\beta) = (1-2\beta N)$, algorithm RecConcave returns an index $j \in [0,T]$ s.t. $Q(S,j) \geq (1-\alpha)r$. □

Combining Lemma 3.12 and Lemma 3.14 we get the following theorem.

Theorem 3.15. *Let algorithm RecConcave be executed on a range $[0,T]$, a sensitivity-1 quality function Q, a database S, a bound on the recursion depth N, privacy parameters $\varepsilon/(3N)$, $\delta/(3N)$, approximation parameter α, and a quality promise r. The following two statements hold:*

1. *Algorithm RecConcave preserves (ε,δ)-differential privacy.*
2. *If S is s.t. $Q(S,\cdot)$ is quasi-concave, and if*

$$\max_{i\in[0,T]}\{Q(S,i)\} \geq r \geq 8^N \cdot \frac{36N}{\alpha\varepsilon}\left\{\log\left(\frac{6N}{\beta\delta}\right) + \underbrace{\log\log\cdots\log}_{N \text{ times}}(T)\right\} \tag{3.1}$$

then algorithm RecConcave fails to outputs an index j s.t. $Q(S,j) \geq (1-\alpha)r$ with probability at most β.

Remark 3.16. Recall that the number of recursive calls on a range $[0,T]$ is always bounded by $\log^*(T)$, and note that for $N = \log^*(T)$ we have that $\log^{[N]}(T) \leq 1$. Therefore, the promise requirement in Inequality (3.1) can be replaced with

$$8^{\log^*(T)} \cdot \frac{36\log^*(T)}{\alpha\varepsilon}\log\left(\frac{12\log^*(T)}{\beta\delta}\right).$$

Remark 3.17. The computational efficiency of algorithm RecConcave 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 true because an iteration of algorithm RecConcave, operating on a range $[0,T]$, can easily be implemented in time $\text{poly}(T)$, and the range given as input to recursive calls is logarithmic in the size of the initial range.

3.4 A proper (ε,δ)-private learner for THRESH_d

As we will now see, algorithm RecConcave can be used as a proper $(\alpha,\beta,\varepsilon,\delta,m)$-private learner for THRESH_d. Recall Example 3.8 (showing that the goal of choosing a hypothesis with small empirical error can be viewed as a quasi-concave promise problem), and consider the following algorithm.

Mechanism outputs a specific solution $h \in (A \cup B)$ with $Q(S,h) \geq (1-\alpha)r$ is at most

$$\frac{\exp(\frac{\varepsilon}{2}(1-\alpha)r)}{\frac{1}{16}|A\cup B|\exp(\frac{\varepsilon}{2}(1-\frac{5\alpha}{8})r)}.$$

Hence, the probability that the Exponential Mechanism outputs *any* solution $h \in (A \cup B)$ with $Q(S,h) \geq (1-\alpha)r$ is at most

$$16\frac{\exp(\frac{\varepsilon}{2}(1-\alpha)r)}{\exp(\frac{\varepsilon}{2}(1-\frac{5\alpha}{8})r)},$$

which is at most β for our choice of r.

Algorithm LearnThresholds
Input: A labeled sample $S = (x_i, y_i)_{i=1}^m$ and parameters $\alpha, \varepsilon, \delta, N$.

1. Denote $\hat{\alpha} = \frac{\alpha}{2}$, $\hat{\varepsilon} = \frac{\varepsilon}{3N}$, and $\hat{\delta} = \frac{\delta}{3N}$.
2. For every $0 \leq j \leq 2^d$, define $Q(S, j) = |\{i : c_j(x_i) = y_i\}|$.
3. Execute algorithm RecConcave on the sample S, the range $[0, 2^d]$, the quality function $Q(\cdot,\cdot)$, the promise m, and parameters $\hat{\alpha}, \hat{\varepsilon}, \hat{\delta}, N$. Denote the returned value as k.
4. Return c_k.

Theorem 3.18. *For every $1 \leq N \leq \log^*(2^d)$, the Algorithm LearnThresholds is an efficient proper $(\alpha, \beta, \varepsilon, \delta, m)$-PPAC learner for $\mathtt{THRESH}_d$, where the sample size is*

$$m = O\left(\frac{8^N \cdot N}{\alpha \cdot \min\{\alpha, \varepsilon\}} \left\{\log\left(\frac{N}{\alpha\beta\delta}\right) + \underbrace{\log\log\cdots\log}_{N \text{ times}}(2^d)\right\}\right).$$

Proof. By Theorem 3.15, algorithm LearnThresholds is (ε, δ)-differentially private. For the utility analysis, fix a target concept $c_j \in \mathtt{THRESH}_d$, and a distribution $\mathcal{D}$ on X_d, and let S be a sample drawn i.i.d. from $\mathcal{D}$ and labeled by c_j. Define the following two good events:

E_1 : $\forall\, h \in \underset{d}{\mathtt{THRESH}}$, $|\mathrm{error}_{\mathcal{D}}(h, c_j) - \mathrm{error}_S(h)| \leq \frac{\alpha}{2}$.

E_2 : Algorithm RecConcave returns k s.t. $\mathrm{error}_S(c_k) \leq \frac{\alpha}{2}$.

Clearly, when both E_1, E_2 occur, algorithm LearnThresholds succeeds in outputting an α-good hypothesis for c_j and $\mathcal{D}$. Note that as $\mathrm{VC}(\mathtt{THRESH}_d) = 1$, Theorem 2.14 ensures that for

$$m \geq \frac{200}{\alpha^2} \ln\left(\frac{4}{\alpha\beta}\right),$$

event E_1 happens with probability at least $(1 - \beta/2)$.

Next, note that for the target concept c_j we have $Q(S, j) = m$, and algorithm RecConcave is executed in step 3 with a valid quality promise. Moreover, as shown in Example 3.8, algorithm RecConcave is executed with a quasi-concave quality function.

So, algorithm RecConcave is executed on step 3 with a valid quality promise and with a quasi-concave quality function. For

$$m \geq 8^N \cdot \frac{72N}{\alpha\varepsilon} \left\{\log\left(\frac{12N}{\beta\delta}\right) + \underbrace{\log\log\cdots\log}_{N \text{ times}}(T)\right\},$$

algorithm RecConcave ensures that with probability at least $(1 - \beta/2)$, the index k at step 2 is s.t. $Q(k) \geq (1 - \alpha/2)m$. The empirical error of c_k is at most $\alpha/2$ in such a case. Therefore, Event E_2

happens with probability at least $(1-\beta/2)$. Overall, we conclude that LearnThresholds is a proper $(\alpha,\beta,\varepsilon,\delta,m)$-PPAC learner for C, where

$$m \geq \max\left\{ \frac{200}{\alpha^2}\ln\left(\frac{4}{\alpha\beta}\right), \frac{8^N 72N}{\alpha\varepsilon}\left(\log\left(\frac{12N}{\beta\delta}\right) + \underbrace{\log\log\cdots\log}_{N \text{ times}}(2^d)\right)\right\}.$$

□

Remark 3.19. By using $N = \log^*(2^d)$ in Theorem 3.18, we can bound the sample complexity of LearnThresholds by

$$m = O\left(\frac{8^{\log^*(d)}\cdot\log^*(d)}{\alpha\cdot\min\{\alpha,\varepsilon\}}\log\left(\frac{\log^*(d)}{\alpha\beta\delta}\right)\right).$$

3.5 Axis-aligned rectangles in high dimension

Consider the class of all axis-aligned rectangles (or hyperrectangles) in the Euclidean space $\mathbb{R}^n$. A concept in this class could be thought of as the product of n intervals, one on each axis. We briefly describe an efficient approximate-private proper-learner for a discrete version of this class.

Formally,

Definition 3.20. Let $X_d^n = (\{0,1\}^d)^n$ denote a discrete n-dimensional domain, in which every axis consists of 2^d points $\{0,1,\ldots,2^d-1\}$. For every $\vec{a}=(a_1,\ldots,a_n),\vec{b}=(b_1,\ldots,b_n)\in X_d^n$ define the concept $c_{[\vec{a},\vec{b}]}: X_d^n \to \{0,1\}$ where $c_{[\vec{a},\vec{b}]}(\vec{x})=1$ if and only if for every $1\leq i\leq n$ we have $a_i\leq x_i\leq b_i$. Define the concept class of all axis-aligned rectangles over X_d^n as $\texttt{RECTANGLE}_d^n = \{c_{[\vec{a},\vec{b}]}\}_{\vec{a},\vec{b}\in X_d^n}$.

The VC dimension of this class is $2n$, and, thus, it can be learned non-privately with sample complexity $O_{\alpha,\beta}(n)$. Note that $|\texttt{RECTANGLE}_d^n| = 2^{O(nd)}$, and, therefore, the generic construction of Kasiviswanathan et al. [22] yields an inefficient proper ε-private learner for this class with sample complexity $O_{\alpha,\beta,\varepsilon}(nd)$.

In [23], Kearns gave an efficient (noise resistant) non-private learner for this class. The learning model there was a variant of the statistical queries model [23], in which the learner is also being given access to the underling distribution $\mathcal{D}$. Every learning algorithm in the statistical queries model can be transformed to satisfy differential privacy while preserving efficiency [6, 22]. However, as Kearns' algorithm assumes direct access to $\mathcal{D}$, this transformation cannot be applied directly.

Kearns' algorithm begins by sampling $\mathcal{D}$ and using the drawn samples to divide each axis $i\in[n]$ into $O(n/\alpha)$ intervals $\mathcal{I}_i=\{I\}$ with the property that the x_i component of a random point from $\mathcal{D}$ is approximately equally likely to fall into each of the intervals in $\mathcal{I}_i$. The algorithm proceeds by estimating the boundary of the target rectangle separately for every dimension i: For every interval $I\in\mathcal{I}_i$, the algorithm uses statistical queries to estimate the probability that a positively labeled input has its x_i component in I, i. e.,

$$p_I = \Pr_{x\sim\mathcal{D}}\left[\,(x \text{ is labeled } 1)\ \wedge\ (x_i\in I)\,\right].$$

The algorithm places the left boundary of the hypothesis rectangle in the i-th dimension at the left-most interval $I\in\mathcal{I}_i$ such that p_I is significant, and analogously on the right.

Note that once the interval sets $\mathcal{I}_i$ are defined for each axis $i \in [n]$, estimating every single p_I can be done via statistical queries, and can, therefore, be made private using the transformation of [6, 22]. Alternatively, estimating (simultaneously) all the p_I (on the i^{th} axis) could be done privately using the Laplace Mechanism. This use of the Laplace Mechanism is known as a histogram (see Theorem 2.24).

Thus, our task is to privately partition each axis. The straight forward approach for privately finding $\mathcal{I}_i$ is by a noisy binary search for the boundary of each of the n/α intervals (in each axis). This would result in $\Omega(d)$ noisy comparisons, which, in turn, results in a private learner with a high sample complexity.

We now overcome this issue using a sanitizer for $\texttt{THRESH}_d$. Such a sanitizer will be constructed in Section 4.2; here we use it for privately finding $\mathcal{I}_i$.

Theorem 3.21 (Restatement of Theorem 4.13)**.** *Fix $\alpha,\beta,\varepsilon,\delta$. There exists an efficient $(\alpha,\beta,\varepsilon,\delta,m)$-sanitizer for $\texttt{THRESH}_d$, where*

$$m = \widetilde{O}_{\beta,\varepsilon,\delta}\left(\frac{1}{\alpha^{2.5}} \cdot 8^{\log^*(d)}\right).$$

As we next explain, such a sanitizer can be used to (privately) divide the axes. Given an interval $[a,b] \subseteq X_d$ and a sample S, we denote the probability mass of $[a,b]$ under $\mathcal{D}$ as $\mathcal{D}[a,b]$, and the number of sample points in this interval as $\#_S[a,b]$. Standard arguments in learning theory (specifically, Theorem 2.14) state that for a large enough sample (whose size is bigger than the VC dimensions of the intervals class) w.h.p. $(1/|S|)\,\#_S[a,b] \approx \mathcal{D}[a,b]$ for *every* interval $[a,b] \subseteq X_d$.

On an input database $S \in (X_d)^*$, a sanitizer for $\texttt{THRESH}_d$ outputs an alternative database $\hat{S} \in (X_d)^*$ s.t.

$$\frac{1}{|\hat{S}|}\#_{\hat{S}}[0,b] \approx \frac{1}{|S|}\#_S[0,b]$$

for every interval $[0,b] \subseteq X_d$. Hence, for every interval $[a,b] \subseteq X_d$ we have that

$$\begin{aligned}
\frac{1}{|\hat{S}|}\#_{\hat{S}}[a,b] &= \frac{1}{|\hat{S}|}\#_{\hat{S}}[0,b] - \frac{1}{|\hat{S}|}\#_{\hat{S}}[0,a-1] \\
&\approx \frac{1}{|S|}\#_S[0,b] - \frac{1}{|S|}\#_S[0,a-1] \\
&= \frac{1}{|S|}\#_S[a,b] \\
&\approx \mathcal{D}[a,b].
\end{aligned}$$

So, in order to divide the i^{th} axis we apply a sanitizer for $\texttt{THRESH}_d$, and divide the axis using the returned sanitized database. In order to accumulate error of up to α/n on each axis (as required by Kearns' algorithm), we need to execute the sanitizer with an approximation parameter of (roughly) α/n. Every such execution requires, therefore, a sample of $\widetilde{O}_{\alpha,\beta,\varepsilon,\delta}\left(n^{2.5} \cdot 8^{\log^*(d)}\right)$ elements. As there are n such executions (one for each axis), using Theorem 2.5 (composition theorem), the described learner is of sample complexity $\widetilde{O}_{\alpha,\beta,\varepsilon,\delta}\left(n^3 \cdot 8^{\log^*(d)}\right)$.

Theorem 3.22. *There exists an efficient $(\alpha,\beta,\varepsilon,\delta,m)$-PPAC proper-learner for $\texttt{RECTANGLE}_d^n$, where*

$$m = O\left(\frac{n^3}{\alpha^{2.5}\varepsilon} \cdot 8^{\log^*(d)} \cdot \log^*(d) \cdot \log\left(\frac{n}{\alpha\delta}\right) \cdot \log\left(\frac{n \cdot \log^*(d)}{\alpha\beta\varepsilon\delta}\right)\right).$$

This should be contrasted with $\theta_{\alpha,\beta}(n)$, which is the non-private sample complexity for this class (as the VC-dimension of $\texttt{RECTANGLE}_d^n$ is $2n$), and with $\theta_{\alpha,\beta,\varepsilon}(nd)$ which is the pure-private sample complexity for this class.[4]

4 Sanitization with approximate privacy

In this section we present (ε,δ)-private sanitizers for several concept classes, and separate the database size necessary for $(\varepsilon,0)$-private sanitizers from the database size sufficient for (ε,δ)-private sanitizers.

4.1 The Choosing Mechanism

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 $\mathcal{A}_{\text{dist}}$ to identify it). This is not the case in the context of sanitization, as a given database S can have many α-close sanitized databases $\hat{S}$. We will overcome this issue by using the following private tool for approximating a restricted class of choosing problems.

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 $f \in \mathcal{F}$ that (approximately) maximizes $q(S,f)$. We are interested in a subset of these optimization problems, which we call *bounded-growth choice problems*. In this section we consider a database $S \subseteq X^*$ as a multiset.

Definition 4.1. 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 4.2. A quality 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) $q(S_1,f) + 1 \geq q(S_2,f) \geq q(S_1,f)$ for all $f \in \mathcal{F}$; and (ii) there are at most k solutions $f \in \mathcal{F}$ s.t. $q(S_2,f) > q(S_1,f)$.

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 quality function is, in particular, a sensitivity-1 function as two neighboring S, S' must be of the form $D \cup \{x_1\}$ and $D \cup \{x_2\}$ respectively. Hence, $q(S,f) - q(S',f) \leq q(D,f) + 1 - q(D,f) = 1$ for every solution f.

