Bounds on the Sample Complexity for Private Learning and Private Data Release

Amos Beimel[1,⋆], Shiva Prasad Kasiviswanathan[2], and Kobbi Nissim[1,3,⋆⋆]

[1] Dept. of Computer Science, Ben-Gurion University
[2] CCS-3, Los Alamos National Laboratory
[3] Microsoft Audience Intelligence

Abstract. Learning is a task that generalizes many of the analyses that are applied to collections of data, and in particular, collections of sensitive individual information. Hence, it is natural to ask what can be learned while preserving individual privacy. [Kasiviswanathan, Lee, Nissim, Raskhodnikova, and Smith; FOCS 2008] initiated such a discussion. They formalized the notion of *private learning*, as a combination of PAC learning and differential privacy, and investigated what concept classes can be learned privately. Somewhat surprisingly, they showed that, ignoring time complexity, every PAC learning task could be performed privately with polynomially many samples, and in many natural cases this could even be done in polynomial time.

While these results seem to equate non-private and private learning, there is still a significant gap: the sample complexity of (non-private) PAC learning is crisply characterized in terms of the VC-dimension of the concept class, whereas this relationship is lost in the constructions of private learners, which exhibit, generally, a higher sample complexity. Looking into this gap, we examine several private learning tasks and give tight bounds on their sample complexity. In particular, we show strong separations between sample complexities of proper and improper private learners (such separation does not exist for non-private learners), and between sample complexities of efficient and inefficient proper private learners. Our results show that VC-dimension is not the right measure for characterizing the sample complexity of proper private learning.

We also examine the task of *private data release* (as initiated by [Blum, Ligett, and Roth; STOC 2008]), and give new lower bounds on the sample complexity. Our results show that the logarithmic dependence on size of the instance space is essential for private data release.

1 Introduction

Consider a scenario in which a survey is conducted among a sample of random individuals and datamining techniques are applied to learn information on the

⋆ Research partially supported by the Israel Science Foundation (grant No. 938/09) and by the Frankel Center for Computer Science.

⋆⋆ Research partly supported by the Israel Science Foundation (grant No. 860/06).

entire population. If such information will disclose information on the individuals participating in the survey, then they will be reluctant to participate in the survey. To address this question, Kasiviswanathan *et al.* [10] introduced the notion of *private learning*. Informally, a private learner is required to output a hypothesis that gives accurate classification while protecting the privacy of the individual samples from which the hypothesis was obtained. The formal notion of a private learner is a combination of two qualitatively different notions. One is that of PAC learning [17], the other of differential privacy [7]. In PAC (probably approximately correct) learning, a collection of samples (labeled examples) is generalized into a hypothesis. It is assumed that the examples are generated by sampling from some (unknown) distribution $\mathcal{D}$ and are labeled according to an (unknown) concept c taken from some concept class $\mathcal{C}$. The learned hypothesis h should predict with high accuracy the labeling of examples taken from the distribution $\mathcal{D}$, an *average-case* requirement. Differential privacy, on the other hand, is formulated as a *worst-case* requirement. It requires that the output of a learner should not be significantly affected if a particular example d is replaced with arbitrary d', for all d and d'. This strong notion provides rigorous privacy guarantees even against attackers empowered with arbitrary side information [11].

Recent research on privacy has shown, somewhat surprisingly, that it is possible to design differentially private variants of many analyses (see [6] for a recent survey). In this line, the work of [10] demonstrated that private learning is generally feasible – any concept class that is PAC learnable can be learned privately (but not necessarily efficiently), by a "Private Occam's Razor" algorithm, with sample complexity that is logarithmic in the size of the hypothesis class. Furthermore, taking into account the earlier result of [2] (that all concept classes that can be efficiently learned in the *statistical queries* model can be learned privately and efficiently) and the efficient private parity learner of [10], we get that most "natural" computational learning tasks can be performed privately and efficiently (i.e., with polynomial resources). This is important as learning problems generalize many of the computations performed by analysts over collections of sensitive data.

The results of [2, 10] show that private learning is feasible in an extremely broad sense, and hence one can essentially equate learning and private learning. However, the costs of the private learners constructed in [2, 10] are generally higher than those of non-private ones by factors that depend not only on the privacy, accuracy, and confidence parameters of the private learner. In particular, the well-known relationship between the sampling complexity of PAC learners and the VC-dimension of the concept class (ignoring computational efficiency) [5] does not hold for the above constructions of private learners – as their sample complexity is proportional to the logarithm of the size of the concept class. Recall that the VC-dimension of a concept class is bounded by the logarithm of its size, and is significantly lower for many interesting concept classes, hence there may exist learning tasks for which "very practical" non-private learner exists, but any private learner is "impractical".

The focus of this work is on a fine-grain examination of the differences in complexity between private and non-private learning. The hope is that such an examination will eventually lead to an understanding of which complexity measure is relevant for the sample complexity of private learning, similar to the well-understood relationship between the VC-dimension and sample complexity of PAC learning. Such an examination is interesting also for other tasks, and a second task we examine is that of releasing a *sanitization* of a data set that simultaneously protects privacy of individual contributors and offers utility to the data analyst. See the discussion in Section 1.1.

1.1 Our Contributions

We now give a brief account of our results. Throughout this rather informal discussion we will treat the accuracy, confidence, and privacy parameters as constants (a detailed analysis revealing the dependency on these parameters is presented in the technical sections). We use the term "efficient" for polynomial time computations.

Following standard computational learning terminology, we will call learners for a concept class $\mathcal{C}$ that only output hypotheses in $\mathcal{C}$ *proper*, and other learners *improper*. The original motivation in computational learning theory for this distinction is that there exist concept classes $\mathcal{C}$ for which proper learning is computationally intractable [16], whereas it is possible to efficiently learn $\mathcal{C}$ improperly [17]. As we will see below, the distinction between proper and improper learning is useful also when discussing private learning, and for more reasons than making intractable learning tasks tractable.

Proper and Improper Private Learning. It is instructive to look into the construction of the Private Occam's Razor algorithm of [10] and see why its sample complexity is proportional to the logarithm of the size of the hypothesis class used. The algorithm uses the exponential mechanism of McSherry and Talwar [14] to choose a hypothesis. The choice is probabilistic, where the probability mass that is assigned to each of the hypotheses decreases exponentially with the number of samples that are inconsistent with it. A union-bound argument is used in the claim that the construction actually yields a learner, and a sample size that is logarithmic in the size of the hypothesis class is needed for the argument to go through.

For our analyses in this paper, we consider a simple, but natural, class $POINT_d$ containing the concepts $c_j : \{0,1\}^d \to \{0,1\}$ where $c_j(x) = 1$ for $x = j$, and 0 otherwise. The VC-dimension of $POINT_d$ is one, and hence it can be learned (non-privately and efficiently, properly or improperly) with merely $O(1)$ samples.

In sharp contrast, (when used for properly learning $POINT_d$) the Private Occam's Razor algorithm requires $O(\log |POINT_d|) = O(d)$ samples – obtaining the largest possible gap in sample complexity when compared to non-private learners! Our first result is a matching lower bound. We prove that *any* proper private learner for $POINT_d$ must use $\Omega(d)$ samples, therefore, answering negatively the question (from [10]) of whether proper private learners should exhibit sample complexity that is approximately the VC-dimension (or even a function of the VC-dimension) of the concept class[4].

