What Can We Learn Privately?*

Shiva Prasad Kasiviswanathan† Homin K. Lee‡ Kobbi Nissim§
Sofya Raskhodnikova¶ Adam Smith¶

February 13, 2013

Abstract

Learning problems form an important category of computational tasks that generalizes many of the computations researchers apply to large real-life data sets. We ask: what concept classes can be learned privately, namely, by an algorithm whose output does not depend too heavily on any one input or specific training example? More precisely, we investigate learning algorithms that satisfy *differential privacy*, a notion that provides strong confidentiality guarantees in contexts where aggregate information is released about a database containing sensitive information about individuals.

Our goal is a broad understanding of the resources required for private learning in terms of samples, computation time, and interaction. We demonstrate that, ignoring computational constraints, it is possible to privately agnostically learn any concept class using a sample size approximately logarithmic in the cardinality of the concept class. Therefore, almost anything learnable is learnable privately: specifically, if a concept class is learnable by a (non-private) algorithm with polynomial sample complexity and output size, then it can be learned privately using a polynomial number of samples. We also present a computationally efficient private PAC learner for the class of *parity* functions. This result dispels the similarity between learning with noise and private learning (both must be robust to small changes in inputs), since parity is thought to be very hard to learn given random classification noise.

Local (or *randomized response*) algorithms are a practical class of private algorithms that have received extensive investigation. We provide a precise characterization of local private learning algorithms. We show that a concept class is learnable by a local algorithm if and only if it is learnable in the *statistical query* (SQ) model. Therefore, for local private learning algorithms, the similarity to learning with noise is stronger: local learning is equivalent to SQ learning, and SQ algorithms include most known noise-tolerant learning algorithms. Finally, we present a separation between the power of *interactive* and *noninteractive* local learning algorithms. Because of the equivalence to SQ learning, this result also separates *adaptive* and *nonadaptive* SQ learning.

1 Introduction

The data privacy problem in modern databases is similar to that faced by statistical agencies and medical researchers: to learn and publish global analyses of a population while maintaining the confidentiality of the

*A preliminary version of this paper appeared in 49th Annual IEEE Symposium on Foundations of Computer Science [37].

†CCS-3, Los Alamos National Laboratory. Part of this work done while a student at Pennsylvania State University and supported by NSF award CCF-072891.

‡Department of Computer Science, Columbia University, hkl7@columbia.edu

§Department of Computer Science, Ben-Gurion University, kobbi@cs.bgu.ac.il. Supported in part by the Israel Science Foundation (grant 860/06), and by the Frankel Center for Computer Science.

¶Department of Computer Science and Engineering, Pennsylvania State University, {sofya,asmith}@cse.psu.edu. S.R. and A.S. are supported in part by NSF award CCF-0729171.

participants in a survey. There is a vast body of work on this problem in statistics and computer science. However, until recently, most schemes proposed in the literature lacked rigorous analysis of privacy and utility.

A recent line of work [29, 26, 11, 24, 22, 21, 47, 25, 44, 7, 48, 14, 27] , initiated by Dinur and Nissim [20] and called *private data analysis*, seeks to place data privacy on firmer theoretical foundations and has been successful at formulating a strong, yet attainable privacy definition. The notion of *differential privacy* [24] that emerged from this line of work provides rigorous guarantees even in the presence of a malicious adversary with access to arbitrary auxiliary information. It requires that whether an individual supplies her actual or fake information has almost no effect on the outcome of the analysis.

Given this definition, it is natural to ask: what computational tasks can be performed while maintaining privacy? Research on data privacy, to the extent that it formalizes precise goals, has mostly focused on *function evaluation* ("what is the value of $f(\mathrm{z})$?"), namely, how much privacy is possible if one wishes to release (an approximation to) a particular function f, evaluated on the database z. (A notable exception is the recent work of McSherry and Talwar, using differential privacy in the design of auction mechanisms [44]). Our goal is to expand the utility of private protocols by examining which other computational tasks can be performed in a privacy-preserving manner.