Example 4.3. As an example of a 1-bounded growth quality function, consider the following $q : X^* \times X \to \mathbb{N}$. Given a database $S = (x_1,\ldots,x_m)$ containing elements from some domain X, define

$$q(S,a) = \left|\{i : x_i = a\}\right|.$$

That is, $q(S,a)$ is the number of appearances of a in S. Clearly, $q(\emptyset,f) = 0$ for all $f \in X$. Moreover, adding an element $a \in X$ to a database S increases by 1 the quality of $q(S,a)$, and does not effect the quality of every other $b \neq a$.

[4] The general construction of Kasiviswanathan et al. [22] yields an (inefficient) pure-private proper-learner for this class with sample complexity $O_{\alpha,\beta,\varepsilon}(nd)$. Feldman and Xiao [17] showed that this is in fact optimal, and every ε-private (proper or improper) learner for this class must have sample complexity $\Omega(nd)$.

The Choosing Mechanism (in Figure 8) 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 $\mathcal{F}$.

The Choosing Mechanism
Input: a database S, a quality function q, and parameters $\alpha, \beta, \varepsilon, \delta$.

1. Set $\text{best}(S) = \max_{f \in \mathcal{F}} \{q(S,f)\} + \text{Lap}(4/\varepsilon)$. 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 $\varepsilon/2$.

Figure 8: The Choosing Mechanism.

Lemma 4.4. *When q is a k-bounded-growth quality function, the Choosing Mechanism preserves (ε, δ)-differential privacy for databases of*

$$m \geq \frac{16}{\alpha\varepsilon} \ln\left(\frac{16k}{\alpha\beta\varepsilon\delta}\right)$$

elements.

Proof. Let S, S' be neighboring databases of m elements. We need to show that $\Pr[A(S) \in R] \leq \exp(\varepsilon) \cdot \Pr[A(S') \in R] + \delta$ for any set of outputs R. Note first that by the properties of the Laplace Mechanism,

$$\begin{aligned} \Pr[A(S) = \perp] &= \Pr\left[\text{best}(S) < \frac{\alpha m}{2}\right] \\ &\leq \exp\left(\frac{\varepsilon}{4}\right) \cdot \Pr\left[\text{best}(S') < \frac{\alpha m}{2}\right] \\ &= \exp\left(\frac{\varepsilon}{4}\right) \cdot \Pr[A(S') = \perp]. \end{aligned} \tag{4.1}$$

Case (a): $q(S,f) < \frac{\alpha m}{4}$ for all f. Using

$$m \geq \frac{16}{\alpha\varepsilon} \ln\left(\frac{1}{2\delta}\right),$$

we get that

$$\Pr[A(S) \neq \perp] \leq \Pr\left[\text{Lap}\left(\frac{4}{\varepsilon}\right) > \frac{\alpha m}{4}\right] = \frac{1}{2}\exp\left(-\frac{\varepsilon}{4}\frac{\alpha m}{4}\right) \leq \delta.$$

Hence, for every set of outputs R

$$\begin{aligned} \Pr[A(S) \in R] &\leq \mathbb{1}_{(\perp \in R)} \cdot \Pr[A(S) = \perp] + \Pr[A(S) \neq \perp] \\ &\leq \mathbb{1}_{(\perp \in R)} \cdot \exp\left(\frac{\varepsilon}{4}\right) \cdot \Pr[A(S') = \perp] + \delta \\ &\leq \exp\left(\frac{\varepsilon}{4}\right) \cdot \Pr[A(S') \in R] + \delta. \end{aligned}$$

Case (b): There exists $\hat{f}$ s.t. $q(S,\hat{f}) \geq \frac{\alpha m}{4}$. Let $G(S)$ and $G(S')$ be the sets used in step 2 in the execution on S and on S' respectively. We will show that the following two facts hold:

Fact 1: For every $f \in G(S) \setminus G(S')$, we have $\Pr[A(S) = f] \leq \frac{\delta}{k}$.

Fact 2: For every possible output $f \notin G(S) \setminus G(S')$, we have $\Pr[A(S) = f] \leq e^{\varepsilon} \Pr[A(S') = f]$.

We first show that the two facts imply that the lemma holds for Case (b). Let $B \triangleq G(S) \setminus G(S')$, and note that as q is k-growth-bounded, $|B| \leq k$. Denote $B = \{b_1, \ldots, b_\ell\}$, where $\ell \leq k$. Using the two facts, for every set of outputs R we have

$$\begin{aligned}
\Pr[A(S) \in R] &= \Pr[A(S) \in R \setminus B] + \sum_{i:\, b_i \in R} \Pr[A(S) = b_i] \\
&\leq e^{\varepsilon} \Pr[A(S') \in R \setminus B] + \sum_{i:\, b_i \in R} \frac{\delta}{k} \\
&\leq e^{\varepsilon} \Pr[A(S') \in R] + \delta.
\end{aligned}$$

For proving Fact 1, let $f \in G(S) \setminus G(S')$. That is, $q(S,f) \geq 1$ and $q(S',f) = 0$. As q is (in particular) a sensitivity-1 function, it must be, therefore, that $q(S,f) = 1$. As there exists $\hat{f} \in S$ with $q(S,\hat{f}) \geq \alpha m/4$, we have that

$$\Pr[A(S) = f] \leq \Pr\left[\begin{array}{c}\text{The exponential}\\ \text{mechanism chooses } f\end{array}\right] \leq \frac{\exp(\frac{\varepsilon}{4})}{\exp(\frac{\varepsilon}{4}\frac{\alpha m}{4})},$$

which is at most δ/k for

$$m \geq \frac{16}{\alpha\varepsilon}\left(\frac{\varepsilon}{4} + \ln\left(\frac{k}{\delta}\right)\right).$$

For proving Fact 2, let $f \notin G(S) \setminus G(S')$ be a possible output of $A(S)$. If $f \notin (G(S) \cup \{\perp\})$ then trivially $\Pr[A(S) = f] = 0 \leq e^{\varepsilon} \Pr[A(S') = f]$. We have already established (in Inequality (4.1)) that $\Pr[A(S) = \perp] \leq e^{\varepsilon/4} \Pr[A(S') = \perp]$. It remains, hence, to deal with the case where $f \in G(S) \cap G(S')$. For this case, we use the following Fact 3, proved below.

Fact 3: $\displaystyle\sum_{h \in G(S')} \exp\left(\frac{\varepsilon}{4} q(S',h)\right) \leq e^{\varepsilon/2} \cdot \sum_{h \in G(S)} \exp\left(\frac{\varepsilon}{4} q(S,h)\right).$

Using Fact 3, for every possible output $f \in G(S) \cap G(S')$ we have that

$$\begin{aligned}
&\frac{\Pr[A(S) = f]}{\Pr[A(S') = f]} \\
&= \left(\Pr[A(S) \neq \perp] \frac{\exp(\frac{\varepsilon}{4} q(f,S))}{\sum_{h \in G(S)} \exp(\frac{\varepsilon}{4} q(h,S))}\right) \Big/ \left(\Pr[A(S') \neq \perp] \frac{\exp(\frac{\varepsilon}{4} q(f,S'))}{\sum_{h \in G(S')} \exp(\frac{\varepsilon}{4} q(h,S'))}\right) \\
&= \frac{\Pr[A(S) \neq \perp]}{\Pr[A(S') \neq \perp]} \cdot \frac{\exp(\frac{\varepsilon}{4} q(f,S)) \cdot \sum_{h \in G(S')} \exp(\frac{\varepsilon}{4} q(h,S'))}{\exp(\frac{\varepsilon}{4} q(f,S')) \cdot \sum_{h \in G(S)} \exp(\frac{\varepsilon}{4} q(h,S))} \leq e^{\frac{\varepsilon}{4}} \cdot e^{\frac{\varepsilon}{4}} \cdot e^{\frac{\varepsilon}{2}} = e^{\varepsilon}.
\end{aligned}$$

We now prove Fact 3. Denote

$$\mathcal{X} \triangleq \sum_{h \in G(S)} \exp\Big(\frac{\varepsilon}{4} q(S,h)\Big).$$

We first show that

$$k \cdot e^{\varepsilon/4} + e^{\varepsilon/4} \cdot \mathcal{X} \leq e^{\varepsilon/2} \mathcal{X}. \tag{4.2}$$

That is, we need to show that $\mathcal{X} \geq k/(e^{\varepsilon/4} - 1)$. As $1 + \varepsilon/4 \leq e^{\varepsilon/4}$, it suffices to show that $\mathcal{X} \geq 4k/\varepsilon$. Recall that there exists a solution $\hat{f}$ s.t. $q(S,\hat{f}) \geq \alpha m/4$. Therefore,

$$\mathcal{X} \geq \exp\Big(\frac{\varepsilon}{4} \frac{\alpha m}{4}\Big),$$

which is at least $4k/\varepsilon$ for

$$m \geq \frac{16}{\alpha\varepsilon} \ln\Big(\frac{4k}{\varepsilon}\Big).$$

This proves (4.2).

Now, recall that as q is k-growth-bounded, for every $h \in \mathcal{F}$ we have $|q(S,h) - q(S',h)| \leq 1$. Moreover, $|G(S') \setminus G(S)| \leq k$, and every $h \in (G(S') \setminus G(S))$ obeys $q(S',h) = 1$. Hence,

$$\begin{aligned}
\sum_{h \in G(S')} \exp\Big(\frac{\varepsilon}{4} q(S',h)\Big) &\leq k \cdot \exp\Big(\frac{\varepsilon}{4}\Big) + \sum_{h \in G(S') \cap G(S)} \exp\Big(\frac{\varepsilon}{4} q(S',h)\Big) \\
&\leq k \cdot \exp\Big(\frac{\varepsilon}{4}\Big) + \exp\Big(\frac{\varepsilon}{4}\Big) \cdot \sum_{h \in G(S') \cap G(S)} \exp\Big(\frac{\varepsilon}{4} q(S,h)\Big) \\
&\leq k \cdot \exp\Big(\frac{\varepsilon}{4}\Big) + \exp\Big(\frac{\varepsilon}{4}\Big) \cdot \sum_{h \in G(S)} \exp\Big(\frac{\varepsilon}{4} q(S,h)\Big) \\
&= k \cdot e^{\varepsilon/4} + e^{\varepsilon/4} \cdot \mathcal{X} \leq e^{\varepsilon/2} \mathcal{X}.
\end{aligned}$$

This concludes the proof of Fact 3, and completes the proof of the lemma. □

The utility analysis for the Choosing Mechanism is rather straightforward:

Lemma 4.5. *When q is a k-bounded-growth quality function, given a database S of*

$$m \geq \frac{16}{\alpha\varepsilon} \ln\Big(\frac{16k}{\alpha\beta\varepsilon\delta}\Big)$$

elements, the Choosing Mechanism outputs an α-good solution for S with probability at least $1 - \beta$.

Proof. Note that if $q(S,f) < \alpha m$ for every solution f, then every solution is an α-good solution, and the mechanism cannot fail. Assume, therefore, that there exists a solution f s.t. $q(f,S) \geq \alpha m$, and recall that the mechanism defines best(S) as $\max_{f \in \mathcal{F}} \{q(f,S)\} + \mathrm{Lap}(4/\varepsilon)$. Now consider the following two good events:

E_1: $\text{best}(S) \geq \dfrac{\alpha m}{2}$.

E_2: The Exponential Mechanism chooses a solution f s.t. $q(S,f) \geq \text{opt}(S) - \alpha m$.

If E_2 occurs then the mechanism outputs an α-good solution. Note that the event E_2 is contained inside the event E_1, and, therefore, $\Pr[E_2] = \Pr[E_1 \wedge E_2] = \Pr[E_1] \cdot \Pr[E_2 \mid E_1]$. By the properties of the Laplace Mechanism,

$$\Pr[E_1] \geq \left(1 - \frac{1}{2}\exp\left(-\frac{\varepsilon}{4}\frac{\alpha m}{2}\right)\right),$$

which is at least $(1-\beta/2)$ for

$$m \geq \frac{8}{\alpha\varepsilon}\ln\left(\frac{1}{\beta}\right).$$

By the growth-boundedness of q, and as S is of size m, there are at most km possible solutions f with $q(f,S) > 0$. That is, $|G(S)| \leq km$. By the properties of the Exponential Mechanism, we have that

$$\Pr[E_2 \mid E_1] \geq \left(1 - km \cdot \exp\left(-\frac{\alpha\varepsilon m}{4}\right)\right),$$

which is at least $(1-\beta/2)$ for

$$m \geq \frac{8}{\alpha\varepsilon}\ln\left(\frac{16k}{\alpha\beta\varepsilon}\right).$$

For our choice of m we have, therefore, that $\Pr[E_2] \geq (1-\beta/2)(1-\beta/2) \geq (1-\beta)$.

All in all, for

$$m \geq \frac{16}{\alpha\varepsilon}\ln\left(\frac{16k}{\alpha\beta\varepsilon\delta}\right)$$

we get that with probability at least $(1-\beta)$ it outputs an α-good solution for its input database. □

4.2 (ε,δ)-private sanitizer for $\texttt{POINT}_d$

Beimel et al. [3] showed that every pure ε-private sanitizer for $\texttt{POINT}_d$, must operate on databases of $\Omega(d)$ elements. In this section we present an (ε,δ)-private sanitizer for $\texttt{POINT}_d$ with sample complexity $O_{\alpha,\beta,\varepsilon,\delta}(1)$. This separates the database size necessary for $(\varepsilon,0)$-private sanitizers from the database size sufficient for (ε,δ)-private sanitizers.

Let $S = (x_1, x_2, \ldots, x_m) \in X_d^m$ be a database of d-bit strings. For every $c_j \in \texttt{POINT}_d$, the query $Q_{c_j} : X_d^* \to [0,1]$ is defined to be the fraction of the strings in the database that equal j

$$Q_{c_j}(S) = \frac{1}{m}|\{i : c_j(x_i) = 1\}| = \frac{1}{m}|\{i : x_i = j\}|.$$

Our sanitizing algorithm invokes the Choosing Mechanism to choose points $x \in X_d$. Consider the following $q : X_d^* \times X_d \to \mathbb{N}$. Given a database $S \in X_d^m$ and a point $x \in X_d$, define $q(S,x)$ to be the number of appearances of x in S. By Example 4.3, q defines a 1-bounded-growth choosing problem. Moreover, given a subset $R \subseteq X_d$ consider the restriction of q to the subset R defined as $q_R(S,x) = q(S,x)$ for $x \in R$ and zero otherwise. The function q_R is a 1-bounded-growth quality function. Our sanitizer SanPoints appears in Figure 9.

Algorithm SanPoints
Inputs: a database $S = (x_1, \ldots, x_m)$, and parameters $\alpha, \beta, \varepsilon, \delta$.

1. Initialize: $\forall x \in X_d$ let $\mathrm{Est}(x) = 0$ and let $R = X_d$.
2. For $i = 1$ to $2/\alpha$
   (a) Choose $b \in R$ using the Choosing Mechanism with quality function q_R, approximation parameter $\alpha/2$, confidence parameter $\alpha\beta/4$, and privacy parameters
   $$\tilde{\varepsilon} \triangleq \frac{\varepsilon}{\sqrt{\frac{32}{\alpha}\ln(\frac{5}{\delta})}} \qquad \text{and} \qquad \tilde{\delta} \triangleq \frac{\alpha\delta}{5}.$$
   (b) If $b \neq \perp$ then let $\mathrm{Est}(b) := Q_{c_b}(S) + \mathrm{Lap}\left(\frac{1}{\tilde{\varepsilon}}\frac{1}{m}\right)$ and let $R = R \setminus \{b\}$.
3. Return $\mathrm{Est}(\cdot)$.

Figure 9: Algorithm SanPoints.

Theorem 4.6. *Fix $\alpha, \beta, \varepsilon, \delta$. For*

$$m \geq O\left(\frac{1}{\alpha^{1.5}\varepsilon}\sqrt{\ln\left(\frac{1}{\delta}\right)}\ln\left(\frac{1}{\alpha\beta\varepsilon\delta}\right)\right),$$

algorithm SanPoints *is an efficient $(\alpha, \beta, \varepsilon, \delta, m)$-improper-sanitizer for* $\mathtt{POINT}_d$.