A natural way to improve on the sample complexity is to use the Private Occam's Razor to improperly learn $POINT_d$ with a smaller hypothesis class that is still expressive enough for $POINT_d$, reducing the sample complexity to the logarithm of the smaller hypothesis class. We show that this indeed is possible, as there exists a hypothesis class of size $O(d)$ that can be used for learning $POINT_d$ improperly, yielding an algorithm with sample complexity $O(\log d)$. Furthermore, this bound is tight, any hypothesis class for learning $POINT_d$ must contain $\Omega(d)$ hypotheses. These bounds are interesting as they give a separation between proper and improper private learning – proper private learning of $POINT_d$ requires $\Omega(d)$ samples, whereas $POINT_d$ can be improperly privately learned using $O(\log d)$ samples. Note that such a combinatorial separation does not exist for non-private learning, as a VC-dimension number of samples are needed and sufficient for both proper and improper non-private learners. Furthermore, the $\Omega(d)$ lower bound on the size of the hypothesis class maps a clear boundary to what can be achieved in terms of sample complexity using the Private Occam's Razor for $POINT_d$. It might even suggest that *any* private learner for $POINT_d$ should use $\Omega(\log d)$ samples.

It turns out, however, that the intuition expressed in the last sentence is at fault. We construct an efficient improper private learner for $POINT_d$ that uses merely $O(1)$ samples, hence establishing the strongest possible separation between proper and improper private learners. For the construction we extrapolate on a technique from the efficient private parity learner of [10]. The construction of [10] utilizes a natural non-private proper learner, and hence results in a proper private learner, whereas, due to the bounds mentioned above, we cannot use a proper learner for $POINT_d$, and hence we construct an improper (rather unnatural) learner to base our construction upon. Our construction utilizes a double-exponential hypothesis class, and hence is inefficient (even outputting a hypothesis requires super-polynomial time). We use a simple compression using pseudorandom functions (akin to [15]) to make the algorithm efficient.

Efficient and Inefficient Proper Private Learning. We apply the above lower bound on the number of samples for proper private learning $POINT_d$ to show a separation in the sample size between efficient and inefficient proper private learning. Assuming the existence of pseudorandom generators with exponential stretch, we present a concept class $\widehat{POINT}_d$ – a variant of $POINT_d$ – such that every efficient proper private learner for this class requires $\Omega(d)$ samples. In contrast, an inefficient proper private learner exists that uses only a super-logarithmic number of samples. This is the first example where requiring efficiency on top of privacy comes at a price of larger sample size.

The Sample Size of Non-Interactive Sanitization Mechanisms. Given a database containing a collection of individual information, a sanitization is

[4] Our proof technique yields lower bounds not only on private learning $POINT_d$ properly, but on private learning of any concept class $\mathcal{C}$ with various hypothesis classes that we call α-minimal for $\mathcal{C}$.

a release that protects the privacy of the individual contributors while offering utility to the analyst using the database. The setting is non-interactive if once the sanitization is released the original database and the curator play no further role. Blum *et al.* [3] presented a construction of such non-interactive sanitizers for count queries. Let $\mathcal{C}$ be a concept class consisting of efficiently computable predicates from a discretized domain X to $\{0,1\}$. Given a collection D of data items taken from X, Blum *et al.* employ the exponential mechanism [14] to (inefficiently) obtain another collection D' with data items from X such that D' maintains approximately correct count of $\sum_{d \in D} c(d)$ for all concepts $c \in \mathcal{C}$. Also, they show that it suffices for D to have a size that is $O(\log |X| \cdot \mathit{VCDIM}(\mathcal{C}))$. The database D' is referred to as a *synthetic* database as it contains data items drawn from the same universe (i.e., from X) as the original database D.

We provide new lower bounds for non-interactive sanitization mechanisms. We show that for POINT_d every non-interactive sanitization mechanism that is useful[5] for POINT_d requires a database of $\Omega(d)$ size. This lower bound is tight as the sanitization mechanism of Blum *et al.* for POINT_d uses a database of $O(d \cdot \mathit{VCDIM}(\mathit{POINT}_d)) = O(d)$ size. Our lower bound holds even if the sanitized output is an arbitrary data structure and not a synthetic database.

1.2 Related Work

The notion of PAC learning was introduced by Valiant [17]. The notion of differential privacy was introduced by Dwork *et al.* [7]. Private learning was introduced in [10]. Beyond proving that (ignoring computation) every concept class can be PAC learned privately (see Theorem 2 below), they proved an equivalence between learning in the statistical queries model and private learning in the local communication model (aka randomized response). The general private data release mechanism we mentioned above was introduced in [3] along with a specific construction for halfspace queries. As we mentioned above, both [10] and [3] use the exponential mechanism of [14], a generic construction of differential private analyses, that (in general) does not yield efficient algorithms.

A recent work of Dwork *et al.* [8] considered the complexity of non-interactive sanitization under two settings: (a) sanitized output is a synthetic database, and (b) sanitized output is some arbitrary data structure. For the task of sanitizing with a synthetic database they show a separation between efficient and inefficient sanitization mechanisms based on whether the size of the instance space and the size of the concept class is polynomial in a (security) parameter or not. For the task of sanitizing with an arbitrary data structure they show a tight connection between the complexity of sanitization and traitor tracing schemes used in cryptography. They leave the problem of separating efficient private and inefficient private learning open.

It is well known that for all concept classes $\mathcal{C}$, every learner for $\mathcal{C}$ requires $\Omega(\mathit{VCDIM}(\mathcal{C}))$ samples [9]. This lower bound on the sample size also holds for

[5] Informally, a mechanism is useful for a concept class if for every input, the output of the mechanism maintains approximately correct counts for all concepts in the concept class.

private learning. Blum, Ligett, and Roth [4] have recently extended this result to the setting of private data release. They show that for all concept classes $\mathcal{C}$, every non-interactive sanitization mechanism that is useful for $\mathcal{C}$ requires $\Omega(\mathit{VCDIM}(\mathcal{C}))$ samples. We show in Section 4 that this bound is not tight – there exists a concept class $\mathcal{C}$ of constant VC-dimension such that every non-interactive sanitization mechanism that is useful for $\mathcal{C}$ requires a much larger sample size.

2 Preliminaries

Notation. We use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. The notation $O_\gamma(g(n))$ is a shorthand for $O(h(\gamma) \cdot g(n))$ for some non-negative function h. Similarly, the notation $\Omega_\gamma(g(n))$. We use $\mathrm{negl}(\cdot)$ to denote functions from $\mathbb{R}^+$ to $[0,1]$ that decrease faster than any inverse polynomial.

2.1 Preliminaries from Privacy

A database is a vector $D = (d_1, \ldots, d_m)$ over a domain X, where each entry $d_i \in D$ represents information contributed by one individual. Databases D and D' are called *neighbors* if they differ in exactly one entry (i.e., the Hamming distance between D and D' is 1). An algorithm is private if neighboring databases induce nearby distributions on its outcomes. Formally:

Definition 1 (Differential Privacy [7]). *A randomized algorithm $\mathcal{A}$ is ϵ-differentially private if for all neighboring databases D, D', and for all sets $\mathcal{S}$ of outputs,*

$$\Pr[\mathcal{A}(D) \in \mathcal{S}] \leq \exp(\epsilon) \cdot \Pr[\mathcal{A}(D') \in \mathcal{S}]. \tag{1}$$

The probability is taken over the random coins of $\mathcal{A}$.

An immediate consequence of Equation (1) is that for any two databases D, D' (not necessarily neighbors) of size m, and for all sets $\mathcal{S}$ of outputs, $\Pr[\mathcal{A}(D) \in \mathcal{S}] \geq \exp(-\epsilon m) \cdot \Pr[\mathcal{A}(D') \in \mathcal{S}]$.

2.2 Preliminaries from Learning Theory

We consider Boolean classification problems. A concept is a function that labels *examples* taken from the domain X by the elements of the range $\{0,1\}$. The domain X is understood to be an ensemble $X = \{X_d\}_{d\in\mathbb{N}}$. A *concept class* $\mathcal{C}$ is a set of concepts, considered as an ensemble $\mathcal{C} = \{\mathcal{C}_d\}_{d\in\mathbb{N}}$ where $\mathcal{C}_d$ is a class of concepts from $\{0,1\}^d$ to $\{0,1\}$.