Private Learning. Learning problems form an important category of computational tasks that generalizes many of the computations researchers apply to large real-life data sets. In this work, we ask what can be learned *privately*, namely, by an algorithm whose output does not depend too heavily on any one input or specific training example. Our goal is a broad understanding of the resources required for private learning in terms of samples, computation time, and interaction. We examine two basic notions from computational learning theory: Valiant's probabilistically approximately correct (PAC) learning [51] model and Kearns' statistical query (SQ) model [39].

Informally, a concept is a function from examples to labels, and a class of concepts is learnable if for any distribution $\mathcal{D}$ on examples, one can, given limited access to examples sampled from $\mathcal{D}$ labeled according to some target concept c, find a small circuit (hypothesis) which predicts c's labels with high probability over future examples taken from the same distribution. In the PAC model, a learning algorithm can access a polynomial number of labeled examples. In the SQ model, instead of accessing examples directly, the learner can specify some properties (i.e., predicates) on the examples, for which he is given an estimate, up to an additive polynomially small error, of the probability that a random example chosen from $\mathcal{D}$ satisfies the property. PAC learning is strictly stronger than the SQ learning [39].

We model a statistical database as a vector $\mathrm{z} = (z_1, \cdots, z_n)$, where each entry has been contributed by an individual. When analyzing how well a private algorithm learns a concept class, we assume that entries z_i of the database are random examples generated i.i.d. from the underlying distribution $\mathcal{D}$ and labeled by a target concept c. This is exactly how (not necessarily private) learners are analyzed. For instance, an example might consist of an individual's gender, age, and blood pressure history, and the label, whether this individual has had a heart attack. The algorithm has to learn to predict whether an individual has had a heart attack, based on gender, age, and blood pressure history, generated according to $\mathcal{D}$.

We require a private algorithm to keep entire examples (not only the labels) confidential. In the scenario above, it translates to not revealing each participant's gender, age, blood pressure history, and heart attack incidence. More precisely, the output of a private learner should not be significantly affected if a particular example z_i is replaced with arbitrary z_i', for all z_i and z_i'. In contrast to correctness or utility, which is analyzed with respect to distribution $\mathcal{D}$, differential privacy is a worst-case notion. Hence, when we analyze the privacy of our learners we do not make any assumptions on the underlying distribution. Such as-

sumptions are fragile and, in particular, would fall apart in the presence of auxiliary knowledge (also called background knowledge or side information) that the adversary might have: conditioned on the adversary's auxiliary knowledge, the distribution over examples might look very different from $\mathcal{D}$.

1.1 Our Contributions

We introduce and formulate private learning problems, as discussed above, and develop novel algorithmic tools and bounds on the sample size required by private learning algorithms. Our results paint a picture of the classes of learning problems that are solvable subject to privacy constraints. Specifically, we provide:

(1) A Private Version of Occam's Razor. We present a generic private learning algorithm. For any concept class $\mathcal{C}$, we give a distribution-free differentially-private agnostic PAC learner for $\mathcal{C}$ that uses a number of samples proportional to $\log |\mathcal{C}|$. This is a private analogue of the "cardinality version" of *Occam's razor*, a basic sample complexity bound from (non-private) learning theory. The sample complexity of our version is similar to that of the original, although the private algorithm is very different. As in Occam's razor, the learning algorithm is not necessarily computationally efficient.

(2) An Efficient Private Learner for Parity. We give a computationally efficient, distribution-free differentially private PAC learner for the class of *parity* functions[1] over $\{0,1\}^d$. The sample and time complexity are comparable to that of the best non-private learner.