Proof. We start with the utility analysis. Fix a database $S = (x_1, \ldots, x_m)$, and consider the execution of algorithm SanPoints on S. Denote the element chosen by the Choosing Mechanism on the i^{th} iteration of step 2 by b_i, and denote the set of all such elements as $B = \{b_1, \ldots, b_{2/\alpha}\} \setminus \{\perp\}$. Moreover, let R_i denote the set R as is was at the beginning of the i^{th} iteration. Consider the following two bad events:

E_1. $\exists b \in B$ s.t. $|Q_{c_b}(S) - \mathrm{Est}(b)| > \alpha$.

E_2. $\exists a \notin B$ s.t. $Q_{c_a}(S) > \alpha$.

If none of these two events happen, then algorithm SanPoints succeeds in outputting an estimation Est s.t. $\forall c_j \in \mathtt{POINT}_d \ \ \left|Q_{c_j}(S) - \mathrm{Est}(j)\right| \leq \alpha$. We now bound the probability of both events.

Consider an iteration i in which an element $b_i \neq \perp$ is chosen in step 2a. We have that

$$\Pr[|Q_{c_{b_i}}(S) - \mathrm{Est}(b_i)| > \alpha] = \Pr\left[\left|\mathrm{Lap}\left(\frac{1}{\tilde{\varepsilon}}\frac{1}{m}\right)\right| > \alpha\right] = \exp(-\tilde{\varepsilon}\alpha m).$$

Using the union bound on the number of iterations, we get that

$$\Pr[E_1] \leq \frac{2}{\alpha}\exp(-\tilde{\varepsilon}\alpha m).$$

For

$$m \geq \frac{6}{\alpha^{1.5}\varepsilon} \ln\left(\frac{4}{\alpha\beta}\right) \sqrt{\ln\left(\frac{5}{\delta}\right)}$$

we get that $\Pr[E_1] \leq \beta/2$.

We now bound $\Pr[E_2]$. By the properties of the Choosing Mechanism (Lemma 4.5), with probability at least $(1-\alpha\beta/4)$, an execution of the Choosing Mechanism on step 2a returns an $\alpha/2$-good solution b_i s.t.

$$q_{R_i}(S,b_i) \geq \max_{x \in X_d}\{q_{R_i}(S,x)\} - \frac{\alpha}{2}m. \tag{4.3}$$

Using the union bound on the number of iterations, we get that with probability at least $(1-\beta/2)$, Inequality (4.3) holds for every iteration $1 \leq i \leq \alpha/2$. We will now see that in such a case, event E_2 does not occur. Assume to the contrary that there exists an $a \notin B$ s.t. $Q_{c_a}(S) > \alpha$. Therefore, for every iteration i we have $\max_{x \in X_d}\{q_{R_i}(S,x)\} > \alpha m$ and thus $q_{R_i}(S,b_i) > (\alpha/2)m$. This means that there exist (at least) $2/\alpha$ different points $b_i \in X_d$ that appear in S more than $(\alpha/2)m$ times, which contradicts the fact that the size of S is m.

All in all, $\Pr[E_2] \leq \beta/2$, and the probability of algorithm SanPoints failing to output an estimation Est s.t. $\forall c_j \in \texttt{POINT}_d \ \ \left|Q_{c_j}(S) - \mathrm{Est}(j)\right| \leq \alpha$ is at most β.

We now proceed with the simple privacy analysis. Note that algorithm SanPoints accesses its input database only using the Choosing Mechanism on step 2a and using the Laplace Mechanism on step 2b. Every interaction with the Laplace Mechanism preserves $(\tilde{\varepsilon},0)$-differential privacy, and there exactly $2/\alpha$ such interactions. For our choice of m, every interaction with the Choosing Mechanism preserves $(\tilde{\varepsilon},\tilde{\delta})$-differential privacy, and there are exactly $2/\alpha$ such interactions. Applying Theorem 2.5 (the composition theorem) with our choice of $\tilde{\varepsilon},\tilde{\delta}$, we get that algorithm SanPoints preserves (ε,δ)-differential privacy. □

Algorithm SanPoints can also be used as a sanitizer for the concept class $\texttt{k-POINT}_d$, defined as follows. For every $A \subseteq X_d$ s.t. $|A| = k$, the concept class $\texttt{k-POINT}_d$ contains the concept $c_A : X_d \to \{0,1\}$, defined as $c_A(x) = 1$ if $x \in A$ and $c_A(x) = 0$ otherwise.

Let $S = (x_1, x_2, \ldots, x_m) \in X_d^m$ be a database. For every $c_I \in \texttt{k-POINT}_d$, the query $Q_{c_I} : X_d^* \to [0,1]$ is defined as

$$Q_{c_I}(S) = \frac{1}{m}|\{i : c_I(x_i) = 1\}| = \frac{1}{m}|\{i : x_i \in I\}|.$$

Consider the following algorithm.

Theorem 4.7. *Fix $k, \alpha, \beta, \varepsilon, \delta$. For*

$$m \geq O\left(\frac{k^{1.5}}{\alpha^{1.5}\varepsilon}\sqrt{\ln\left(\frac{1}{\delta}\right)}\ln\left(\frac{k}{\alpha\beta\varepsilon\delta}\right)\right),$$

the algorithm in Figure 10 is an efficient $(\alpha,\beta,\varepsilon,\delta,m)$-improper-sanitizer for $\texttt{k-POINT}_d$.

A Sanitizer for k-POINT$_d$.
Input: parameters $\alpha, \beta, \varepsilon, \delta$ and a dataset S of m elements.

1. Execute SanPoints on $S, \frac{\alpha}{k}, \beta, \varepsilon, \delta$, and denote the returned function as $\mathrm{Est}(\cdot)$.
2. For every $I \subseteq X_d$ s.t. $|I| = k$, define $e(I) = \sum_{i \in I} \mathrm{Est}(i)$.
3. Return $e(\cdot)$.

Figure 10: A Sanitizer for k-POINT$_d$.

Proof. The privacy of the algorithm is immediate. Fix a database $S = (x_1, x_2, \ldots, x_m) \in X_d^m$. By Theorem 4.6, with probability at least $(1-\beta)$, the estimation Est on step 1 is s.t.

$$\forall j \in X_d \quad \left| \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}_{\{x_i = j\}} - \mathrm{Est}(j) \right| \leq \frac{\alpha}{k}.$$

Now fix a set $I \subseteq X_d$ of cardinality k. As $Q_{c_I}(S) = \frac{1}{m}|\{i \,:\, x_i \in I\}|$, we have that

$$\left| Q_{c_I}(S) - \sum_{i \in I} \mathrm{Est}(i) \right| \leq k \frac{\alpha}{k} = \alpha. \qquad \square$$

4.3 (ε, δ)-private sanitizer for THRESH$_d$

Recall that THRESH$_d = \{c_0, \ldots, c_{2^d}\}$, where $c_j(x) = 1$ if and only if $x < j$. Let $S = (x_1, \ldots, x_m) \in X_d^m$ be a database. For every $c_j \in$ THRESH$_d$, the query $Q_{c_j} : X_d^* \to [0,1]$ is defined as

$$Q_{c_j}(S) = \frac{1}{m}|\{i : c_j(x_i) = 1\}| = \frac{1}{m}|\{i : x_i < j\}|.$$

As $|\,\text{THRESH}_d\,| = 2^d + 1$, one can use the generic construction of Blum et al. [7], and get an ε-private sanitizer for this class with sample complexity $O(d)$. By [3], this is the best possible when guaranteeing pure privacy (ignoring the dependency on α, β and ε). We next present a recursive sanitizer for THRESH$_d$, guaranteeing approximated privacy and exhibiting sample complexity $\widetilde{O}_{\alpha,\beta,\varepsilon,\delta}(8^{\log^*(d)})$.

The algorithm maintains its sanitized database $\hat{S}$ as a global variable, which is initialized as the empty set. In addition, for the privacy analysis, we would need a bound on the number of recursive calls. It will be convenient to maintain another global variable, calls, initialized at the desired bound and decreased in every recursive call.

Given a database $S = (x_1, \ldots, x_m) \in X_d^m$, and a subset $R \subseteq X_d$, we denote by $\#_S[R]$ the number of examples $x \in S$ s.t. $x \in R$. That is, $\#_S[R] = |\{i : x_i \in R\}|$. At every recursive call, the algorithm, which is executed on a range $[k, \ell] \subseteq X_d$, identifies an interval $[a,b] \subseteq [k,\ell]$ s.t. $\#_S[a,b]$ is "not too big," but "not too small" either. Then the algorithm estimates $\hat{\#}[a,b] = \#_S[a,b] + \mathrm{Lap}(1/\varepsilon)$, and adds $\hat{\#}[a,b]$ copies of the point b to the constructed sanitized database. Afterwards, the algorithm proceeds recursively on the range

$[k, a-1]$ and on the range $[b+1, \ell]$. As the number of points in [a,b] is "not too small," the depth of this recursion is bounded. Our sanitizer SanThresholds appears in Figure 11. We next prove its properties, starting with the privacy analysis.

Lemma 4.8. *When permitted c recursive calls on a sample S of*

$$m \geq \frac{1024}{\alpha\varepsilon} \ln\Big(\frac{2048}{\alpha\beta\varepsilon\delta}\Big)$$

elements, algorithm SanThresholds preserves $(\tilde{\varepsilon}, 5c\delta)$-differential privacy, where

$$\tilde{\varepsilon} = \sqrt{8c \ln\Big(\frac{1}{c\delta}\Big)}\varepsilon + 8c\varepsilon^2 .$$

Proof. Every iteration of algorithm SanThresholds can access its input database at most twice using the Laplace Mechanism (on steps 2,11), at most once using the Choosing Mechanism (on step 9 or on step 10b), and at most once using algorithm RecConcave (on step 8). By the properties of the Laplace Mechanism, every interaction with it preserves $(\varepsilon, 0)$-differential privacy. Note that the quality function with which we call the Choosing Mechanism is at most 2-growth-bounded. Therefore, as

$$m \geq \frac{1024}{\alpha\varepsilon} \ln\Big(\frac{2048}{\alpha\beta\varepsilon\delta}\Big),$$

every such interaction with the Choosing Mechanism preserves (ε, δ)-differential privacy. Last, for our choice of $\hat{\varepsilon}, \hat{\delta}$, every interaction with algorithm RecConcave preserves (ε, δ)-differential privacy.

That is, throughout its entire execution, algorithm SanThresholds invokes at most $4c$ mechanisms, each (ε, δ)-differentially private. By Theorem 2.5, algorithm SanThresholds is $(\tilde{\varepsilon}, 5c\delta)$-differential private, where

$$\tilde{\varepsilon} = \sqrt{8c \ln\Big(\frac{1}{c\delta}\Big)}\varepsilon + 8c\varepsilon^2 .$$

□

We start the utility analysis of SanThresholds with the following simple claim.

Claim 4.9. *The function $Q(S, \cdot)$, defined on step 6, is quasi-concave.*

Proof. First note that the function $I(S, \cdot)$ defined on step 5 is non-decreasing. Now, let $u \leq v \leq w$ be s.t. $Q(S,u), Q(S,w) \geq x$. That is,

$$I(S,u) - \frac{\alpha m}{32} \geq x, \qquad \frac{3\alpha m}{32} - I(S,u-1) \geq x, \qquad \text{and} \qquad I(S,w) - \frac{\alpha m}{32} \geq x, \qquad \frac{3\alpha m}{32} - I(S,w-1) \geq x.$$

Using the fact that $I(S, \cdot)$ is non-decreasing, we have that $I(S,u) \leq I(S,v)$ and that $I(S,v-1) \leq I(S,w-1)$. Therefore

$$I(S,v) - \frac{\alpha m}{32} \geq I(S,u) - \frac{\alpha m}{32} \geq x,$$
$$\frac{3\alpha m}{32} - I(S,v-1) \geq \frac{3\alpha m}{32} - I(S,w-1) \geq x,$$

and $Q(S,v) \geq x$. □

Algorithm SanThresholds
Input: a range $[k,\ell]$, parameters $\alpha,\beta,\varepsilon,\delta$, and a dataset S of m elements.
Global variables: the sanitized database $\hat{S}$ (initially empty) and calls (initialized to $77/\alpha$).

1. If calls $= 0$ then halt. Otherwise, set calls := calls -1.
2. Compute $\hat{\#}[k,\ell] = \#_S[k,\ell] + \mathrm{Lap}(1/\varepsilon)$.
3. If $\hat{\#}[k,\ell] < \alpha m/8$ then define $[a,b] := [k,\ell]$, add $\hat{\#}[a,b]$ copies of the point b to $\hat{S}$, and halt.
4. Let T be the smallest power of 2 s.t. $T \geq (\ell - k + 1)$.
5. For every $0 \leq j \leq \log(T)$, define $I(S,j) = \max_{\substack{[x,y]\subseteq[k,\ell] \\ y-x+1\leq 2^j}} \left\{ \#_S[x,y] \right\}$.
   % Every interval $[x,y] \subseteq [k,\ell]$ of length 2^j contains at most $I(S,j)$ points in S, and there exists at least one interval of length (at most) 2^j containing exactly $I(S,j)$ points.
6. For every $0 \leq j \leq \log(T)$, define its quality $Q(S,j)$ as
$$Q(S,j) = \min\left\{ I(S,j) - \frac{\alpha m}{32}, \frac{3\alpha m}{32} - I(S,j-1) \right\} \qquad \text{where} \qquad I(S,-1) \triangleq 0$$
   % If $Q(S,j)$ is high (for some j), then there exists an interval $[a,b]$ of length 2^j containing significantly more points than $\frac{\alpha m}{32}$, and every interval of length $\frac{1}{2}2^j$ contains significantly less points than $\frac{3\alpha m}{32}$.
7. Define $r = \alpha m/32$.
8. Execute algorithm RecConcave on the range $[0,\log(T)]$, the quality function $Q(\cdot,\cdot)$, the quality promise r, accuracy parameter $1/4$, and privacy parameters
$$\hat{\varepsilon} = \frac{\varepsilon}{3\log^*(d)} \qquad \text{and} \qquad \hat{\delta} = \frac{\delta}{3\log^*(d)}.$$
   Denote the returned value as z, and let $Z = 2^z$.
   % Assuming the recursive call was successful, the returned z is s.t. $Q(S,z) \geq (1-\frac{1}{4})r = \frac{3\alpha m}{128}$. That is, $I(S,z) \geq \frac{7\alpha m}{128}$ and $I(S,z-1) \leq \frac{9\alpha m}{128}$.
9. If $z = 0$ then Choose $b \in [k,\ell]$ using the Choosing Mechanism with parameters $\alpha/64,\beta,\varepsilon,\delta$ and the quality function $\#_S[\cdot]$. Denote $[a,b] = [b,b]$.
   % That is, the quality of $b \in [k,\ell]$ is the number of appearances of the point b is S, and the Choosing Mechanism chooses a frequent point.

Figure 11: Sanitizer SanThresholds for $\mathtt{THRESH}_d$.

Algorithm SanThresholds (continued)

10. Otherwise (if $z \geq 1$) then
    (a) Divide $[k,\ell]$ into the following intervals of length $2Z$ (the last ones might be trimmed):

$$A_1 = [k, k+2Z-1],\ A_2 = [k+2Z, k+4Z-1],\ A_3 = [k+4Z, k+6Z-1], \ldots$$
$$B_1 = [k+Z, k+3Z-1],\ B_2 = [k+3Z, k+5Z-1],\ B_3 = [k+5Z, k+7Z-1], \ldots$$

    (b) Choose $[a,b] \in \{A_i\} \cup \{B_i\}$ using the Choosing Mechanism with parameters $\frac{\alpha}{64}, \beta, \varepsilon, \delta$ and the quality function $\#_S[\cdot]$.
        % The union of the A_i and the B_i causes the quality function $\#_S[\cdot]$ to be 2-growth-bounded.
11. Compute $\hat{\#}[a,b] = \#_S[a,b] + \mathrm{Lap}(1/\varepsilon)$, and add $\hat{\#}[a,b]$ copies of the point b to $\hat{S}$.
12. If $a > k$, then execute SanThresholds recursively on the range $[k, a-1]$, the parameters $\alpha, \beta, \varepsilon, \delta$, the database S, and the references to $\hat{S}$ and to calls.
13. If $b < \ell$, then execute SanThresholds recursively on the range $[b+1, \ell]$, the parameters $\alpha, \beta, \varepsilon, \delta$, the database S, and the references to $\hat{S}$ and to calls.