A concept class comes implicitly with a way to represent concepts and $\mathrm{size}(c)$ is the size of the (smallest) representation of c under the given representation scheme. Let $\mathcal{D}$ be a distribution on X_d. PAC learning algorithms are designed assuming a promise that the examples are labeled consistently with some *target* concept c from a class $\mathcal{C}$. Define,

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

Definition 2 (PAC Learning [17]). *An algorithm $\mathcal{A}$ is an (α, β)-*PAC learner *of a concept class $\mathcal{C}_d$ over X_d using hypothesis class $\mathcal{H}_d$ and sample size n if for all concepts $c \in \mathcal{C}_d$, all distributions $\mathcal{D}$ on X_d, given an input $D = (d_1, \cdots, d_n)$, where $d_i = (x_i, c(x_i))$ and x_i are drawn i.i.d. from $\mathcal{D}$ for $i \in [n]$, algorithm $\mathcal{A}$ outputs a hypothesis $h \in \mathcal{H}_d$ satisfying $\Pr[\mathrm{error}_{\mathcal{D}}(c, h) \leq \alpha] \geq 1 - \beta$. The probability is taken over the random choice of the examples D and the coin tosses of the learner.*

An algorithm $\mathcal{A}$, whose inputs are d, α, β, and a set of samples (labeled examples) D, is a PAC learner *of a concept class $\mathcal{C} = \{\mathcal{C}_d\}_{d\in\mathbb{N}}$ over $X = \{X_d\}_{d\in\mathbb{N}}$ using hypothesis class $\mathcal{H}_d = \{\mathcal{H}_d\}_{d\in\mathbb{N}}$ if there exists a polynomial $p(\cdot,\cdot,\cdot,\cdot)$ such that for all $d \in \mathbb{N}$ and $0 < \alpha, \beta < 1$, the algorithm $\mathcal{A}(d, \alpha, \beta, \cdot)$ is an (α, β)-PAC learner of the concept class $\mathcal{C}_d$ over X_d using hypothesis class $\mathcal{H}_d$ and sample size $n = p(d, \mathrm{size}(c), 1/\alpha, \log(1/\beta))$. If $\mathcal{A}$ runs in time polynomial in $d, \mathrm{size}(c), 1/\alpha, \log(1/\beta)$, we say that it is an* efficient PAC learner. *Also, the learner is called a* proper *PAC learner if $\mathcal{H} = \mathcal{C}$, otherwise it is called an* improper *PAC learner.*

A concept class $\mathcal{C} = \{\mathcal{C}_d\}_{d\in\mathbb{N}}$ over $X = \{X_d\}_{d\in\mathbb{N}}$ is PAC learnable *using hypothesis class $\mathcal{H} = \{\mathcal{H}_d\}_{d\in\mathbb{N}}$ if there exists a PAC learner $\mathcal{A}$ learning $\mathcal{C}$ over X using hypothesis class $\mathcal{H}$. If $\mathcal{A}$ is an efficient PAC learner, we say that $\mathcal{C}$ is efficiently PAC learnable.*

It is well known that improper learning is more powerful than proper learning. For example, Pitt and Valiant [16] show that unless **RP**=**NP**, k-term DNF formulae are not learnable by k-term DNF, whereas it is possible to learn a k-term DNF using k-CNF [17]. For more background on learning theory, see, e.g., [13].

Definition 3 (VC-Dimension [18]). *Let $\mathcal{C} = \{\mathcal{C}_d\}$ be a class of concepts over $X = \{X_d\}$. We say that $\mathcal{C}_d$ shatters a point set $Y \subset X_d$ if $|\{c(Y) : c \in \mathcal{C}_d\}| = 2^{|Y|}$, i.e., the concepts in $\mathcal{C}_d$ when restricted to Y produce all the $2^{|Y|}$ possible assignments on Y. the VC-dimension of $\mathcal{C}$ is defined as the size of the maximum point set that is shattered by $\mathcal{C}_d$, as a function of d.*

Theorem 1 ([5]). *A concept class $\mathcal{C} = \{\mathcal{C}_d\}$ over $X = \{X_d\}$ is PAC learnable using $\mathcal{C}$ by a PAC learner $\mathcal{A}$ that uses $O((\mathit{VCDIM}(\mathcal{C}_d) \cdot \log \frac{1}{\alpha} + \log \frac{1}{\beta})/\alpha)$ samples.*

2.3 Private Learning

Definition 4 (Private PAC Learning [10]). *Let d, α, β be as in Definition 2 and $\epsilon > 0$. Concept class $\mathcal{C}$ is ϵ-differentially privately PAC learnable using $\mathcal{H}$ if there exists an algorithm $\mathcal{A}$ that takes inputs $\epsilon, d, \alpha, \beta, D$, where n, the number of samples (labeled examples) in D is polynomial in $1/\epsilon, d, \mathrm{size}(c), 1/\alpha, \log(1/\beta)$, and satisfies*

PRIVACY. *For all d and $\epsilon, \alpha, \beta > 0$, algorithm $\mathcal{A}(\epsilon, d, \alpha, \beta, \cdot)$ is ϵ-differentially private (Definition 1);*

Utility. *For all $\epsilon > 0$, algorithm $\mathcal{A}(\epsilon, \cdot, \cdot, \cdot, \cdot)$ PAC learns $\mathcal{C}$ using $\mathcal{H}$ (Definition 2).*

$\mathcal{A}$ is an efficient private PAC learner if it runs in time polynomial in $1/\epsilon$, d, $\text{size}(c)$, $1/\alpha$, $\log(1/\beta)$. Also, the private learner is called proper *if $\mathcal{H} = \mathcal{C}$, otherwise it is called* improper.

Remark 1. The privacy requirement in Definition 4 is a worst-case requirement. That is, Equation (1) must hold for every pair of neighboring databases D, D' (even if these databases are not consistent with any concept in $\mathcal{C}$). In contrast, the utility requirement is an average-case requirement, where we only require the learner to succeed with high probability over the distribution of the databases. This qualitative difference between the utility and privacy of private learners is crucial. A wrong assumption on how samples are formed that leads to a meaningless outcome can usually be replaced with a better one with very little harm. No such amendment is possible once privacy is lost due to a wrong assumption.

Note also that each entry d_i in a database D is a labeled example. That is, we protect the privacy of both the example and its label.

Observation 1. The computational separation between proper and improper learning also holds when we add the privacy constraint. That is unless **RP**=**NP** no proper private learner can learn k-term DNF, whereas there exists an efficient improper private learner that can learn k-term DNF using a k-CNF. The efficient k-term DNF learner of [17] uses statistical queries (SQ) [12] which can be simulated efficiently and privately as shown by [2, 10].

More generally, such a gap can be shown for any concept class that cannot be properly PAC learned, but can be efficiently learned (improperly) in the statistical queries model.

3 Learning vs. Private Learning

We begin by recalling the upper bound on the sample (database) size for private learning from [10]. The bound in [10] is for agnostic learning, and we restate it for (non-agnostic) PAC learning using the following notion of α-representation:

Definition 5. *We say that a hypothesis class $\mathcal{H}_d$ α-represents a concept class $\mathcal{C}_d$ over the domain X_d if for every $c \in \mathcal{C}_d$ and every distribution $\mathcal{D}$ on X_d there exists a hypothesis $h \in \mathcal{H}_d$ such that $\text{error}_{\mathcal{D}}(c, h) \leq \alpha$.*

Theorem 2 (Kasiviswanathan *et al.* [10], restated). *Assume that there is a hypothesis class $\mathcal{H}_d$ that α-represents a concept class $\mathcal{C}_d$. Then, there exists a private PAC learner for $\mathcal{C}_d$ using $\mathcal{H}_d$ that uses $O((\log |\mathcal{H}_d| + \log(1/\beta))/(\epsilon\alpha))$ labeled examples, where ϵ, α, and β are parameters of the private learner. The learner might not be efficient.*

In other words, using Theorem 2 the number of labeled examples required for learning a concept class $\mathcal{C}_d$ is logarithmic in the size of the smallest hypothesis class that α-represents $\mathcal{C}_d$. For comparison, the number of labeled examples required for learning $\mathcal{C}_d$ non-privately is proportional to the VC-dimension of $\mathcal{C}_d$ [5, 9].

3.1 Separation Between Private and Non-private PAC Learning

Our first result shows that private learners may require many more samples than non-private ones. We consider a very simple concept class of VC-dimension one, and hence is (non-privately) properly learnable using $O_{\alpha,\beta}(1)$ labeled examples. We prove that for any proper learner for this class the required number of labeled examples is at least logarithmic in the size of the concept class, matching Theorem 2.

Proving the lower bound, we show that a large collection of m-record databases $D_1, \ldots, D_N$ exists, with the property that every PAC learner has to output a different hypothesis for each of these databases (recall that in our context a database is a collection of labeled examples, supposedly drawn from some distribution and labeled consistently with some target concept).

As any two databases D_a and D_b differ on at most m entries, a private learner must, because of the differential privacy requirement, output on input D_a the hypothesis that is accurate for D_b (and not accurate for D_a) with probability at least $(1-\beta)\cdot\exp(-\epsilon m)$. Since this holds for every pair of databases, unless m is large enough we get that the private learner's output on D_a is, with high probability, a hypothesis that is not accurate for D_a. We use the following notion of α-minimality:

Definition 6. *If $\mathcal{H}_d$ α-represents $\mathcal{C}_d$, and every $\mathcal{H}'_d \subsetneq \mathcal{H}_d$ does not α-represent $\mathcal{C}_d$, then we say that $\mathcal{H}_d$ is α-minimal for $\mathcal{C}_d$.*

Theorem 3. *Let $\mathcal{H}_d$ be an α-minimal class for $\mathcal{C}_d$. Then any private PAC learner that learns $\mathcal{C}_d$ using $\mathcal{H}_d$ requires $\Omega((\log|\mathcal{H}_d| + \log(1/\beta))/\epsilon)$ labeled examples.*

Proof. Let $\mathcal{C}_d$ be over the domain X_d and let $\mathcal{H}_d$ be α-minimal for $\mathcal{C}_d$. Since for every $h \in \mathcal{H}_d$, $\mathcal{H}_d \setminus \{h\}$ does not α-represent $\mathcal{C}_d$, we get that there exists a concept $c_h \in \mathcal{C}_d$ and a distribution $\mathcal{D}_h$ on X_d such that on inputs drawn from $\mathcal{D}_h$ labeled by c_h, every PAC learner (that learns $\mathcal{C}_d$ using $\mathcal{H}_d$) has to output h with probability at least $1-\beta$.

Let $\mathcal{A}$ be a private learner that learns $\mathcal{C}_d$ using $\mathcal{H}_d$, and suppose $\mathcal{A}$ uses m labeled examples. For every $h \in \mathcal{H}_d$, note that there exists a database $D_h \in X_d^m$ on which $\mathcal{A}$ has to output h with probability at least $1-\beta$. To see that, note that if $\mathcal{A}$ is run on m examples chosen i.i.d. from the distribution $\mathcal{D}_h$ and labeled according to c_h, then $\mathcal{A}$ outputs h with probability at least $1-\beta$ (where the probability is over the sampling from $\mathcal{D}_h$ and over the randomness of $\mathcal{A}$). Hence, a collection of m labeled examples over which $\mathcal{A}$ outputs h with probability $1-\beta$ exists, and D_h can be set to contain these m labeled examples.

Take $h, h' \in \mathcal{H}_d$ such that $h \neq h'$ and consider the two corresponding databases D_h and $D_{h'}$ with m entries each. Clearly, they differ in at most m entries, and hence we get by differential privacy of $\mathcal{A}$ that

$$\Pr[\mathcal{A}(D_h) = h'] \geq \exp(-\epsilon m) \cdot \Pr[\mathcal{A}(D_{h'}) = h'] \;\geq\; \exp(-\epsilon m) \cdot (1-\beta).$$

Since the above inequality holds for every pair of databases, we fix any h and get,

$$\begin{aligned}\Pr[\mathcal{A}(D_h) \neq h] = \Pr[\mathcal{A}(D_h) \in \mathcal{H}_d \setminus \{h\}] &= \sum_{h' \in \mathcal{H}_d \setminus \{h\}} \Pr[\mathcal{A}(D_h) = h'] \\ &\geq (|\mathcal{H}_d| - 1) \cdot \exp(-\epsilon m) \cdot (1-\beta).\end{aligned}$$

On the other hand, we chose D_h such that $\Pr[\mathcal{A}(D_h) = h] \geq 1 - \beta$, equivalently, $\Pr[\mathcal{A}(D_h) \neq h] \leq \beta$. We hence get that $(|\mathcal{H}_d| - 1) \cdot \exp(-\epsilon m) \cdot (1-\beta) \leq \beta$. Solving the last inequality for m, we get $m = \Omega((\log |\mathcal{H}_d| + \log(1/\beta))/\epsilon)$ as required. □

Using Theorem 3, we now prove a lower bound on the number of labeled examples needed for proper private learning a specific concept class. Let $T = 2^d$ and $X_d = \{1, \ldots, T\}$. Define the concept class $POINT_d$ to be the set of points over $\{1, \ldots, T\}$:

Definition 7 (Concept Class $POINT_d$). *For $j \in [T]$ define $c_j : [T] \to \{0,1\}$ as $c_j(x) = 1$ if $x = j$, and $c_j(x) = 0$ otherwise. $POINT_d = \{c_j\}_{j \in [T]}$.*

We note that we use the set $\{1, \ldots, T\}$ for notational convenience only. We never use the fact that the set elements are integer numbers.

Proposition 1. *$POINT_d$ is α-minimal for itself.*

Proof. Clearly, $POINT_d$ α-represents itself. To show minimality, consider a subset $\mathcal{H}'_d \subsetneq POINT_d$, where $c_i \notin \mathcal{H}'_d$. Note that under the distribution $\mathcal{D}$ that chooses i with probability one, $\mathrm{error}_{\mathcal{D}}(c_i, c_j) = 1$ for all $j \neq i$. Hence, $\mathcal{H}'_d$ does not α-represent $POINT_d$. □

The VC-dimension of $POINT_d$ is one[6]. It is well known that a standard (non-private) proper learner uses approximately VC-dimension number of labeled examples to learn a concept class [5]. In contrast, we get that far more labeled examples are needed for any proper private learner for $POINT_d$. The following corollary follows directly from Theorem 3 and Proposition 1:

Corollary 1. *Every proper private PAC learner for $POINT_d$ requires $\Omega((d + \log(1/\beta))/\epsilon)$ labeled examples.*

[6] Note that every singleton $\{j\}$ where $j \in [T]$ is shattered by $POINT_d$ as $c_j(j) = 1$ and $c_{j'}(j) = 0$ for all $j' \neq j$. No set of two points $\{j, j'\}$ is shattered by $POINT_d$ as $c_{j''}(j) = c_{j''}(j') = 1$ for no $j'' \in [T]$.