Figure 11: Sanitizer SanThresholds for $\mathtt{THRESH}_d$, continued.

Note that every iteration of algorithm SanThresholds draws at most 2 random samples (on steps 2 and 11) from $\mathrm{Lap}(1/\varepsilon)$. We now proceed with the utility analysis by identifying 3 good events that occur with high probability (over the coin tosses of the algorithm).

Claim 4.10. *Fix $\alpha, \beta, \varepsilon, \delta$. Let SanThresholds be executed with* calls *initialized to $c \geq 77/\alpha$, and on a database S of*

$$m \geq 8^{\log^*(d)} \cdot \frac{60c}{\alpha\varepsilon} \log^*(d) \log\Big(\frac{12\log^*(d)}{\beta\varepsilon\delta}\Big)$$

elements. With probability at least $(1-3c\beta)$ the following 3 events happen:

B_1 : *In every random draw of $\mathrm{Lap}(1/\varepsilon)$ throughout the execution of SanThresholds we have*

$$\left|\mathrm{Lap}\Big(\frac{1}{\varepsilon}\Big)\right| \leq \frac{\alpha m}{16c}.$$

B_2 : *Every interaction with algorithm RecConcave on step 8 succeeds in returning a value z s.t.*

$$Q(S,z) \geq \frac{3\alpha m}{128}.$$

B_3 : *Every iteration that halts after step 13, defines an interval $[a,b]$ s.t.*

$$\#_S[a,b] \geq \frac{5\alpha m}{128}.$$

Proof. First note that it suffices to lower bound the terms $\Pr[B_1]$, $\Pr[B_2 \mid B_1]$, and $\Pr[B_3 \mid B_1 \wedge B_2]$, as by the chain rule of conditional probability we have

$$\Pr[B_1 \wedge B_2 \wedge B_3] = \Pr[B_1] \cdot \Pr[B_2 \mid B_1] \cdot \Pr[B_3 \mid B_1 \wedge B_2] .$$

We now bound each of those terms, starting with $\Pr[B_1]$. In every single draw, the probability of $|\mathrm{Lap}(1/\varepsilon)| > \alpha m/(16c)$ is at most $\exp(-\alpha\varepsilon m/(16c))$, which is at most $\beta/2$ for

$$m \geq \frac{16c}{\alpha\varepsilon} \ln\Big(\frac{2}{\beta}\Big) .$$

As c (the initial value of calls) limits the number of iterations, we get that $\Pr[B_1] \geq (1 - c\beta)$.

For the analysis of $\Pr[B_2 \mid B_1]$, consider an iteration of algorithm SanThresholds that executes RecConcave on step 8. In particular, this iteration passed step 3 and $\hat{\#}[k,\ell] \geq \alpha m/8$. As event B_1 has occurred, we have that $\#_S[k,\ell] \geq \alpha m/16$. Recall that (by definition) $I(S,-1) = 0$, and so, there exists a $j \in [0, \log(T)]$ s.t. $I(S,j) \geq \alpha m/16$, and $I(S,j-1) < \alpha m/16$. Plugging those inequalities in the definition of $Q(\cdot,\cdot)$, for this j we have that $Q(S,j) \geq \alpha m/32 = r$, and the quality promise used to execute algorithm RecConcave is valid. Moreover, the function $Q(S,\cdot)$ defined on step 6 is quasi-concave (by Claim 4.9). And so, for

$$m \geq 8^{\log^*(d)} \cdot \frac{4608}{\alpha\varepsilon} \log^*(d) \log\Big(\frac{12\log^*(d)}{\beta\delta}\Big) ,$$

algorithm RecConcave ensures that with probability at least $(1-\beta)$, the returned z is s.t. $Q(S,z) \geq (1-14)r = 3\alpha m/128$. As there are at most c iterations, $\Pr[B_2 \mid B_1] \geq (1 - c\beta)$.

For the analysis of $\Pr[B_3 \mid B_1 \wedge B_2]$, consider an iteration of algorithm SanThresholds that halts after step 13, and let z be the value returned by algorithm RecConcave on step 8. As event B_2 has occurred, $Q(S,z) \geq 3\alpha m/128$. In particular, $I(S,z) \geq 7\alpha m/128$, and there exists an interval $G \subseteq [k,\ell]$ of length (at most) 2^z containing at least $7\alpha m/128$ points in S. Assume $z > 0$, and consider the intervals in $\{A_i\}$ and $\{B_i\}$ defined on Step 10a. As those intervals are of length $2 \cdot 2^z$, and the $\{B_i\}$ are shifted by 2^z, there must exist an interval $C \in \{A_i\} \cup \{B_i\}$ s.t. $G \subseteq C$. The quality of C is at least $\#_S[C] \geq 7\alpha m/128$. Moreover, this quality function $\#[\cdot]$ over $\{A_i\} \cup \{B_i\}$ is 2-bounded. Therefore, as

$$m \geq \frac{1024}{\alpha\varepsilon} \ln\Big(\frac{2048}{\alpha\beta\varepsilon\delta}\Big) ,$$

with probability at least $(1-\beta)$, the Choosing Mechanism returns an interval $[a,b]$ s.t.

$$\#_S[a,b] \geq \frac{7\alpha m}{128} - \frac{\alpha m}{64} = \frac{5\alpha m}{128} .$$

This also holds when $z = 0$ (on Step 9).

So, given that event $(B_1 \wedge B_2)$ has occurred, in every iteration that halts after step 13, the probability of defining $[a,b]$ s.t. $\#_S[a,b] < 5\alpha m/128$ is at most β. As there are at most c iterations, we see that $\Pr[B_3 \mid B_1 \wedge B_2] \geq (1 - c\beta)$.

All in all, for $c\beta \leq 3$ we get that

$$\Pr[B_1 \wedge B_2 \wedge B_3] \geq (1-c\beta)(1-c\beta)(1-c\beta) \geq 1 - 3c\beta . \qquad \square$$

Every iteration of algorithm SanThresholds that does not halt on step 1 defines an interval $[a,b]$ (on exactly one of the steps 3,9,10b). This interval $[a,b]$ is not part of any range that is given as input to any future recursive call. Moreover, if none of the recursive calls throughout the execution of SanThresholds halts on step 1, these $[a,b]$ intervals form a partition of the initial range. We now proceed with the utility analysis by identifying yet another 3 good events (at a somewhat higher level) that occur whenever $(B_1 \wedge B_2 \wedge B_3)$ occur.

Claim 4.11. *Fix $\alpha,\beta,\varepsilon,\delta$. Let SanThresholds be executed with* calls *initialized to $c \geq 77/\alpha$, and on a database S of*

$$m \geq 8^{\log^*(d)} \cdot \frac{60c}{\alpha\varepsilon} \log^*(d) \log\Big(\frac{12\log^*(d)}{\beta\varepsilon\delta}\Big)$$

elements. With probability at least $(1-3c\beta)$ the following 3 events happen:

E_1 : *There are at most $77/\alpha$ recursive calls, none of them halts on the first step.*

E_2 : *Every iteration defines $[a,b]$ s.t. $\#_S[a,b-1] \leq \alpha m/2$. That is, every iteration defines $[a,b]$ s.t. the interval $[a,b-1]$ contains at most $\alpha m/2$ points in S.*

E_3 : *In every iteration $\left|\#_S[a,b] - \hat{\#}[a,b]\right| \leq \frac{\alpha m}{4}\frac{\alpha}{77}$.*

Proof. Consider again events B_1,B_2,B_3 defined in Claim 4.10. We will show that the event $(E_1 \wedge E_2 \wedge E_3)$ is implied by $(B_1 \wedge B_2 \wedge B_3)$ (which happens with probability at least $(1-3c\beta)$ by Claim 4.10). We, therefore, continue the proof assuming that $(B_1 \wedge B_2 \wedge B_3)$ has occurred.

We begin by showing that event E_1 occurs. Denote the number of iterations that halts on steps 1-3 as y_1, and the number of complete iterations (i. e., that halts after step 13) as y_2. Clearly, $y_1 \leq 2y_2$. Now, as event B_3 has occurred, we have that every iteration that halts after step 13 defines an interval $[a,b]$ s.t. $\#_S[a,b] \geq 5\alpha m/128$. This interval does not intersect any range given as input to future calls, and, therefore, $y_2 \leq 128/(5\alpha)$. The total number of iterations is, therefore, bounded by $3y_2 \leq 384/(5\alpha) < 77/\alpha$. Thus, whenever calls is initialized to at least $77/\alpha$, there are at most $77/\alpha$ iterations, none of them halts on step 1. That is, E_1 occurs.

We next show that E_3 occurs. As we have seen, event B_3 ensures that no iteration halts on step 1. Therefore every iteration defines $\hat{\#}[a,b]$ by adding a random draw of $\mathrm{Lap}(1/\varepsilon)$ to $\#_S[a,b]$. As event B_1 has occurred, we have that

$$\left|\#_S[a,b] - \hat{\#}[a,b]\right| \leq \frac{\alpha m}{16c} \leq \frac{\alpha m}{4}\frac{\alpha}{77}.$$

So, E_3 occurs.

It remains to show that E_2 occurs. As B_3 has occurred, no iteration of algorithm SanThresholds halts on step 1. In particular, every iteration defines $[a,b]$ on exactly one of the steps 3,9,10b. Consider an iteration of algorithm SanThresholds that defines $[a,b]$ on step 3. In that iteration, $\hat{\#}[k,\ell] < \alpha m/8$. As event B_1 has occurred, we have that $\#_S[k,\ell] \leq \alpha m/2$. Therefore the interval $[a,b-1] = [k,\ell-1]$ contains at most $\alpha m/2$ points.

Consider an iteration of algorithm SanThresholds that defines $[a,b]$ on step 9. In that iteration, $[a,b]$ is defined as $[a,a]$. Trivially, the empty interval $[a,b-1] = [a,a-1]$ contains at most $\alpha m/2$ points.

Consider an iteration of algorithm SanThresholds that defines $[a,b]$ on step 10b (of length at most $2\cdot 2^z$). As event B_2 has occurred, z is s.t. $Q(S,z)\geq 3\alpha m/128$. In particular $L(S,z-1)\leq 9\alpha m/128$, and every interval of length $(1/2)2^z$ contains at most $9\alpha m/128$ points in S. Therefore $\#_S[a,b-1]\leq 49\alpha m/128\leq \alpha m/2$. Note that we needed z to be at least 1 (ensured by the If condition on step 10), as otherwise the constraint on intervals of length $(1/2)2^z$ has no meaning.

At any case, we have that E_2 must occur.

All in all,

$$\Pr[E_1\wedge E_2\wedge E_3]\geq \Pr[B_1\wedge B_2\wedge B_3]\geq (1-3c\beta)\,. \qquad \square$$

We will now complete the utility analysis by showing that the input database S and the sanitized database $\hat{S}$ (at the end of SanThresholds' execution) are α-close whenever $(E_1\wedge E_2\wedge E_3)$ occurs.

Lemma 4.12. *Fix $\alpha,\beta,\varepsilon,\delta$. Let SanThresholds be executed on the range X_d, a global variable* calls *initialize to $c\geq 77/\alpha$, and on a database S of*

$$m\geq 8^{\log^*(d)}\cdot\frac{60c}{\alpha\varepsilon}\log^*(d)\log\Big(\frac{12\log^*(d)}{\beta\varepsilon\delta}\Big)$$

elements. With probability at least $(1-3c\beta)$, the sanitized database $\hat{S}$ at the end of the execution is s.t. $|Q_{c_j}(S)-Q_{c_j}(\hat{S})|\leq\alpha$ for every $c_j\in\mathtt{THRESH}_d$.

Proof. Denote $S=(x_1,\ldots,x_m)$, and $\hat{S}=(\hat{x_1},\ldots,\hat{x_n})$. Note that $|S|=m$ and that $|\hat{S}|=n$. By Claim 4.11, the event $E_1\cap E_2\cap E_3$ occurs with probability at least $(1-3c\beta)$. We will show that in such a case, the sanitized database $\hat{S}$ is s.t. $|Q_{c_j}(S)-Q_{c_j}(\hat{S})|\leq\alpha$ for every $c_j\in\mathtt{THRESH}_d$.

As event E_1 has occurred, the intervals $[a,b]$ defined throughout the execution of SanThresholds defines a partition of the domain X_d. Denote those intervals as $[a_1,b_1]$, $[a_2,b_2]$, …, $[a_w,b_w]$, where $a_1=0$, $b_w=2^d-1$, and $a_{i+1}=b_i+1$. Now fix some $c_j\in\mathtt{THRESH}_d$, and let t be s.t. $j\in[a_t,b_t]$. We have that

$$Q_{c_j}(S)=\frac{1}{m}\#_S[0,j-1]=\frac{1}{m}\left(\#_S[a_t,j-1]+\sum_{i=1}^{t-1}\#_S[a_i,b_i]\right).$$

As event $E_2\cap E_3$ has occurred,

$$Q_{c_j}(S)\leq\frac{1}{m}\left(\frac{\alpha m}{2}+\sum_{i=1}^{t-1}\Big[\hat{\#}[a_i,b_i]+\frac{\alpha m}{4}\frac{\alpha}{77}\Big]\right).$$

As event E_1 has occurred, $t\leq 77/\alpha$, and

$$Q_{c_j}(S)\leq\frac{\alpha}{2}+\frac{\alpha}{4}+\frac{1}{m}\sum_{i=1}^{t-1}\hat{\#}[a_i,b_i]=\frac{3\alpha}{4}+\frac{1}{m}\#_{\hat{S}}[0,j-1]\,.$$

Similar arguments show that

$$Q_{c_j}(S)\geq-\frac{3\alpha}{4}+\frac{1}{m}\#_{\hat{S}}[0,j-1]\,,\qquad\text{and so}\qquad\left|Q_{c_j}(S)-\frac{1}{m}\#_{\hat{S}}[0,j-1]\right|\leq\frac{3\alpha}{4}\,.$$

Recall that the sanitized database $\hat{S}$ is of size n, and that

$$Q_{c_j}(\hat{S}) = \frac{1}{n}\#_{\hat{S}}[0, j-1]\,.$$

As event $(E_1 \cap E_3)$ has occurred, we have that

$$n \leq m + \frac{\alpha m}{4} = \left(1 + \frac{\alpha}{4}\right)m\,.$$

Therefore,

$$\frac{\#_{\hat{S}}[0, j-1]}{m} - \frac{\#_{\hat{S}}[0, j-1]}{n} = \left(\frac{1}{m} - \frac{1}{n}\right)\#_{\hat{S}}[0, j-1] \leq \frac{\alpha}{4n}\#_{\hat{S}}[0, j-1] \leq \frac{\alpha}{4}.$$

Similar arguments show that

$$\left|\frac{1}{m}\#_{\hat{S}}[0, j-1] - \frac{1}{n}\#_{\hat{S}}[0, j-1]\right| \leq \frac{\alpha}{4}\,.$$

By the triangle inequality we have therefore that

$$\left|Q_{c_j}(S) - Q_{c_j}(\hat{S})\right| \leq \frac{3\alpha}{4} + \frac{\alpha}{4} = \alpha\,. \qquad \square$$

The following theorem is an immediate consequence of Lemma 4.12 and Lemma 4.8.

Theorem 4.13. *Fix $\alpha, \beta, \varepsilon, \delta$. There exists an efficient $(\alpha, \beta, \varepsilon, \delta, m)$-sanitizer for $\texttt{THRESH}_d$, where*

$$m = O\left(8^{\log^*(d)} \cdot \frac{\log^*(d)}{\alpha^{2.5}\varepsilon} \log\left(\frac{\log^*(d)}{\alpha\beta\varepsilon\delta}\right)\sqrt{\log\left(\frac{1}{\alpha\delta}\right)}\right).$$

4.4 Sanitization with pure privacy

Here we give a general lower bound on the database size of pure private sanitizers. Beimel et al. [3] showed that every pure ε-private sanitizer for $\texttt{POINT}_d$ must operate on databases of $\Omega(d)$ elements. With slight modifications, their proof technique can yield a much more general result.