Remark 2. We note that the lower bound for $POINT_d$ can be improved to $\Omega((d+\log(1/\beta))/(\epsilon\alpha))$ labeled examples, matching the upper bound from Theorem 2. Also, the proper learner for $POINT_d$ from Theorem 2 can be made efficient. Details are deferred to the full version [1].

We conclude this section showing that every hypothesis class $\mathcal{H}$ that α-represents $POINT_d$ should have at least d hypotheses. Therefore, if we use Theorem 2 to learn $POINT_d$ we need $\Omega(\log d)$ labeled examples. At first sight, it may seem that the relationship between $|\mathcal{H}|$ and the sample complexity is essential, and hence, the number of labeled examples needed for *every* private PAC learner for $POINT_d$ is super-constant. However, this turns out not to be the case. In Section 3.2, we present a private learner for $POINT_d$ that uses $O_{\alpha,\beta,\epsilon}(1)$ labeled examples. For this construction, we use techniques that are very different from those used in the proof of Theorem 2. In particular, our private learner uses a very large hypothesis class.

Lemma 1. *Let $\alpha < 1/2$. $|\mathcal{H}| \geq d$ for every hypothesis class $\mathcal{H}$ that α-represents $POINT_d$.*

Proof. Let $\mathcal{H}$ be a hypothesis class with $|\mathcal{H}| < d$. Consider a table whose $T = 2^d$ columns correspond to the possible 2^d inputs $1, \ldots, T$, and whose $|\mathcal{H}|$ rows correspond to the hypothesis in $\mathcal{H}$. The (i, j)th entry is 0 or 1 depending on whether the ith hypothesis gives 0 or 1 on input j. Since $|\mathcal{H}| < d = \log T$, at least two columns $j \neq j'$ are identical. That is, $h(j) = h(j')$ for every $h \in \mathcal{H}$. Consider the concept $c_j \in POINT_d$ (defined as $c_j(x) = 1$ if $x = j$, and 0 otherwise), and the distribution $\mathcal{D}$ with probability mass $1/2$ on both j and j'. We get that $\text{error}_{\mathcal{D}}(c_j, h) \geq 1/2 > \alpha$ for all $h \in \mathcal{H}$ (since any hypothesis either errs on j or on j'). Therefore, $\mathcal{H}$ does not α-represent $POINT_d$. □

3.2 Separation Between Proper and Improper Private PAC Learning

We now use $POINT_d$ to show a separation between proper and improper private PAC learning. We show that $POINT_d$ can be privately (and efficiently) learned by an improper learner using $O_{\alpha,\beta,\epsilon}(1)$ labeled examples. We begin by presenting a non-private improper PAC learner $\mathcal{A}_1$ for $POINT_d$ that succeeds with only constant probability. Roughly, $\mathcal{A}_1$ applies a simple proper learner for $POINT_d$, and then modifies its outcome by adding random "noise". We then use sampling to convert $\mathcal{A}_1$ into a private learner $\mathcal{A}_2$, and like $\mathcal{A}_1$ the probability that $\mathcal{A}_2$ succeeds in learning $POINT_d$ is only a constant. Later we amplify the success probability of $\mathcal{A}_2$ to get a private PAC learner. Both $\mathcal{A}_1$ and $\mathcal{A}_2$ are inefficient as they output hypotheses with exponential description length. However, using a pseudorandom function it is possible to compress the outputs of $\mathcal{A}_1$ and $\mathcal{A}_2$, and hence achieve efficiency.

Algorithm $\mathcal{A}_1$. Given labeled examples $(x_1, y_1), \ldots, (x_m, y_m)$, algorithm $\mathcal{A}_1$ performs the following:

1. If $(x_1, y_1), \ldots, (x_m, y_m)$ are not consistent with any concept in $POINT_d$, return $\perp$ (this happens only if $x_i \neq x_j$ and $y_i = y_j = 1$ for some $i, j \in [m]$ or if $x_i = x_j$ and $y_i \neq y_j$).
2. If $y_i = 0$ for all $i \in [m]$, then let $c = \mathbf{0}$ (the all zero hypothesis); otherwise, let c be the (unique) hypothesis from $POINT_d$ that is consistent with the m input labeled examples.
3. Modify c at random to get a hypothesis h by letting $h(x) = c(x)$ with probability $1 - \alpha/8$, and $h(x) = 1 - c(x)$ otherwise for all $x \in [T]$. Return h.

Let $m = 2\ln(4)/\alpha$. We next argue that if m examples are drawn i.i.d. according to a distribution $\mathcal{D}$ on $[T]$, and the examples are labeled consistently according to some $c_j \in POINT_d$, then $\Pr[\mathrm{error}_{\mathcal{D}}(c_j, c) > \alpha/2] \leq 1/4$. If the examples are labeled consistently according to some $c_j \neq \mathbf{0}$, then $c \neq c_j$ only if $(j, 1)$ is not in the sample and in this case $c = \mathbf{0}$. If $\Pr_{x\sim\mathcal{D}}[x = j] \leq \alpha/2$ and $(j, 1)$ is not in the sample, then $c = \mathbf{0}$ and $\mathrm{error}_{\mathcal{D}}(c_j, \mathbf{0}) \leq \alpha/2$. If $\Pr_{x\sim\mathcal{D}}[x = j] \leq \alpha/2$ and $(j, 1)$ is in the sample, then $c = c_j$ and $\mathrm{error}_{\mathcal{D}}(c_j, c) = 0$. Otherwise if $\Pr_{x\sim\mathcal{D}}[x = j] > \alpha/2$, the probability that all m examples are not $(j, 1)$ is at most $(1 - \alpha/2)^m = ((1 - \alpha/2)^{2/\alpha})^{\ln 4} \leq 1/4$.

To see that $\mathcal{A}_1$ PAC learns $POINT_d$ (with accuracy α and confidence 1/4) note that

$$\mathop{\mathbb{E}}_{h}[\mathop{\mathrm{error}}_{\mathcal{D}}(c, h)] = \mathop{\mathbb{E}}_{h} \mathop{\mathbb{E}}_{x\sim\mathcal{D}}[|h(x) - c(x)|] = \mathop{\mathbb{E}}_{x\sim\mathcal{D}} \mathop{\mathbb{E}}_{h}[|h(x) - c(x)|] = \frac{\alpha}{8},$$

and hence, using Markov's Inequality, $\Pr_h[\mathrm{error}_{\mathcal{D}}(c, h) > \alpha/2] \leq 1/4$. Combining this with $\Pr[\mathrm{error}_{\mathcal{D}}(c_j, c) > \alpha/2] \leq 1/4$ and $\mathrm{error}_{\mathcal{D}}(c_j, h) \leq \mathrm{error}_{\mathcal{D}}(c_j, c) + \mathrm{error}_{\mathcal{D}}(c, h)$, implies that $\Pr[\mathrm{error}_{\mathcal{D}}(c_j, h) > \alpha] \leq 1/2$.

Algorithm $\mathcal{A}_2$. We now modify learner $\mathcal{A}_1$ to get a private learner $\mathcal{A}_2$ (a similar idea was used in [10] for learning parity functions). Given labeled examples $(x_1, y_1), \ldots, (x_{m'}, y_{m'})$, algorithm $\mathcal{A}_2$ performs the following:

1. With probability $\alpha/8$, return $\perp$.
2. Construct a set $S \subseteq [m']$ by picking each element of $[m']$ with probability $p = \alpha/4$. Run the non-private learner $\mathcal{A}_1$ on the examples indexed by S.