Definition 4.14. Given a concept class C over a domain X, we denote the *effective* size of X w.r.t. C as

$$X_C = \max\left\{|\widetilde{X}| \;:\; \widetilde{X} \subseteq X \;\text{ s.t. }\; \forall x_1 \neq x_2 \in \widetilde{X}\;\; \exists f \in C \;\text{ s.t. }\; f(x_1) \neq f(x_2)\right\}.$$

That is, X_C is the cardinality of the biggest subset $\widetilde{X} \subseteq X$ s.t. every two different elements of $\widetilde{X}$ are labeled differently by at least one concept in C.

Lemma 4.15. *Let C be a concept class over a domain X. For every $(\alpha, \beta, \varepsilon, m)$-sanitizer for C (proper or improper) we have that*

$$m = \Omega\left(\frac{1}{\varepsilon\alpha}\left(\log X_C + \log(1/\beta)\right)\right).$$

Proof. Let $\widetilde{X} \subseteq X$ be s.t. $|\widetilde{X}| = X_C$ and every two different elements of $\widetilde{X}$ are labeled differently by at least one concept in C. Fix some $x_1 \in \widetilde{X}$, and for every $x_i \in \widetilde{X}$, construct a database $S_i \in \widetilde{X}^m$ by setting $(1-3\alpha)m$ entries as x_1 and the remaining $3\alpha m$ entries as x_i (for $i = 1$ all entries of S_1 are x_1). Note that for all $i \neq j$, databases S_i and S_j differ on $3\alpha m$ entries.

Let $\mathbb{S}_i$ be the set of all databases that are α-close to S_i. That is,

$$\mathbb{S}_i = \left\{ \widehat{S} \in X^* \ : \ \forall c \in C \text{ we have } |Q_c(\widehat{S}) - Q_c(S_i)| \leq \alpha \right\}.$$

For every $i \neq j$ we have that $\widehat{\mathbb{S}}_i \cap \widehat{\mathbb{S}}_j = \emptyset$. To see this, let $f \in C$ be s.t. $f(x_i) \neq f(x_j)$ (such a concept exists, by the definition of $\widetilde{X}$). For this f we have $|Q_f(S_i) - Q_f(S_j)| = 3\alpha$. Therefore (by the triangle inequality), there cannot exist a database $\widehat{S}$ for which $|Q_f(S_i) - Q_f(\widehat{S})| \leq \alpha$ and $|Q_f(S_j) - Q_f(\widehat{S})| \leq \alpha$.

Let A be an $(\alpha, \beta, \varepsilon, m)$-sanitizer for C. Without loss of generality, we can assume that A is a *proper* sanitizer (otherwise, we could transform it into a proper one by replacing α with 2α). See Remark 2.18.

For all i, on input S_i the mechanism A should pick an output from $\mathbb{S}_i$ with probability at least $1-\beta$. Hence,

$$\begin{aligned}
\beta &\geq \Pr[A(S_1) \notin \mathbb{S}_1] \\
&\geq \Pr\left[A(S_1) \in \bigcup_{i \neq 1} \mathbb{S}_i \right] \\
&= \sum_{i \neq 1} \Pr[A(S_1) \in \mathbb{S}_i] && \text{(the sets } \mathbb{S}_i \text{ are disjoint)} \\
&\geq \sum_{i \neq 1} \exp(-3\varepsilon\alpha m) \Pr[A(S_i) \in \mathbb{S}_i] && \text{(by the differential privacy of } \mathsf{A}) \\
&\geq (X_C - 1)\exp(-3\varepsilon\alpha m) \cdot (1-\beta).
\end{aligned}$$

Solving for m, we get that

$$m = \Omega\left(\frac{1}{\varepsilon\alpha} \left(\log X_C + \log(1/\beta) \right) \right). \qquad \square$$

Lemma 4.15, together with a lower bound from [7], yields the following result:

Theorem 4.16. *Let C be a concept class over a domain X. If A is an $(1/8, 1/8, 1/2, m)$-sanitizer for C, then $m = \Omega(\log(X_C) + \mathrm{VC}(C))$.*

Proof. Immediate from Lemma 4.15 and Theorem 2.20. $\square$

The lower bound in Theorem 4.16 is the best possible general lower bound in terms of X_C and $\mathrm{VC}(C)$ (up to a factor of $\log \mathrm{VC}(C)$). To see this, let $n < d$, and consider a concept class over X_d containing the following two kinds of concepts. The first kind are 2^n concepts shattering the left n points of X_d (and zero everywhere else). The second kind are $(2^d - n)$ "point concepts" over the right $(2^d - n)$ points of X_d (and zero on the first n). Formally, for every $j = (j_0, j_1, \ldots, j_{n-1}) \in \{0,1\}^n$, let $c_j : X_d \to \{0,1\}$ be defines as $c_j(x) = j_x$ if $x < n$ and $c_j(x) = 0$ otherwise. Define the concept class $C_L = \{c_j\}_{j \in X_n}$. For every $n \leq j < 2^d$, define $f_j : X_d \to \{0,1\}$ as $f_j(x) = 1$ if $x = j$ and $f_j(x) = 0$ otherwise. Define the concept class $C_R = \{f_j\}_{n \leq j < 2^d}$. Now define $C = C_L \bigcup C_R$.

We can now construct a sanitizer for C by applying the generic construction of [7] separately for C_L and for C_R. Given a database S, this will result in two sanitized databases $\widehat{S}_L, \widehat{S}_R$, with which we can answer all queries in the class C – a query for $c \in C_L$ is answered using $\widehat{S}_L$, and a query for $f \in C_R$ is answered using $\widehat{S}_R$. The described (improper) sanitizer for C is of sample complexity $O_{\alpha,\beta,\varepsilon}(\log(X_C) + \mathrm{VC}(C)\log \mathrm{VC}(C))$.

5 Sanitization and proper private PAC

Similar techniques are used for both data sanitization and private learning, suggesting relationships between the two tasks. 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. A similar reduction was given by Gupta et al. [18], where it is stated in terms of statistical queries. They showed that the existence of a sanitizer that accesses the database using at most k statistical queries, implies the existence of a learner that makes at most $2k$ statistical queries. We complement their proof and add the necessary details in order to show that the existence of an *arbitrary* sanitizer (that is not restricted to access its data via statistical queries) implies the existence of a private learner.

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

5.1 Sanitization implies proper PPAC

We show that sanitization of a class C implies private learning of C. Consider an input labeled sample $S = (x_i, y_i)_{i=1}^m \in (X \times \{0,1\})^m$, labeled by some concept $c \in C$. The key observation is that in order to privately output a good hypothesis it is suffices to first produce a sanitization $\hat{S}$ of S (w.r.t. a slightly different concept class C^{label}, to be defined) and then to output a hypothesis $h \in C$ that minimizes the empirical error *over the sanitized database* $\hat{S}$. To complete the proof we then show that sanitization for C implies sanitization for C^{label}.

In order for the chosen hypothesis h to have small *generalization* error (rather then just small empirical error), our input database S must contain at least

$$\frac{\mathrm{VC}(C)}{\alpha^2}\log\Big(\frac{1}{\alpha\beta}\Big)$$

elements. We therefore start with the following simple (technical) lemma, handling a case where our initial sanitizer operates only on smaller databases.

Lemma 5.1. *If there exists an $(\alpha,\beta,\varepsilon,m)$-sanitizer for a class C, then for every $q \in \mathbb{N}$ s.t. $q \geq (18/\beta)\ln(1/\beta)$ there exists a $((2\alpha+2\beta),\beta,\varepsilon,qm)$-sanitizer for C.*

Proof. Fix $q \in \mathbb{N}$ and let A be an $(\alpha,\beta,\varepsilon,m)$-sanitizer for a class C over a domain X. Note that by Theorem 2.21, there exists a $(2\alpha,(3/2)\beta,\varepsilon,m)$-sanitizer A' s.t. the sanitized databases returned by A' are

always of fixed sized

$$n = O\left(\frac{\mathrm{VC}(C)}{\alpha^2}\log\left(\frac{1}{\alpha\beta}\right)\right).$$

We now construct a $((2\alpha+2\beta),\beta,\varepsilon,qm)$-sanitizer B as follows.

> Inputs: a database $S=(z_1,z_2,\ldots,z_{qm})\in(X)^{qm}$
> 1. Partition S into $S_1=(z_i)_{i=1}^{m}$, $S_2=(z_i)_{i=m+1}^{2m}$, $\ldots,S_q=(z_i)_{i=qm-m+1}^{qm}$.
> 2. For every $1\le i\le q$, $\hat{S}_i\leftarrow \mathsf{A}'(S_i)$.
> 3. Output $\hat{S}=\langle\hat{S}_1,\hat{S}_2,\ldots,\hat{S}_q\rangle$.

As A' is ε-differentially private, so is B. Denote $\hat{S}=(\hat{z}_1,\hat{z}_1,\ldots,\hat{z}_{qn})\in(X)^{qn}$. Recall that

$$q\ge\frac{18}{\beta}\ln\left(\frac{1}{\beta}\right),$$

and, hence, using the Chernoff bound, with probability at least $(1-\beta)$ we have that at least $(1-2\beta)q$ of the $\hat{S}_i$ are 2α-good for their matching S_i. In such a case $\hat{S}$ is $(2\alpha+2\beta)$-good for S: for every $f\in C$ we have that

$$\begin{aligned}
Q_f(S) &= \frac{1}{qm}\left|\left\{i:\ \begin{array}{c}1\le i\le qm\\ f(z_i)=1\end{array}\right\}\right| \\
&= \frac{1}{qm}\left|\left\{i:\ \begin{array}{c}1\le i\le m\\ f(z_i)=1\end{array}\right\}\right|+\cdots+\frac{1}{qm}\left|\left\{i:\ \begin{array}{c}qm-m+1\le i\le qm\\ f(z_i)=1\end{array}\right\}\right| \\
&= \frac{1}{q}\left[Q_f(S_1)+\cdots+Q_f(S_q)\right].
\end{aligned}$$

As at least $(1-2\beta)q$ of the $\hat{S}_i$ are 2α-good for their matching S_i, and as trivially $Q_f(S_i)\le 1$ for each database $\hat{S}_i$ that is not 2α-good,

$$\begin{aligned}
Q_f(S) &\le \frac{1}{q}\left[Q_f(\hat{S}_1)+\cdots+Q_f(\hat{S}_q)+(1-2\beta)q2\alpha+2\beta q\right] \\
&\le \frac{1}{q}\left[Q_f(\hat{S}_1)+\cdots+Q_f(\hat{S}_{t/m})\right]+(2\alpha+2\beta) \\
&= \frac{1}{q}\left[\frac{1}{n}\left|\left\{i:\ \begin{array}{c}1\le i\le n\\ f(\hat{z}_i)=1\end{array}\right\}\right|+\cdots+\frac{1}{n}\left|\left\{i:\ \begin{array}{c}qn-n+1\le i\le qn\\ f(\hat{z}_i)=1\end{array}\right\}\right|\right]+(2\alpha+2\beta) \\
&= \frac{1}{qn}\left|\left\{i:\ \begin{array}{c}1\le i\le qn\\ f(\hat{z}_i)=1\end{array}\right\}\right|+(2\alpha+2\beta) \\
&= Q_f(\hat{S})+(2\alpha+2\beta).
\end{aligned}$$

Similar arguments show that $Q_f(S)\ge Q_f(\hat{S})-(2\alpha+2\beta)$.

Algorithm B is, therefore, a $((2\alpha+2\beta),\beta,\varepsilon,qm)$-sanitizer for C, as required. □

As mentioned above, our first step in showing that sanitization for a class C implies private learning for C is to show that privately learning C is implied by sanitization for the slightly modified class C^{label}, defined as follows. For a given predicate c over X_d, we define the predicate c^{label} over X_{d+1} as

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

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

Claim 5.2. $\text{VC}(C) \leq \text{VC}(C^{\text{label}}) \leq 2 \cdot \text{VC}(C)$.

Proof. For the first inequality notice that if a set $S \subseteq X_d$ is shattered by C then the set $S \circ 0$ is shattered by C^{label}. For the second inequality, assume $S \subseteq X_{d+1}$ is shattered by C^{label}. Consider the partition of S to S_0 and S_1, where $S_\sigma = \{\vec{x} \circ y \in S : y = \sigma\}$. For at least one $\sigma \in \{0,1\}$, we have $|S_\sigma| \geq |S|/2$. Hence, the set $\hat{S} = \{\vec{x} : \vec{x} \cdot \sigma \in S_\sigma\}$ is shattered by C and $\text{VC}(C^{\text{label}}) \leq 2 \cdot |\hat{S}| \leq 2 \cdot \text{VC}(C)$. □

The next lemma shows that for every concept class C, a sanitizer for C^{label} implies a private learner for C. In the next lemma, this connection is made under the assumption that the given sanitizer operates on large enough databases. This assumption will be removed in the lemma that follows.

Lemma 5.3. *Let C be a class of predicates over X_d. If there exists an $(\alpha, \beta, \varepsilon, m)$-sanitizer A for C^{label}, where*

$$m \geq \frac{50\,\text{VC}(C)}{\gamma^2} \ln\left(\frac{1}{\gamma\beta}\right)$$

for some $\gamma > 0$, then there exists a proper $((2\alpha+\gamma), 2\beta, \varepsilon, m)$-PPAC learner for C.

Proof. Let A be an $(\alpha, \beta, \varepsilon, m)$-sanitizer, and consider the following algorithm Learn:

> Inputs: a database $S = (x_i, y_i)_{i=1}^m$
> 1. $\hat{S} \leftarrow A(S)$.
> 2. Output $c \in C$ minimizing $\text{error}_{\hat{S}}(c)$.

As A is ε-differentially private, so is Learn. For the utility analysis, fix some target concept $c_t \in C$ and a distribution $\mathcal{D}$ over X_d, and define the following two good events:

$$E_1 : \forall h \in C, \ \left|\text{error}_S(h) - \text{error}_{\hat{S}}(h)\right| \leq \alpha.$$

$$E_2 : \forall h \in C, \ |\text{error}_{\mathcal{D}}(h, c_t) - \text{error}_S(h)| \leq \gamma.$$

We first show that if these 2 good events happen, algorithm Learn returns a $(2\alpha+\gamma)$-good hypothesis. As the target concept satisfies $\text{error}_S(c_t) = 0$, event E_1 ensures the existence of a concept $f \in C$ s.t. $\text{error}_{\hat{S}}(f) \leq \alpha$. Thus, algorithm Learn chooses a hypothesis $h \in C$ s.t. $\text{error}_{\hat{S}}(h) \leq \alpha$. Using event E_1 again, this h obeys $\text{error}_S(h) \leq 2\alpha$. Therefore, event E_2 ensures that h satisfies $\text{error}_{\mathcal{D}}(h, c_t) \leq 2\alpha + \gamma$.

We will now show that these 2 events happen with high probability. By the definition of C^{label}, for every $c^{\text{label}} \in C^{\text{label}}$ we have that

$$Q_{c^{\text{label}}}(S) = \frac{1}{|S|}\left|\left\{i \,:\, c^{\text{label}}(x_i \circ y_i) = 1\right\}\right| = \frac{1}{|S|}\left|\{i \,:\, c(x_i) \neq y_i\}\right| = \text{error}_S(c)\,.$$

Therefore, as A is an $(\alpha,\beta,\varepsilon,m)$-sanitizer for C^{label}, event E_1 happens with probability at least $(1-\beta)$. As

$$m \geq \frac{50\,\text{VC}(C)}{\gamma^2}\ln\left(\frac{1}{\gamma\beta}\right),$$

Theorem 2.14 ensures that event E_2 happens with probability at least $(1-\beta)$ as well. All in all, Learn is a proper $((2\alpha+\gamma),2\beta,\varepsilon,m)$-PPAC learner for C. □

Lemma 5.3 describes a reduction from the task of privately learning a concept class C to the sanitization task of the slightly different concept class C^{label}. We next show that given a sanitizer for a class C, it is possible to construct a sanitizer for C^{label}. Along the way we will also slightly increase the sample complexity of the starting sanitizer, in order to be able to use Lemma 5.3. This results in a reduction from the task of privately learning a concept class C to the sanitization task of the same concept class C.