We first show that, given $m' = 8m/\alpha$ labeled examples, $\mathcal{A}_2$ PAC learns $POINT_d$ with confidence $\Theta(1)$. Note that, by Chernoff bound, $\Pr[|S| \leq m] \leq \exp(-m/4) = O_\alpha(1)$. Therefore, we get that $\mathcal{A}_2$ PAC learns $POINT_d$ with accuracy parameter $\alpha' = \alpha$ and confidence parameter $\beta' = 1/2 + \alpha/8 + \exp(-m/4) = \Theta(1)$. We now show that $\mathcal{A}_2$ is ϵ^*-differentially private with bounded ϵ^*.

Claim. Algorithm $\mathcal{A}_2$ is ϵ^*-differentially private, where $\epsilon^* = \ln(4)$.

Proof. Let D, D' be two neighboring databases, and assume that they differ on the ith entry. First let us analyze the probability of $\mathcal{A}_2$ outputting $\perp$:

$$\begin{aligned}
\frac{\Pr[\mathcal{A}_2(D) = \perp]}{\Pr[\mathcal{A}_2(D') = \perp]} &= \frac{p \cdot \Pr[\mathcal{A}_2(D) = \perp \mid i \in S] + (1-p) \cdot \Pr[\mathcal{A}_2(D) = \perp \mid i \notin S]}{p \cdot \Pr[\mathcal{A}_2(D') = \perp \mid i \in S] + (1-p) \cdot \Pr[\mathcal{A}_2(D') = \perp \mid i \notin S]} \\
&\leq \frac{p \cdot 1 + (1-p) \cdot \Pr[\mathcal{A}_2(D) = \perp \mid i \notin S]}{p \cdot 0 + (1-p) \cdot \Pr[\mathcal{A}_2(D') = \perp \mid i \notin S]} \\
&= \frac{p}{(1-p) \cdot \Pr[\mathcal{A}_2(D') = \perp \mid i \notin S]} + 1 \leq \frac{8p}{\alpha(1-p)} + 1,
\end{aligned}$$

where the last equality follows noting that if $i \notin S$ then $\mathcal{A}_2$ is equally likely to output $\perp$ on D and D', and the last inequality follows as $\perp$ is returned with probability $\alpha/8$ in Step 1 of Algorithm $\mathcal{A}_2$.

For the more interesting case, where $\mathcal{A}_2$ outputs a hypothesis h, we get:

$$\begin{aligned}
\frac{\Pr[\mathcal{A}_2(D) = h]}{\Pr[\mathcal{A}_2(D') = h]} &= \frac{p \cdot \Pr[\mathcal{A}_2(D) = h \mid i \in S] + (1-p) \cdot \Pr[\mathcal{A}_2(D) = h \mid i \notin S]}{p \cdot \Pr[\mathcal{A}_2(D') = h \mid i \in S] + (1-p) \cdot \Pr[\mathcal{A}_2(D') = h \mid i \notin S]} \\
&\leq \frac{p \cdot \Pr[\mathcal{A}_2(D) = h \mid i \in S] + (1-p) \cdot \Pr[\mathcal{A}_2(D) = h \mid i \notin S]}{p \cdot 0 + (1-p) \cdot \Pr[\mathcal{A}_2(D') = h \mid i \notin S]} \\
&= \frac{p}{1-p} \cdot \frac{\Pr[\mathcal{A}_2(D) = h \mid i \in S]}{\Pr[\mathcal{A}_2(D) = h \mid i \notin S]} + 1,
\end{aligned}$$

where the last equality uses the fact that if $i \notin S$ then $\mathcal{A}_2$ is equally likely to output h on D and D'. To conclude our proof, we need to bound the ratio of $\Pr[\mathcal{A}_2(D) = h \mid i \in S]$ to $\Pr[\mathcal{A}_2(D) = h \mid i \notin S]$.

$$\begin{aligned}
&\frac{\Pr[\mathcal{A}_2(D) = h \mid i \in S]}{\Pr[\mathcal{A}_2(D) = h \mid i \notin S]} \\
&= \frac{\sum_{R \subseteq [m'] \setminus \{i\}} \Pr[\mathcal{A}_2(D) = h \mid S = R \cup \{i\}] \cdot \Pr[\mathcal{A}_2 \text{ selects } R \text{ from } [m'] \setminus \{i\}]}{\sum_{R \subseteq [m'] \setminus \{i\}} \Pr[\mathcal{A}_2(D) = h \mid S = R] \cdot \Pr[\mathcal{A}_2 \text{ selects } R \text{ from } [m'] \setminus \{i\}]} \\
&\leq \max_{R \subseteq [m'] \setminus \{i\}} \frac{\Pr[\mathcal{A}_2(D) = h \mid S = R \cup \{i\}]}{\Pr[\mathcal{A}_2(D) = h \mid S = R]}.
\end{aligned}$$

Now, having or not having access to (x_i, y_i) can only affect the choice of $h(x_i)$, and since, $\mathcal{A}_1$ flips the output with probability $\alpha/8$, we get

$$\max_{R \subseteq [m'] \setminus \{i\}} \frac{\Pr[\mathcal{A}_2(D) = h \mid S = R \cup \{i\}]}{\Pr[\mathcal{A}_2(D) = h \mid S = R]} \leq \frac{1 - \alpha/8}{\alpha/8} \leq \frac{8}{\alpha}.$$

Putting everything together, we get

$$\frac{\Pr[\mathcal{A}_2(D) = h]}{\Pr[\mathcal{A}_2(D') = h]} \leq \frac{8p}{\alpha(1-p)} + 1 = \frac{8}{(4-\alpha)} + 1 < 3 + 1 = e^{\epsilon^*}.$$

□

We can reduce ϵ^* to any desired ϵ using the following simple lemma (implicit in [10], see proof in [1]):

Lemma 2. *Let $\mathcal{A}$ be an ϵ^*-differentially private algorithm. Construct an algorithm $\mathcal{B}$ that on input a database $D = (d_1, \ldots, d_n)$ constructs a new database D_s whose ith entry is d_i with probability $f(\epsilon, \epsilon^*) = (\exp(\epsilon) - 1)/(\exp(\epsilon^*) + \exp(\epsilon) - \exp(\epsilon - \epsilon^*) - 1)$ and $\perp$ otherwise, and then runs $\mathcal{A}$ on D_s. Then, $\mathcal{B}$ is ϵ-differentially private.*

It is clearly possible to incorporate the sampling in the lemma directly in Step 2 of $\mathcal{A}_2$ (note that for small ϵ, $f(\epsilon, \epsilon^*) \approx \epsilon/(\exp(\epsilon^*) - 1)$). We get that the number of labeled examples required to get a private learner with confidence parameter $\Theta(1)$ is $O_{\alpha,\epsilon}(1)$. The confidence parameter of the learner can be boosted privately from $\Theta(1)$ to any value $\beta > 0$ as explained in [10]. In doing this boosting, the number of labeled examples required for the learner increases by a factor of $O(\log(1/\beta))$. Therefore, we get that a sample size that is polynomial in $1/\epsilon, 1/\alpha$, and $\log(1/\beta)$ is sufficient to learn $POINT_d$ improperly with privacy parameter ϵ, accuracy parameter α, and confidence parameter β.

Making the Learner Efficient. Recall that the outcome of $\mathcal{A}_1$ (hence $\mathcal{A}_2$) is an exponentially long description of a hypothesis. We now complete our construction by compressing this description using a pseudorandom function. We use a slightly non-standard definition of (non-uniform) pseudorandom functions from binary strings of size d to bits; these pseudorandom functions can be easily constructed given regular pseudorandom functions.