Lemma 5.4. *If there exists an $(\alpha,\beta,\varepsilon,m)$-sanitizer for a class C, then there exists a $((5\alpha+4\beta),5\beta,6\varepsilon,t)$-sanitizer for C^{label}, where*

$$\frac{100m}{\alpha^2}\ln\left(\frac{1}{\alpha\beta}\right) \leq t \leq \frac{150}{\alpha^2\beta}\ln\left(\frac{2}{\alpha\beta}\right)\left(m+\frac{1}{\varepsilon}\right).$$

Proof. Let A' be an $(\alpha,\beta,\varepsilon,m)$-sanitizer for a class C. By replacing α with 2α, and β with 2β, we can assume that the sanitized databases returned by A' are always of fixed size

$$n = O\left(\frac{\text{VC}(C)}{\alpha^2}\log\left(\frac{1}{\alpha\beta}\right)\right)$$

(see Theorem 2.21). Moreover, we can assume that A' treats its input database as a multiset (as otherwise we could alter A to first randomly shuffle its input database). Denote

$$M = m\left\lceil\frac{18}{\beta}\ln\left(\frac{2}{\alpha\beta}\right)\cdot\left(1+\frac{1}{m\varepsilon}\right)\right\rceil.$$

By Lemma 5.1 for every qM (where $q \in \mathbb{N}$) there exists a $((4\alpha+4\beta),2\beta,\varepsilon,qM)$ sanitizer A for C (as $qM = q'm$ for an integer q'). Denote $t = \lceil 6/\alpha^2\rceil M$, and consider algorithm B presented in Figure 12

Note that the output on Step 8 is just a post-processing of the 4 outputs on Step 7. We first show that each of those 4 outputs preserves differential privacy, and, hence, B is private (with slightly bigger privacy parameter, see Theorem 2.3).

By the properties of the Laplace Mechanism, $\hat{m}_0$ and $\hat{m}_1$ each preserves ε-differential privacy. The analysis for $\widetilde{S}_0$ and $\widetilde{S}_1$ is symmetric, and we next give the analysis for $\widetilde{S}_0$. Denote by B_0 an algorithm identical to the first 7 steps of B, except that the only output of B_0 on Step 7 is $\widetilde{S}_0$. We now show that B_0 is private.

Input: database $D = (x_i, y_i)_{i=1}^t \in (X_{d+1})^t$.

1. Divide $(x_i)_{i=1}^t$ into $S_0, S_1 \in (X_d)^*$, where S_σ contains all x_i s.t. $y_i = \sigma$.
2. Sample $\ell_0, \ell_1 \leftarrow \lfloor \mathrm{Lap}(1/\varepsilon) \rfloor$; that is, ℓ_0 and ℓ_1 are two independent rounded instantiations of a Laplace random variable.
3. Set $m_0 = \max\{0\ ,\ |S_0| + \ell_0\}$ and $m_1 = \max\{0\ ,\ |S_1| + \ell_1\}$.
4. Set $\hat{m}_0 = \left\lfloor \frac{m_0}{M} + \frac{1}{2} \right\rfloor M$, and $\hat{m}_1 = \left\lfloor \frac{m_1}{M} + \frac{1}{2} \right\rfloor M$.
5. For $\sigma \in \{0,1\}$, either add copies of the entry 0^d to S_σ, or remove the last entries from S_σ until $|S_\sigma| = \hat{m}_\sigma$. Denote the resulting multiset as $\hat{S_\sigma}$.
6. Compute $\widetilde{S_0} \leftarrow A(\hat{S_0})$ and $\widetilde{S_1} \leftarrow A(\hat{S_1})$, where if $\hat{S_\sigma} = \emptyset$, then set $\widetilde{S_\sigma} = \emptyset$.
7. Output $\hat{m}_0, \hat{m}_1, \widetilde{S_0}, \widetilde{S_1}$.
8. Construct and output a database $\widetilde{D} \in (X_{d+1})^*$ containing $\hat{m}_0$ copies of $\widetilde{S_0} \circ 0$, and $\hat{m}_1$ copies of $\widetilde{S_1} \circ 1$.

Figure 12: Algorithm B

Notations. We use $S_0[D]$ and $\hat{S_0}[D]$ to denote the databases $S_0, \hat{S_0}$ defined on Steps 1 and 5 in the execution of B_0 on D. Moreover, we use $m_0[D]$ to denote the value of $|S_0| + \ell_0$ in the execution on D, and for every value $m_0[D] = L$, we use $\hat{S_0}[D, L]$ to denote the database $\hat{S_0}$ defined on Step 5, given that $m_0[D] = L$.

Fix two neighboring databases D, D', and let F be a set of possible outputs. Note that as D, D' are neighboring, we have that $S_0[D]$ and $S_0[D']$ are identical up to an addition or a change of one entry. Therefore, whenever $m_0[D] = m_0[D'] = L$, we have that $S_0[D, L]$ and $S_0[D', L]$ are neighboring databases. Moreover, by the properties of the Laplace Mechanism, for every value L we have that

$$\Pr[m_0[D] = L] \leq e^\varepsilon \Pr[m_0[D'] = L]\,.$$

Hence,

$$\begin{aligned}
\Pr[B_0(D) \in F] &= \sum_{L=-\infty}^{\infty} \Pr[m_0[D] = L] \cdot \Pr[B_0(D) \in F \mid m_0[D] = L] \\
&= \sum_{L=-\infty}^{\infty} \Pr[m_0[D] = L] \cdot \Pr[A(\hat{S}_0[D,L]) \in F] \\
&\leq \sum_{L=-\infty}^{\infty} e^{\varepsilon} \cdot \Pr[m_0[D'] = L] \cdot e^{\varepsilon} \cdot \Pr[A(\hat{S}_0[D',L]) \in F] \\
&= e^{2\varepsilon} \cdot \sum_{L=-\infty}^{\infty} \Pr[m_0[D'] = L] \cdot \Pr[B_0(D') \in F \mid m_0[D'] = L] \\
&= e^{2\varepsilon} \cdot \Pr[B_0(D') \in F] .
\end{aligned}$$

Overall (since we use two ε-private algorithms and two (2ε)-private algorithms), algorithm B is (6ε)-differentially private. As for the utility analysis, fix a database $D = (x_i, y_i)_{i=1}^{t}$ and consider the execution of B on D. We now show that w.h.p. the sanitized database $\widetilde{D}$ is $(5\alpha + 4\beta)$-close to D.

First note that by the properties of the Laplace Mechanism, for

$$M \geq \frac{2}{\varepsilon} \ln\left(\frac{2}{\beta}\right),$$

with probability at least $(1-\beta)$ we have that $|\ell_0|, |\ell_1| \leq M/2$. We proceed with the analysis assuming that this is the case. Moreover, note that after the rounding (on Step 4) we have that $|m_\sigma - \hat{m_\sigma}| \leq M/2$. Therefore, for every $\sigma \in \{0,1\}$

$$|S_\sigma| - M \leq |\hat{S_\sigma}| \leq |S_\sigma| + M .$$

Fix a concept $c^{\text{label}} \in C^{\text{label}}$. We have that

$$\begin{aligned}
Q_{c^{\text{label}}}(D) &= \frac{1}{t} |\{i : c^{\text{label}}(x_i, y_i) = 1\}| \\
&= \frac{1}{t} \left[\left| \left\{ i : \begin{array}{c} y_i = 0 \\ c^{\text{label}}(x_i, y_i) = 1 \end{array} \right\} \right| + \left| \left\{ i : \begin{array}{c} y_i = 1 \\ c^{\text{label}}(x_i, y_i) = 1 \end{array} \right\} \right| \right] \\
&= \frac{1}{t} \left[\left| \left\{ i : \begin{array}{c} y_i = 0 \\ c(x_i) = 1 \end{array} \right\} \right| + \left| \left\{ i : \begin{array}{c} y_i = 1 \\ c(x_i) = 0 \end{array} \right\} \right| \right] \\
&\leq \frac{1}{t} \left[\left| \left\{ i : \begin{array}{c} x_i \in \hat{S}_0 \\ c(x_i) = 1 \end{array} \right\} \right| + M + \left| \left\{ i : \begin{array}{c} x_i \in \hat{S}_1 \\ c(x_i) = 0 \end{array} \right\} \right| + M \right] \\
&= \frac{1}{t} \left[\, \hat{m}_0 \cdot Q_c(\hat{S}_0) + \hat{m}_1 \left(1 - Q_c(\hat{S}_1)\right) \, \right] + \frac{2M}{t}.
\end{aligned}$$

By the properties of algorithm A, with probability at least $(1-4\beta)$ we have that $\widetilde{S_0}$ and $\widetilde{S_1}$ are $(4\alpha + 4\beta)$-close to $\hat{S_0}$ and to $\hat{S_1}$ (respectively). We proceed with the analysis assuming that this is the

case. Hence,

$$\begin{aligned} Q_{c^{\text{label}}}(D) &\leq \frac{1}{t}\left[\, \hat{m}_0 \cdot Q_c(\widetilde{S}_0) + (4\alpha+4\beta)\hat{m}_0 + \hat{m}_1\left(1 - Q_c(\widetilde{S}_1)\right) + (4\alpha+4\beta)\hat{m}_1 \,\right] + \frac{2M}{t} \\ &= \frac{1}{t}\left[\, \hat{m}_0 \cdot Q_c(\widetilde{S}_0) + \hat{m}_1\left(1 - Q_c(\widetilde{S}_1)\right) \,\right] + (4\alpha+4\beta)\frac{\hat{m}_0+\hat{m}_1}{t} + \frac{2M}{t} \\ &\leq \frac{1}{t}\left[\, \hat{m}_0 \cdot Q_c(\widetilde{S}_0) + \hat{m}_1\left(1 - Q_c(\widetilde{S}_1)\right) \,\right] + (4\alpha+4\beta)\frac{t+2M}{t} + \frac{2M}{t} \\ &\leq \frac{1}{t}\left[\, \hat{m}_0 \cdot Q_c(\widetilde{S}_0) + \hat{m}_1\left(1 - Q_c(\widetilde{S}_1)\right) \,\right] + (4\alpha+4\beta) + \frac{4M}{t}\,. \end{aligned}$$

Note that as $c^{\text{label}}(x_i \circ 0) = c(x_i)$ and as $c^{\text{label}}(x_i \circ 1) = 1 - c(x_i)$, we have that

$$\begin{aligned} Q_{c^{\text{label}}}(\widetilde{S}_0 \circ 0) &= Q_c(\widetilde{S}_0)\,; \\ Q_{c^{\text{label}}}(\widetilde{S}_1 \circ 1) &= 1 - Q_c(\widetilde{S}_1)\,. \end{aligned}$$

Hence,

$$Q_{c^{\text{label}}}(D) \leq \frac{1}{t}\left[\, \hat{m}_0 \cdot Q_{c^{\text{label}}}(\widetilde{S}_0 \circ 0) + \hat{m}_1 \cdot Q_{c^{\text{label}}}(\widetilde{S}_1 \circ 1) \,\right] + (4\alpha+4\beta) + \frac{4M}{t}\,.$$

Denoting $\widetilde{D} = (z_i)_{i=1}^r \in (X_{d+1})^r$ (where $r = n(\hat{m}_0+\hat{m}_1)$), we get

$$\begin{aligned} &Q_{c^{\text{label}}}(D) \\ &\leq \frac{1}{nt}\left[\, \hat{m}_0 \cdot \left|\left\{ i \,:\, \begin{array}{c} z_i \in \widetilde{S}_0 \circ 0 \\ c^{\text{label}}(z_i) = 1 \end{array} \right\}\right| + \hat{m}_1 \cdot \left|\left\{ i \,:\, \begin{array}{c} z_i \in \widetilde{S}_1 \circ 1 \\ c^{\text{label}}(z_i) = 1 \end{array} \right\}\right| \,\right] + (4\alpha+4\beta) + \frac{4M}{t} \\ &= \frac{1}{nt}\left|\left\{ i \,:\, c^{\text{label}}(z_i) = 1 \right\}\right| + (4\alpha+4\beta) + \frac{4M}{t} \\ &= \frac{\hat{m}_0+\hat{m}_1}{t} \cdot Q_{c^{\text{label}}}(\widetilde{D}) + (4\alpha+4\beta) + \frac{4M}{t} \\ &\leq \frac{t+2M}{t} \cdot Q_{c^{\text{label}}}(\widetilde{D}) + (4\alpha+4\beta) + \frac{4M}{t} \\ &\leq Q_{c^{\text{label}}}(\widetilde{D}) + (4\alpha+4\beta) + \frac{6M}{t} \\ &\leq Q_{c^{\text{label}}}(\widetilde{D}) + (5\alpha+4\beta)\,. \end{aligned}$$

Similar arguments show that $Q_{c^{\text{label}}}(D) \geq Q_{c^{\text{label}}}(\widetilde{D}) - (5\alpha+4\beta)$. Algorithm B is, therefore, a $(5\alpha+4\beta), 5\beta, 6\varepsilon, t)$-sanitizer for C^{label}, where

$$t = \left\lceil \frac{6}{\alpha^2} \right\rceil M = \left\lceil \frac{6}{\alpha^2} \right\rceil \cdot \left\lceil \frac{18}{\beta} \ln\!\left(\frac{2}{\alpha\beta}\right)\left(1 + \frac{1}{m\varepsilon}\right) \right\rceil \cdot m = O_{\alpha,\beta,\varepsilon}(m)\,. \qquad \square$$

Theorem 5.5. *Let $\alpha, \varepsilon \leq 1/8$, and let C be a class of predicates. If there exists an $(\alpha, \beta, \varepsilon, m)$-sanitizer A for C, then there exists a proper $((15\alpha+12\beta), 10\beta, 6\varepsilon, t)$-PPAC learner for C, where $t = O_{\alpha,\beta,\varepsilon}(m)$.*

Proof. Let A be an $(\alpha,\beta,\varepsilon,m)$-sanitizer for C. Note that by Theorem 2.20, it must be that $m \geq \mathrm{VC}(C)/2$. By Lemma 5.4, there exists a $((5\alpha+4\beta),5\beta,6\varepsilon,t)$-sanitizer for C^{label}, where $t = O_{\alpha,\beta,\varepsilon}(m)$ and

$$t \geq \frac{100m}{\alpha^2}\ln\left(\frac{1}{\alpha\beta}\right) \geq \frac{50\,\mathrm{VC}(C)}{\alpha^2}\ln\left(\frac{1}{\alpha\beta}\right).$$

By Lemma 5.3, there exists a proper $((15\alpha+12\beta),10\beta,6\varepsilon,t)$-PPAC learner for C. □

Remark 5.6. Given an efficient proper-sanitizer for C and assuming the existence of an efficient *non-private* learner for C, this reduction results in an efficient *private* learner for C.

5.2 A lower bound for k-POINT$_d$

Next we prove a lower bound on the database size of every sanitizer for k-POINT$_d$ that preserves pure differential privacy.

Consider the following concept class over X_d. For every $A \subseteq X_d$ s.t. $|A| = k$, the concept class k-POINT$_d$ contains the concept $c_A : X_d \to \{0,1\}$, defined as $c_A(x) = 1$ if $x \in A$ and $c_A(x) = 0$ otherwise. The VC dimension of k-POINT$_d$ is k (assuming $2^d \geq 2k$).

To prove a lower bound on the sample complexity of sanitization, we first prove a lower bound on the sample complexity of the related learning problem and then use the reduction (Theorem 5.5). Thus, we start by showing that every private proper learner for k-POINT$_d$ requires $\Omega(kd/(\alpha\varepsilon))$ labeled examples. A similar version of this lemma appeared in Beimel et al. [3], where it is shown that every private proper learner for POINT$_d$ requires $\Omega(d/(\alpha\varepsilon))$ labeled examples.