Definition 8. *Let $F = \{F_d\}_{d\in\mathbb{N}}$ be a function ensemble, where for every d, F_d is a set of functions from $\{0,1\}^d$ to $\{0,1\}$. We say that the function ensemble F is q-biased pseudorandom if for every family of polynomial-size circuits with oracle access $\{C_d\}_{d\in\mathbb{N}}$, every polynomial $p(\cdot)$, and all sufficiently large d's,*

$$|\Pr[C_d^f(1^d) = 1] - \Pr[C_d^{H_d^q}(1^d) = 1]| < \frac{1}{p(d)} ,$$

where f is chosen at random from F_d and $H_d^q : \{0,1\}^d \to \{0,1\}$ is a function and the value $H_d^q(x)$ for $x \in \{0,1\}^d$ are selected i.i.d. to be 1 with probability q and 0 otherwise. The probabilities are taken over the random choice of H_d^q, and f.

For convenience, for $d \in \mathbb{N}$, we consider F_d as a set of functions from $\{1, \ldots, T\}$ to $\{0,1\}$, where $T = 2^d$. We set $q = \alpha\beta/4$ in the above definition. Using an $\alpha\beta/4$-biased pseudorandom function ensemble F, we change Step 3 of algorithm $\mathcal{A}_1$ as follows:

3'. If $c = \mathbf{0}$, let h be a random function from F_d. Otherwise (i.e., $c = c_j$ for some $j \in [T]$), let h be a random function from F_d subject to $h(j) = 1$. Return h.

Call the resulting modified algorithm $\mathcal{A}_3$. We next show that $\mathcal{A}_3$ is a PAC learner. Note that the exists a negligible function negl such that for large enough d, $|\Pr[h(x) = 1|h(j) = 1] - \alpha\beta/4| \leq \text{negl}(d)$ for every $x \in \{1, \ldots, T\}$ (as otherwise, we get a non-uniform distinguisher for the ensemble F). Thus,

$$\mathop{\mathbb{E}}_{h\in F_d}[\mathop{\mathrm{error}}_{\mathcal{D}}(c,h)] = \mathop{\mathbb{E}}_{h\in F_d}\mathop{\mathbb{E}}_{x\sim\mathcal{D}}[|h(x)-c(x)|]$$
$$\leq \mathop{\mathbb{E}}_{h\in F_d}\mathop{\mathbb{E}}_{x\sim\mathcal{D}}[h(x)] = \mathop{\mathbb{E}}_{x\sim\mathcal{D}}\mathop{\mathbb{E}}_{h\in F_d}[h(x)] \leq \frac{\alpha\beta}{4} + \mathrm{negl}(d).$$

The first inequality follows as for all $x \in [T]$, $h(x) \geq c(x)$ by our restriction on the choice of h. Thus, by the same arguments as for $\mathcal{A}_1$, Algorithm $\mathcal{A}_3$ is a PAC learner.

We next modify algorithm $\mathcal{A}_2$ by executing the learner $\mathcal{A}_3$ instead of the learner $\mathcal{A}_1$. Call the resulting modified algorithm $\mathcal{A}_4$. To see that algorithm $\mathcal{A}_4$ preserves differential privacy it suffices to give a bound on Equation (2). By comparing the case where $S = R$ with $S = R \cup \{i\}$, we get that the probability for a hypothesis h can increase only if $c = \mathbf{0}$ when $S = R$, and $c = c_{y_i}$ when $S = R \cup \{i\}$. Therefore,

$$\max_{R\subseteq[m']\setminus\{i\}} \frac{\Pr[\mathcal{A}_4(D) = h \mid S = R \cup \{i\}]}{\Pr[\mathcal{A}_4(D) = h \mid S = R]} \leq \frac{1}{(\alpha\beta/4) - \mathrm{negl}(d)} \leq \frac{1}{(\alpha\beta/8)} = \frac{8}{\alpha\beta}.$$

Theorem 4. *There exists an efficient improper private PAC learner for $POINT_d$ that uses $O_{\alpha,\beta,\epsilon}(1)$ labeled examples, where $\epsilon, \alpha,$ and β are parameters of the private learner.*

3.3 Separation Between Efficient and Inefficient Proper Private PAC Learning

In this section, we use the sample size lower bound for proper private learning $POINT_d$ to obtain a separation between efficient and inefficient proper private PAC learning. Let U_r represent a uniformly random string from $\{0,1\}^r$. Let $\ell(d) : \mathbb{N} \to \mathbb{N}$ be a function and $G = \{G_d\}_{d\in\mathbb{N}}$ be a deterministic algorithm such that on input from $\{0,1\}^{\ell(d)}$ it returns an output from $\{0,1\}^d$. Informally, we say that G is pseudorandom generator if on $\ell(d)$ truly random bits it outputs d bits that are indistinguishable from d random bits. Formally, for every probabilistic polynomial time algorithm $\mathcal{B}$ there exists a negligible function negl(d) (i.e., a function that is asymptotically smaller than $1/d^c$ for all $c > 0$) such that

$$|\Pr[\mathcal{B}(G_d(U_{\ell(d)})) = 1] - \Pr[\mathcal{B}(U_d) = 1]| \leq \mathrm{negl}(d).$$

Such exponential stretch pseudorandom generators G (i.e., with $\ell(d) = \omega(\log d)$) exist under various strong hardness assumptions.

Let $POINT_d = \{c_1, \ldots, c_{2^d}\}$. Now to a polynomially bounded private learner, $c_{G_d(U_{\ell(d)})}$ would appear with high probability as a uniformly random concept picked from $POINT_d$. We will show by using ideas similar to the proof of Theorem 3 that a polynomially bounded proper private learner would require $\Omega((d + \log(1/\beta))/\epsilon)$ labeled examples to learn $c_{G_d(U_{\ell(d)})}$. More precisely, define concept class

$$\widehat{POINT}_d = \bigcup_{r\in\{0,1\}^{\ell(d)}} \{c_{G_d(r)}\}.$$

Assume that there is an efficient proper private learner $\mathcal{A}$ for $\widehat{POINT}_d$ with sample size $m = o((d + \log(1/\beta))/\epsilon)$. We use $\mathcal{A}$ to construct a distinguisher for the pseudorandom generator: Given j we construct the database D with m entries $(j, 1)$. If $\mathcal{A}(D) = c_j$, then the distinguisher returns 1, otherwise it returns 0. If $j = G_d(r)$ for some r, then, by the utility of the private learner, $\mathcal{A}$ has to return c_j on this database with probability at least $1 - \beta$. Thus, the distinguisher returns 1 with probability at least $1 - \beta$ when j is chosen from $G_d(U_{\ell(d)})$. Assume that for (say) 1/4 of the values $j \in [2^d]$ algorithm $\mathcal{A}$, when applied to the database with m entries $(j, 1)$, returns c_j with probability at least 1/3. Then, we get a contradiction following the same argument as in the proof of Theorem 3 (as at least a fraction of 1/4 of the c_j's must have probability at least $(1/3)(1-\beta)\cdot\exp(-\epsilon m)$). Thus, the distinguisher returns 1 with probability at most $1/4 + 3/4 \cdot 1/3 = 1/2$ when j is chosen from U_d.

If the learner is not polynomially bounded, then it can use the algorithm from Theorem 2 to privately learn $\widehat{POINT}_d$. Since, $|\widehat{POINT}_d| = 2^{\ell(d)}$, the private learner from Theorem 2 uses $O((\ell(d)+\log(1/\beta))/(\epsilon\alpha))$ labeled examples. We get the following separation between efficient and inefficient proper private learning:

Theorem 5. *Let $\ell(d)$ be any function that grows as $\omega(\log d)$, and G be a be a pseudorandom generator with stretch $d - \ell(d)$. For the concept class $\widehat{POINT}_d$, every polynomial-time proper private PAC learner with probability at least $1 - \mathrm{negl}(d)$ requires $\Omega((d + \log(1/\beta))/\epsilon)$ labeled examples, whereas there exists an inefficient proper private PAC learner that can learn $\widehat{POINT}_d$ using $O((\ell(d) + \log(1/\beta))/(\epsilon\alpha))$ labeled examples.*

Remark 3. In the non-private setting, there exists an efficient proper learner that can learn the concept class $\widehat{POINT}_d$ using $O((\log(1/\alpha) + \log(1/\beta))/\alpha)$ labeled examples (as $VCDIM(\widehat{POINT}_d) = 1$). In the non-private setting we also know that even inefficient learners require $\Omega(\log(1/\beta)/\alpha)$ labeled examples [9, 13]. Therefore, for $\widehat{POINT}_d$ the sample complexities of efficient non-private learners and inefficient non-private learners are almost the same.