Lemma 5.7. *Let $\alpha < 1/5$, and let k,d be s.t. $2^d \geq k^{1.1}$. If L is a proper $(\alpha,1/2,\varepsilon,m)$-PPAC learner for* k-POINT$_d$*, then $m = \Omega(kd/(\alpha\varepsilon))$.*

Proof. Let L be a proper $(\alpha,1/2,\varepsilon,m)$-PPAC learner for k-POINT$_d$. Without loss of generality, we can assume that $m \geq 5\ln(4)/(3\alpha)$ (since L can ignore part of the sample).

Consider a maximal cardinality subset $B \subseteq$ k-POINT$_d$ s.t. for every $c_A \in B$ we have $0^d \notin A$, and moreover, for every $c_{A_1} \neq c_{A_2} \in B$ we have $|A_1 \cap A_2| \leq k/2$. Such a set B satisfies

$$|B| \geq \left(\frac{2^d-1}{4e^2k}\right)^{k/2}.$$

To see this, we could construct such a set using the following greedy algorithm. Initiate $\hat{B} = \emptyset$, and $C =$ k-POINT$_d \setminus \{c_I \in$ k-POINT$_d : 0^d \in I\}$. While $C \neq \emptyset$, arbitrarily choose a concept $c_A \in C$, add c_A to $\hat{B}$, and remove from C every concept c_I s.t. $|A \cap I| \leq k/2$.

Clearly, for every two $c_{A_1} \neq c_{A_2} \in \hat{B}$ we have $|A_1 \cap A_2| \leq k/2$. Moreover, at every step, the number of concepts that are removed from C is at most

$$\sum_{j=k/2}^{k}\binom{k}{j}\cdot\binom{2^d-1-k}{k-j} \leq \binom{k}{k/2}\cdot\binom{2^d-1}{k/2},$$

and, therefore,

$$\hat{B} \geq \frac{\binom{2^d-1}{k}}{\binom{k}{k/2}\cdot\binom{2^d-1}{k/2}} \geq \left(\frac{2^d-1}{4e^2k}\right)^{k/2}.$$

For every $c_A \in B$ we will now define a distribution $\mathcal{D}_A$, a set of hypotheses $G(A)$, and a database S_A. The distribution $\mathcal{D}_A$ is defined as

$$\mathcal{D}_A(x) = \begin{cases} 1-5\alpha, & x = 0^d, \\ 5\alpha/k, & x \in A, \\ 0, & \text{else.} \end{cases}$$

Define the set $G(A) \subseteq \texttt{k-POINT}_d$ as all α-good hypothesis for $(c_A, \mathcal{D}_A)$ in $\texttt{k-POINT}_d$. Note that for every $h_I \in \texttt{k-POINT}_d$ s.t. $\text{error}_{\mathcal{D}_A}(h_I, c_A) \leq \alpha$ we have $|I \cap A| \geq 4k/5$. Therefore, for every $c_{A_1} \neq c_{A_2} \in B$ we have $G(A_1) \cap G(A_2) = \emptyset$ (as $|A_1 \cap A_2| \leq k/2$, and as $|A_1| = |A_2| = |I| = k$).

By the utility properties of L, we have that

$$\Pr_{L,\mathcal{D}_A}[L(S) \in G(A)] \geq \frac{1}{2}.$$

We say that a database S of m labeled examples is *good* if the unlabeled example 0^d appears in S at least $(1-8\alpha)m$ times. Let S be a database constructed by taking m i.i.d. samples from $\mathcal{D}_A$, labeled by c_A. By the Chernoff bound, S is good with probability at least $1-\exp(-3\alpha m/5)$. Hence,

$$\Pr_{\mathcal{D}_A,L}[(L(S) \in G(A)) \wedge (S \text{ is good})] \geq \frac{1}{2} - \exp(-3\alpha m/5) \geq \frac{1}{4}.$$

Note that, as $0^d \notin A$, every appearance of the example 0^d in S is labeled by 0. Therefore, there exists a good database S of m samples that contains the entry $0^d \circ 0$ at least $(1-8\alpha)m$ times, and

$$\Pr_L[L(S) \in G(A)] \geq \frac{1}{4},$$

where the probability is only over the randomness of L. We define S_A as such a database.

Note that all of the databases S_{A_i} defined here are of distance at most $8\alpha m$ from one another. The privacy of L ensures, therefore, that for any two such S_{A_i}, S_{A_j} we have

$$\Pr_L[L(S_{A_i}) \in G(A_j)] \geq \frac{1}{4}\exp(-8\alpha\varepsilon m).$$

Now,

$$\begin{aligned}
1-\frac{1}{4} &\geq \Pr_L[L(S_{A_i}) \notin G(A_i)] \\
&\geq \Pr_L\left[L(S_{A_i}) \in \bigcup_{A_j \neq A_i} G(A_j)\right] \\
&\geq \sum_{A_j \neq A_i} \Pr_L[L(S_{A_i}) \in G(A_j)] \\
&\geq (|B|-1)\frac{1}{4}\exp(-8\alpha\varepsilon m) \\
&\geq \left(\left(\frac{2^d-1}{4e^2k}\right)^{k/2}-1\right)\frac{1}{4}\exp(-8\alpha\varepsilon m)\,. \qquad (5.1)
\end{aligned}$$

Solving for m yields

$$m = \Omega\Big(\frac{k}{\alpha\varepsilon}(d-\ln(k))\Big)\,.$$

Recall that $2^d \geq k^{1.1}$, and, hence, $m = \Omega(kd/(\alpha\varepsilon))$. □

Remark 5.8. The constant 1.1 in Lemma 5.7 could be replaced with any constant strictly bigger than 1. Moreover, whenever $2^d = O(k)$ we have that $|\,\texttt{k-POINT}_d\,| = \binom{2^d}{k} = 2^{O(2^d)}$ and, hence, the generic construction of Kasiviswanathan et al. [22] yields a proper ε-private learner for this class with sample complexity $O_{\alpha,\beta,\varepsilon}(2^d) = O_{\alpha,\beta,\varepsilon}(k)$.

In the next lemma we will use the last lower bound on the sample complexity of private learners, together with the reduction of Theorem 5.5, and derive a lower bound on the database size necessary for pure private sanitizers for $\texttt{k-POINT}_d$.

Theorem 5.9. *Let $\varepsilon \leq 1/8$, and let k and d be s.t. $2^d \geq k^{1.1}$. Every $(1/150, 1/150, \varepsilon, m)$-sanitizer for $\texttt{k-POINT}_d$ requires databases of size*

$$m = \Omega\left(\frac{1}{\varepsilon}\,\mathrm{VC}(\texttt{k-POINT}_d)\cdot\log|X_d|\right)\,.$$

Proof. Let A be a $(1/150, 1/150, \varepsilon, m)$-sanitizer for $\texttt{k-POINT}_d$. By Theorem 5.5, there exists a proper $(9/50, 1/15, 6\varepsilon, t)$-PPAC learner for $\texttt{k-POINT}_d$, where $t = O(m)$. By Lemma 5.7, $t = \Omega(kd/\varepsilon)$, and hence $m = \Omega(kd/\varepsilon)$. □

Recall that in the proof of Theorem 5.5, we increased the sample complexity in order to use Lemma 5.3. This causes a slackness of α^2 in the database size of the resulting learner, which, in turn, eliminates the dependency in α in Theorem 5.9. For the class $\texttt{k-POINT}_d^{\text{label}}$ it is possible to obtain a better lower bound, by using the reduction of Lemma 5.3 twice.

Theorem 5.10. *Let $\alpha \leq 1/50$ and $\varepsilon \leq 1/8$. There exist a $d_0 = d_0(\alpha,\varepsilon)$ s.t. for every k and d s.t. $2^d \geq \max\{k^{1.1}, 2^{d_0}\}$, every $(\alpha, 1/50, \varepsilon, m)$-sanitizer for $\texttt{k-POINT}_d^{\text{label}}$ must operate on databases of size*

$$m = \Omega\left(\frac{1}{\alpha\varepsilon}\,\mathrm{VC}(\texttt{k-POINT}_d^{\text{label}}) \cdot \log|X_d|\right).$$

Proof. Let A be a $(1/50, 1/50, \varepsilon, m)$-sanitizer for a class $\texttt{k-POINT}_d^{\text{label}}$, where $\varepsilon \leq 1/8$. Note that by Theorem 2.20, it must be that

$$m \geq \frac{\mathrm{VC}(\texttt{k-POINT}_d^{\text{label}})}{2} \geq \frac{\mathrm{VC}(\texttt{k-POINT}_d)}{2}.$$

In order to use Lemma 5.3, we need a slightly stronger guarantee, and therefore use Lemma 5.1 to increase the input database size as follows.

Denote $q = \left\lceil 100 \cdot 50^3 \ln(50^2) \right\rceil$. By Lemma 5.1, there exists a $(2/25, 1/50, \varepsilon, t)$-sanitizer B for $\texttt{k-POINT}_d^{\text{label}}$, where

$$t = qm = m\left\lceil 100 \cdot 50^3 \ln(50^2) \right\rceil \geq 50 \cdot 50^3\,\mathrm{VC}(\texttt{k-POINT}_d)\ln(50^2).$$

By Lemma 5.3, there exists a proper $(9/50, 1/25, \varepsilon, t)$-PPAC learner for $\texttt{k-POINT}_d$. By Lemma 5.7, $t = \Omega(kd/\varepsilon)$, and hence

$$m = \Omega\left(\frac{kd}{\varepsilon}\right). \tag{5.2}$$

Let $\alpha \leq 1/50$ and $\varepsilon \leq 1/8$, and let B be an $(\alpha, 1/50, \varepsilon, m)$-sanitizer for $\texttt{k-POINT}_d^{\text{label}}$. As B is, in particular, a $(1/50, 1/50, \varepsilon, m)$-sanitizer for $\texttt{k-POINT}_d^{\text{label}}$, where $\varepsilon \leq 1/8$, Equation 5.2 states that there exists a constant λ s.t. $m \geq \lambda(kd/\varepsilon)$. Asserting that

$$d \geq d_0 \triangleq \frac{50\varepsilon}{\lambda\alpha^2}\ln\left(\frac{50}{\alpha}\right),$$

we ensure that

$$m \geq \frac{50k}{\alpha^2}\ln\left(\frac{50}{\alpha}\right).$$

By reusing Lemme 5.3, we now get that there exists a proper $(3\alpha, 1/25, \varepsilon, m)$-PPAC learner for $\texttt{k-POINT}_d$. Lemma 5.7 now states that

$$m = \Omega\left(\frac{kd}{\alpha\varepsilon}\right) = \Omega\left(\frac{1}{\alpha\varepsilon}\,\mathrm{VC}(\texttt{k-POINT}_d^{\text{label}}) \cdot \log|X_d|\right). \qquad \square$$

6 Label-private learners

6.1 Generic label-private learner

In this section we consider relaxed definitions of private learners preserving pure privacy (i. e., $\delta = 0$). We start with the model of label privacy (see [9] and references therein). In this model, privacy must only

be preserved for the *labels* of the elements in the database, and not necessarily for their identity. This is a reasonable privacy requirement when the identity of individuals in a population are known publicly but not their labels. In general, this is not a reasonable assumption.

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

Definition 6.1 (Label-Private Learner)**.** Let A be an algorithm that gets as input a database $S_x \in X^m$ and its labels $S_y \in \{0,1\}^m$. Algorithm A is an $(\alpha, \beta, \varepsilon, m)$-*Label Private PAC Learner* for a concept class C over X if

PRIVACY. $\forall S_x \in X^m$, algorithm $\mathsf{A}(S_x, \cdot) = \mathsf{A}_{S_x}(\cdot)$ is ε-differentially private (as in Definition 2.2);

UTILITY. Algorithm A is an (α, β, m)-*PAC learner* for C (as in Definition 2.7).

Chaudhuri et al. [9] proved lower bounds on the sample complexity of label-private learners for a class C in terms of its doubling dimension. As we will now see, 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 6.2. *Let C be a concept class over a domain X. For every $\alpha, \beta, \varepsilon$, there exists an $(\alpha, \beta, \varepsilon, m)$-Label Private PAC learner for C, where $m = O_{\alpha,\beta,\varepsilon}(\mathrm{VC}(C))$. The learner might not be efficient.*

Proof. For a concept class C over a domain X, and for a subset $B = \{b_1, \ldots, b_\ell\} \subseteq X$, the projection of C on B is denoted as

$$\Pi_C(B) = \{(c(b_1), \ldots, c(b_\ell)) : c \in C\}.$$

In Figure 13 we describe a label-private algorithm A. Algorithm A constructs a set of hypotheses H as follows: It samples an unlabeled sample S_1, and defines B as the set of points in S_1. For every labeling of the points in B realized by C, add to H an arbitrary concept consistent with this labeling. Afterwards, algorithm A uses the Exponential Mechanism to choose a hypothesis out of H.

Note that steps 1-4 of algorithm A are independent of the labeling vector S_y. By the properties of the Exponential Mechanism (which is used to access S_y on Step 5), for every set of elements S_x, algorithm $\mathsf{A}(S_x, \cdot)$ is ε-differentially private.

For the utility analysis, fix a target concept $c \in C$ and a distribution $\mathcal{D}$ over X, and define the following 3 good events:

E_1 The constructed set H contains at least one hypothesis f s.t. $\mathrm{error}_{S^2}(f) \leq \alpha/4$.

E_2 For every $h \in H$ s.t. $\mathrm{error}_{S^2}(h) \leq \alpha/2$, we have $\mathrm{error}_{\mathcal{D}}(c,h) \leq \alpha$.

E_3 The Exponential Mechanism chooses an h such that $\mathrm{error}_{S^2}(h) \leq \alpha/4 + \min_{f \in H}\{\mathrm{error}_{S^2}(f)\}$.

We first show that if these 3 good events happen, then algorithm A returns an α-good hypothesis. Event E_1 ensures the existence of a hypothesis $f \in H$ s.t. $\mathrm{error}_{S^2}(f) \leq \alpha/4$. Thus, event $E_1 \cap E_3$ ensures algorithm A chooses (using the Exponential Mechanism) a hypothesis $h \in H$ s.t. $\mathrm{error}_{S^2}(h) \leq \alpha/2$. Event E_2 ensures, therefore, that this h obeys $\mathrm{error}_{\mathcal{D}}(c,h) \leq \alpha$.

Generic Label-Private Learner
Input: parameter ε, and a labeled database $S = (x_i, y_i)_{i=1}^m$, where

$$m \geq \frac{768}{\alpha^2 \varepsilon}\left(\mathrm{VC}(C)\ln\left(\frac{64}{\alpha}\right) + 2\ln\left(\frac{8}{\beta}\right)\right).$$

1. Initialize $H = \emptyset$.
2. Set $n = \frac{32}{\alpha}\left(\mathrm{VC}(C)\ln\left(\frac{64}{\alpha}\right) + \ln\left(\frac{8}{\beta}\right)\right)$, and denote $S^1 = (x_i, y_i)_{i=1}^n$, and $S^2 = (x_i, y_i)_{i=n+1}^m$.
3. Let $B = \{b_1, \ldots, b_\ell\}$ be the set of all the (unlabeled) points appearing at least once in S^1.
4. For every $(z_1, \ldots, z_\ell) \in \Pi_C(B)$,
   (a) Let $c \in C$ be an arbitrary concept in C s.t. $c(b_i) = z_i$ for every $1 \leq i \leq \ell$.
   (b) Add c to H.
5. Choose and return $h \in H$ using the Exponential Mechanism with inputs ε, H, S^2 and the quality function $q(S^2, h) = |\{i : h(x_i) = y_i\}|$.

Figure 13: A generic label-private learner.

We will now show that those 3 events happen with high probability. For every $(y_1, \ldots, y_\ell) \in \Pi_C(B)$, algorithm A adds to H a hypothesis f s.t. $\forall 1 \leq i \leq \ell,\ f(b_i) = y_i$. In particular, H contains a hypothesis h^* s.t. $h^*(x) = c(x)$ for every $x \in B$. That is, a hypothesis h^* s.t. $\mathrm{error}_{S^1}(h^*) = 0$. Therefore, by setting

$$n \geq \frac{32}{\alpha}\left(\mathrm{VC}(C)\ln\left(\frac{64}{\alpha}\right) + \ln\left(\frac{8}{\beta}\right)\right),$$

Theorem 2.13 ensures that $\mathrm{error}_{\mathcal{D}}(c, h^*) \leq \alpha/8$ with probability at least $(1 - \beta/4)$. In such a case, using the Chernoff bound, we get that with probability at least $1 - \exp(-(m-n)\alpha/24)$ this hypothesis h^* satisfies $\mathrm{error}_{S^2}(h^*) \leq \alpha/4$. Event E_1 happens, therefore, with probability at least

$$\left(1 - \frac{\beta}{4}\right)\left(1 - \exp\left(-\frac{(m-n)\alpha}{24}\right)\right),$$

which is at least $(1 - \beta/2)$ for

$$m \geq n + \frac{24}{\alpha}\ln\left(\frac{4}{\beta}\right).$$

Fix a hypothesis h s.t. $\mathrm{error}_{\mathcal{D}}(c, h) > \alpha$. Using the Chernoff bound, the probability that $\mathrm{error}_{S^2}(h) \leq \alpha/2$ is less than $\exp(-(m-n)\alpha/8)$. As $|H| = 2^{|B|} \leq 2^n$, the probability that there is such a hypothesis in H is at most $2^n \cdot \exp(-(m-n)\alpha/8)$. For

$$m \geq \frac{8}{\alpha}\left(n + \ln\left(\frac{4}{\beta}\right)\right),$$

this probability is at most $\beta/4$, and event E_2 happens with probability at least $(1-\beta/4)$.

The Exponential Mechanism ensures that the probability of event E_3 is at least $1-|H|\cdot\exp(-\varepsilon\alpha m/8)$ (see Proposition 2.26), which is at least $(1-\beta/4)$ for

$$m \geq \frac{8}{\alpha\varepsilon}\left(n+\ln\left(\frac{4}{\beta}\right)\right).$$

All in all, by setting

$$n = \frac{32}{\alpha}\left(\mathrm{VC}(C)\ln\left(\frac{64}{\alpha}\right)+\ln\left(\frac{8}{\beta}\right)\right) \qquad \text{and} \qquad m \geq \frac{768}{\alpha^2\varepsilon}\left(\mathrm{VC}(C)\ln\left(\frac{64}{\alpha}\right)+2\ln\left(\frac{8}{\beta}\right)\right),$$

we ensure that the probability of A failing to output an α-good hypothesis is at most β. □

6.2 Label privacy extension

We consider a slight generalization of the label privacy model. Recall that given a labeled sample, a private learner is required to preserve the privacy of the entire sample, while a label-private learner is only required to preserve privacy for the labels of each entry.

Consider a scenario where there is no need in preserving the privacy of the distribution $\mathcal{D}$ (for example, $\mathcal{D}$ might be publicly known), but we still want to preserve the privacy of the entire sample S. We can model this scenario as a learning algorithm A which is given as input 2 databases—a labeled database S, and an unlabeled database D. For every database D, algorithm $\mathsf{A}(D,\cdot)=\mathsf{A}_D(\cdot)$ must preserve differential privacy. We will refer to such a learner as a *Semi-Private* learner.

Clearly, $\Omega(\mathrm{VC}(C))$ samples are necessary in order to semi-privately learn a concept class C, as this is the case for non-private learners.[5] This lower bound is tight, as the generic learner in Figure 13 could easily be adjusted for the semi-privacy model, and result in a generic semi-private learner with sample complexity $O_{\alpha,\beta,\varepsilon}(\mathrm{VC}(C))$. To see this, recall that in the generic learner, the input sample S is divided into S_1 and S_2. Note that the labels in S_1 are ignored, and, hence, S_1 could be replaced with an unlabeled database. Moreover, note that S_2 is only accessed using the Exponential Mechanism (on Step 5), which preserves the privacy both for the labels and for the examples in S_2.

Example 6.3. Consider the task of learning a concept class C, and suppose that the relevant distribution over the population is publicly known. Now, given a labeled database S, we can use a semi-private learner and guarantee privacy both for the labellings and for the mere existence of an individual in the database. That is, in such a case, the privacy guarantee of a semi-private learner is the same as that of a private learner. Moreover, the necessary sample complexity is $O_{\alpha,\beta,\varepsilon}(\mathrm{VC}(C))$, which should be contrasted with $O_{\alpha,\beta,\varepsilon}(\log|C|)$ which is the sample complexity that would result from the general construction of Kasiviswanathan et al. [22].

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

[5] The lower bound of $\Omega(\mathrm{VC}(C))$ is worst case over choices of distributions $\mathcal{D}$. For a specific distribution, less samples may suffice.

References

[1] MARTIN ANTHONY AND JOHN SHAWE-TAYLOR: A result of Vapnik with applications. *Discrete Applied Mathematics*, 47(3):207–217, 1993. Erratum in Discrete Applied Mathematics. [doi:10.1016/0166-218X(93)90126-9] 8

[2] MATIN ANTHONY AND PETER L. BARTLETT: *Neural Network Learning: Theoretical Foundations*. Cambridge Univ. Press, 2009. 8

[3] AMOS BEIMEL, HAI BRENNER, SHIVA PRASAD KASIVISWANATHAN, AND KOBBI NISSIM: Bounds on the sample complexity for private learning and private data release. *Machine Learning*, 94(3):401–437, 2014. Preliminary version in TCC'10. [doi:10.1007/s10994-013-5404-1] 2, 3, 5, 13, 32, 35, 42, 51

[4] AMOS BEIMEL, KOBBI NISSIM, AND URI STEMMER: Characterizing the sample complexity of private learners. In *Proc. 4th Innovations in Theoretical Computer Science Conf. (ITCS'13)*, pp. 97–110. ACM Press, 2013. [doi:10.1145/2422436.2422450, arXiv:1402.2224] 3, 13

[5] AMOS BEIMEL, KOBBI NISSIM, AND URI STEMMER: Private learning and sanitization: Pure vs. approximate differential privacy. In *Proc. 17th Internat. Workshop on Randomization and Computation (RANDOM'13)*, volume 8096 of *Lecture Notes in Computer Science*, pp. 363–378. Springer, 2013. [doi:10.1007/978-3-642-40328-6_26, arXiv:1407.2674] 1

[6] AVRIM BLUM, CYNTHIA DWORK, FRANK MCSHERRY, AND KOBBI NISSIM: Practical privacy: The SuLQ framework. In *Proc. 24nd Symp. on Principles of Database Systems (PODS'05)*, pp. 128–138. ACM Press, 2005. [doi:10.1145/1065167.1065184] 2, 4, 5, 26, 27

[7] AVRIM BLUM, KATRINA LIGETT, AND AARON ROTH: A learning theory approach to non-interactive database privacy. *J. ACM*, 60(2):12, 2013. Preliminary version in STOC'08. [doi:10.1145/2450142.2450148] 2, 3, 5, 9, 10, 35, 43, 44

[8] ANSELM BLUMER, ANDRZEJ EHRENFEUCHT, DAVID HAUSSLER, AND MANFRED K. WARMUTH: Learnability and the Vapnik-Chervonenkis dimension. *J. ACM*, 36(4):929–965, 1989. [doi:10.1145/76359.76371] 8

[9] KAMALIKA CHAUDHURI AND DANIEL HSU: Sample complexity bounds for differentially private learning. In *Proc. 24th Ann. Conf. on Learning Theory (COLT'11)*, volume 19, pp. 155–186. JMLR, 2011. Available at NCBL. 4, 5, 54, 55

[10] ANINDYA DE: Lower bounds in differential privacy. In *9th Theory of Cryptography Conf. (TCC'12)*, volume 7194 of *Lecture Notes in Computer Science*, pp. 321–338. Springer, 2012. [doi:10.1007/978-3-642-28914-9_18, arXiv:1107.2183] 5

[11] CYNTHIA DWORK, KRISHNARAM KENTHAPADI, FRANK MCSHERRY, ILYA MIRONOV, AND MONI NAOR: Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pp. 486–503. Springer, 2006. [doi:10.1007/11761679_29] 2, 6

[12] CYNTHIA DWORK AND JING LEI: Differential privacy and robust statistics. In *Proc. 41st STOC*, pp. 371–380. ACM Press, 2009. [doi:10.1145/1536414.1536466] 3, 6, 12

[13] CYNTHIA DWORK, FRANK MCSHERRY, KOBBI NISSIM, AND ADAM SMITH: Calibrating noise to sensitivity in private data analysis. In *3rd Theory of Cryptography Conf. (TCC'06)*, volume 3876 of *Lecture Notes in Computer Science*, pp. 265–284. Springer, 2006. [doi:10.1007/11681878_14] 2, 6, 11

[14] CYNTHIA DWORK, MONI NAOR, OMER REINGOLD, GUY N. ROTHBLUM, AND SALIL P. VADHAN: On the complexity of differentially private data release: efficient algorithms and hardness results. In *Proc. 41st STOC*, pp. 381–390. ACM Press, 2009. [doi:10.1145/1536414.1536467] 3

[15] CYNTHIA DWORK, GUY N. ROTHBLUM, AND SALIL P. VADHAN: Boosting and differential privacy. In *Proc. 51st FOCS*, pp. 51–60. IEEE Comp. Soc. Press, 2010. [doi:10.1109/FOCS.2010.12] 6

[16] ANDRZEJ EHRENFEUCHT, DAVID HAUSSLER, MICHAEL J. KEARNS, AND LESLIE G. VALIANT: A general lower bound on the number of examples needed for learning. *Inf. Comput.*, 82(3):247–261, 1989. Preliminary version in COLT'88. [doi:10.1016/0890-5401(89)90002-3] 8

[17] VITALY FELDMAN AND DAVID XIAO: Sample complexity bounds on differentially private learning via communication complexity. In *Proc. 27th Ann. Conf. on Learning Theory (COLT'14)*, pp. 1000–1019, 2014. Available at JMLR. [arXiv:1402.6278] 3, 14, 28

[18] ANUPAM GUPTA, MORITZ A. W. HARDT, AARON ROTH, AND JONATHAN ULLMAN: Privately releasing conjunctions and the statistical query barrier. *SIAM J. Comput.*, 42(4):1494–1520, 2013. Preliminary version in STOC'11. [doi:10.1137/110857714] 4, 44

[19] MORITZ A. W. HARDT: *A Study of Privacy and Fairness in Sensitive Data Analysis*. Ph. D. thesis, Princeton University, 2011. Available at DataSpace. 3, 4, 10

[20] MORITZ A. W. HARDT AND GUY N. ROTHBLUM: A multiplicative weights mechanism for privacy-preserving data analysis. In *Proc. 51st FOCS*, pp. 61–70. IEEE Comp. Soc. Press, 2010. [doi:10.1109/FOCS.2010.85] 5

[21] MORITZ A. W. HARDT AND KUNAL TALWAR: On the geometry of differential privacy. In *Proc. 42nd STOC*, pp. 705–714. ACM Press, 2010. [doi:10.1145/1806689.1806786] 5

[22] SHIVA PRASAD KASIVISWANATHAN, HOMIN K. LEE, KOBBI NISSIM, SOFYA RASKHODNIKOVA, AND ADAM SMITH: What can we learn privately? *SIAM J. Comput.*, 40(3):793–826, 2011. Preliminary version in FOCS'08. [doi:10.1137/090756090] 2, 5, 9, 12, 14, 26, 27, 28, 53, 57

[23] MICHAEL J. KEARNS: Efficient noise-tolerant learning from statistical queries. *J. ACM*, 45(6):983–1006, 1998. Preliminary version in STOC'93. [doi:10.1145/293347.293351] 4, 26

[24] FRANK MCSHERRY AND KUNAL TALWAR: Mechanism design via differential privacy. In *Proc. 48th FOCS*, pp. 94–103. IEEE Comp. Soc. Press, 2007. [doi:10.1109/FOCS.2007.66] 3, 11

[25] AARON ROTH: Differential privacy and the fat-shattering dimension of linear queries. In *Proc. 14th Internat. Workshop on Randomization and Computation (RANDOM'10)*, volume 6302 of *Lecture Notes in Computer Science*, pp. 683–695. Springer, 2010. [doi:10.1007/978-3-642-15369-3_51] 3

[26] ABHRADEEP THAKURTA AND ADAM SMITH: Differentially private feature selection via stability arguments, and the robustness of the lasso. In *Proc. 26th Ann. Conf. on Learning Theory (COLT'13)*, volume 30, pp. 819–850. JMLR, 2013. Available at JMLR. 3, 12

[27] JONATHAN ULLMAN: Answering $n^{2+o(1)}$ counting queries with differential privacy is hard. In *Proc. 45th STOC*, pp. 361–370. ACM Press, 2013. [doi:10.1145/2488608.2488653] 3

[28] JONATHAN ULLMAN AND SALIL P. VADHAN: PCPs and the hardness of generating private synthetic data. In *8th Theory of Cryptography Conf. (TCC'11)*, volume 6597 of *Lecture Notes in Computer Science*, pp. 400–416. Springer, 2011. [doi:10.1007/978-3-642-19571-6_24] 3

[29] LESLIE G. VALIANT: A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984. Preliminary version in STOC'84. [doi:10.1145/1968.1972] 2, 7

[30] VLADIMIR N. VAPNIK AND ALEXEY Y. CHERVONENKIS: On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971. [doi:10.1137/1116025] 7, 8

AUTHORS

Amos Beimel
Ben-Gurion University
beimel@cs.bgu.ac.il
http://www.cs.bgu.ac.il/~beimel

Kobbi Nissim
Ben-Gurion University
kobbi@cs.bgu.ac.il
http://www.cs.bgu.ac.il/~kobbi

Uri Stemmer
Ben-Gurion University
stemmer@cs.bgu.ac.il
http://www.cs.bgu.ac.il/~stemmer

ABOUT THE AUTHORS

AMOS BEIMEL received the B. A., M. Sc., and D. Sc. degrees in Computer Science from the Technion – Israel Institute of Technology, Haifa, in 1989, 1992, and 1996, respectively. His doctoral thesis was titled "Secure schemes for secret sharing and key distribution." After graduating from the Technion, he spent one year as a Postdoctoral Fellow at the Center for Discrete Mathematics and Computer Science (DIMACS) at Rutgers University, and two years as a Postdoctoral Fellow at the Division of Engineering and Applied Science at Harvard University. In 1999 he joined the Department of Computer Science at Ben-Gurion University, where he is now a professor. In 2005-2006, he spent a year as a visiting assistant professor at the University of California, Davis. In 2012-14, he was the head of the department of computer science at Ben-Gurion university. His research interests include cryptography and complexity theory. He focuses on secret sharing schemes, private information retrieval, secure multiparty computation, and differential privacy. He published more than 50 papers in journals and conferences, received grants from the Israel Science Foundation, the Ministry of Science and Technology, and the Ministry of Economy, and served on various program committees of conferences.

KOBBI NISSIM is a faculty member at the Department of Computer Science, Ben-Gurion University. His research interests are in the foundations of privacy and cryptography, and in particular, formal notions of privacy, differential privacy, privacy-aware mechanism design, private approximations, and secure multiparty computation. He received his Ph. D. from the Weizmann Institute in 2001 where he studied under the direction of Moni Naor. In 2013, Nissim received, with Irit Dinur, the "Alberto O. Mendelzon Test-of-Time" Award for their PODS 2003 paper *Revealing Information while Preserving Privacy*. In 2016, Nissim received, with Cynthia Dwork, Frank McSherry, and Adam Smith the "TCC Test-of-Time" Award for their TCC 2006 paper *Calibrating Noise to Sensitivity in Private Data Analysis* where differential privacy was introduced.

URI STEMMER is a Ph. D. candidate at Ben-Gurion University, advised by Amos Beimel and Kobbi Nissim. His research interests lie in private data analysis and computational learning theory.