4 Lower Bounds for Non-Interactive Sanitization

We now prove a lower bound on the database size (or sample size) needed to privately release an output that is useful for all concepts in a concept class. We start by recalling a definition and a result of Blum *et al.* [3].

Let $X = \{X_d\}_{d\in\mathbb{N}}$ be some discretized domain and consider a class of predicates $\mathcal{C}$ over X. A database D contains points taken from X_d. A predicate query Q_c for $c : X_d \to \{0, 1\}$ in $\mathcal{C}$ is defined as

$$Q_c(D) = \frac{|\{d_i \in D : c(d_i) = 1\}|}{|D|}.$$

A sanitizer (or data release mechanism) is a differentially private algorithm $\mathcal{A}$ that on input a database D outputs another database $\widehat{D}$ with entries taken from

X_d. An algorithm $\mathcal{A}$ is (α,β)-useful for predicates in class $\mathcal{C}$ if with probability at least $1-\beta$ for every $c \in C$, and every database D, for $\widehat{D} = \mathcal{A}(D)$,

$$|Q_c(D) - Q_c(\widehat{D})| < \alpha.$$

Theorem 6 (Blum *et al.* [3]). *For any class of predicates $\mathcal{C}$, and any database $D \in X_d^m$, such that*

$$m \geq O\left(\frac{\log|X_d| \cdot \mathit{VCDIM}(\mathcal{C}) \log(1/\alpha)}{\alpha^3 \epsilon} + \frac{\log(1/\beta)}{\epsilon\alpha}\right),$$

there exists an (α,β)-useful mechanism $\mathcal{A}$ that preserves ϵ-differential privacy. The algorithm might not be efficient.

We show that the dependency on $\log|X_d|$ in Theorem 6 is essential: there exists a class of predicates $\mathcal{C}$ with VC-dimension $O(1)$ that requires $|D| = \Omega_{\alpha,\beta,\epsilon}(\log|X_d|)$. For our lower bound, the sanitized output $\widehat{D}$ could be any arbitrary data structure (not necessarily a synthetic database). For simplicity, however, here we focus on the case where the output is a synthetic database. The proof of this lower bound uses ideas from Section 3.1.

Let $T = 2^d$ and $X_d = [T]$ be the domain. Consider the class POINT_d (where $i \in [T]$). For every $i \in [T]$, construct a database $D_i \in X_d^m$ by setting $(1-3\alpha)m$ entries at 1 and the remaining $3\alpha m$ entries at i (for $i=1$ all entries of D_1 are 1). For $i \in [T] \setminus \{1\}$ we say that a database $\widehat{D}$ is α-useful for D_i if $2\alpha < Q_{c_i}(\widehat{D}) < 4\alpha$ and $1-4\alpha < Q_{c_1}(\widehat{D}) < 1-2\alpha$. We say that $\widehat{D}$ is α-useful for D_1 if $1-\alpha < Q_{c_1}(\widehat{D}) \leq 1$. It follows that for $i \neq j$ if $\widehat{D}$ is α-useful for D_i then it is not α-useful for D_j.

Let $\widehat{\mathbb{D}}_i$ be the set of all databases that are α-useful for D_i. Note that for all $i \neq 1$, D_1 and D_i differ on $3\alpha m$ entries, and by our previous observation, $\widehat{\mathbb{D}}_1 \cap \widehat{\mathbb{D}}_i = \emptyset$. Let $\mathcal{A}$ be an (α,β)-useful private release mechanism for POINT_d. For all i, on input D_i mechanism $\mathcal{A}$ should pick an output from $\widehat{\mathbb{D}}_i$ with probability at least $1-\beta$. We get by the differential privacy of $\mathcal{A}$ that

$$\Pr[\mathcal{A}(D_1) \in \widehat{\mathbb{D}}_i] \geq \exp(-3\epsilon\alpha m) \Pr[\mathcal{A}(D_i) \in \widehat{\mathbb{D}}_i] \geq \exp(-3\epsilon\alpha m) \cdot (1-\beta).$$

Hence,

$$\begin{aligned}
\Pr[\mathcal{A}(D_1) \notin \widehat{\mathbb{D}}_1] &\geq \Pr[\mathcal{A}(D_1) \in \bigcup_{i \neq 1} \widehat{\mathbb{D}}_i] \\
&= \sum_{i \neq 1} \Pr[\mathcal{A}(D_1) \in \widehat{\mathbb{D}}_i] \qquad \text{(sets } \widehat{\mathbb{D}}_i \text{ are disjoint)} \\
&\geq (T-1)\exp(-3\epsilon\alpha m) \cdot (1-\beta).
\end{aligned}$$

On the other hand, since $\mathcal{A}$ is (α,β)-useful, $\Pr[\mathcal{A}(D_1) \notin \widehat{\mathbb{D}}_1] < \beta$, and hence we get that $m = \Omega((d + \log(1/\beta))/(\epsilon\alpha))$.

Theorem 7. *Every ϵ-differentially private non-interactive mechanism that is (α,β)-useful for POINT_d requires an input database of $\Omega((d+\log(1/\beta))/(\epsilon\alpha))$ size.*

Acknowledgments

We thank Benny Applebaum, Eyal Kushilevitz, and Adam Smith for helpful initial discussions.

References

[1] Amos Beimel, Shiva Kasiviswanathan, and Kobbi Nissim. Bounds on the Sample Complexity for Private Learning and Private Data Release. Full version, 2009.
[2] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: The SuLQ framework. In *PODS*, pages 128–138. ACM, 2005.
[3] Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to non-interactive database privacy. In *STOC*, pages 609–618. ACM, 2008.
[4] Avrim Blum, Katrina Ligett, and Aaron Roth. Private communication, 2008.
[5] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the Association for Computing Machinery*, 36(4):929–965, 1989.
[6] Cynthia Dwork. The differential privacy frontier (extended abstract). In *TCC*, volume 5444 of *LNCS*, pages 496–502. Springer, 2009.
[7] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, volume 3876 of *LNCS*, pages 265–284. Springer, 2006.
[8] Cynthia Dwork, Moni Naor, Omer Reingold, Guy Rothblum, and Salil Vadhan. On the complexity of differentially private data release. In *STOC*, pages 381–390. ACM, 2009.
[9] 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.
[10] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? In *FOCS*, pages 531–540. IEEE Computer Society, 2008.
[11] Shiva Prasad Kasiviswanathan and Adam Smith. A note on differential privacy: Defining resistance to arbitrary side information. *CoRR*, arXiv:0803.39461 [cs.CR], 2008.
[12] Michael J. Kearns. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM*, 45(6):983–1006, 1998. Preliminary version in *proceedings of STOC'93*.
[13] Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT press, Cambridge, Massachusetts, 1994.
[14] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *FOCS*, pages 94–103. IEEE, 2007.
[15] Nina Mishra and Mark Sandler. Privacy via pseudorandom sketches. In *PODS*, pages 143–152. ACM, 2006.
[16] Leonard Pitt and Leslie G. Valiant. Computational limitations on learning from examples. *Journal of the ACM*, 35(4):965–984, 1988.
[17] Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.
[18] 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*, 16:264, 1971.