(3) Equivalence of Local ("Randomized Response") and SQ Learning. We precisely characterize the power of *local*, or *randomized response*, private learning algorithms. Local algorithms are a special (practical) class of private algorithms and are popular in the data mining and statistics literature [53, 2, 1, 3, 52, 29, 45, 36]. They add randomness to each individual's data independently before processing the input. We show that a concept class is learnable by a local differentially private algorithm if and only if it is learnable in the *statistical query* (SQ) model. This equivalence relates notions that were conceived in very different contexts.

(4) Separation of Interactive and Noninteractive Local Learning. Local algorithms can be *noninteractive*, that is, using one round of interaction with individuals holding the data, or *interactive*, that is, using more than one round (and in each receiving randomized responses from individuals). We construct a concept class, called *masked-parity*, that is efficiently learnable by *interactive* local algorithms under the uniform distribution on examples, but requires an exponential (in the dimension) number of samples to be learned by a *noninteractive* local algorithm. The equivalence **(3)** of local and SQ learning shows that interaction in local algorithms corresponds to *adaptivity* in SQ algorithms. The *masked-parity* class thus also separates adaptive and nonadaptive SQ learning.

1.1.1 Implications

"Anything" learnable is privately learnable using few samples. The generic agnostic learner **(1)** has an important consequence: if some concept class $\mathcal{C}$ is learnable by any algorithm, not necessarily a private one, whose output length in bits is polynomially bounded, then $\mathcal{C}$ is learnable privately using a polynomial number of samples (possibly in exponential time). This result establishes the basic feasibility of private learning: it was not clear *a priori* how severely privacy affects sample complexity, even ignoring computation time.

[1] While the generic learning result **(1)** extends easily to "agnostic" learning (defined below), the learner for parity does not. The limitation is not surprising, since even non-private agnostic learning of parity is at least as hard as learning parity with random noise.

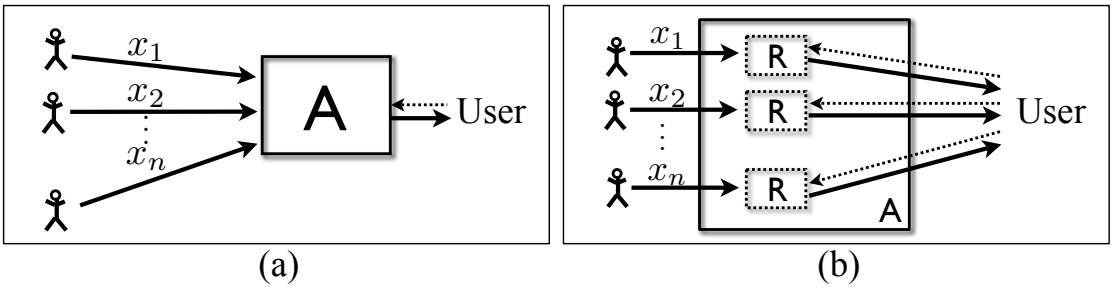

Figure 1: Two basic models for database privacy: (a) the *centralized* model, in which data is collected by a trusted agency that publishes aggregate statistics or answers users' queries; (b) the *local* model, in which users retain their data and run a randomization procedure locally to produce output which is safe for publication. The dotted arrows from users to data holders indicate that protocols may be completely noninteractive: in this case there is a single publication, without feedback from users.

Learning with noise is different from private learning. There is an intuitively appealing similarity between learning from noisy examples and private learning: algorithms for both problems must be robust to small variations in the data. This apparent similarity is strengthened by a result of Blum, Dwork, McSherry and Nissim [11] showing that any algorithm in Kearns' *statistical query* (SQ) model [39] can be implemented in a differentially private manner. SQ was introduced to capture a class of noise-resistant learning algorithms. These algorithms access their input only through a sequence of approximate averaging queries. One can *privately* approximate the average of a function with values in $[0, 1]$ over the data set of n individuals to within additive error $O(1/n)$ (Dwork and Nissim [26]). Thus, one can simulate the behavior of an SQ algorithm privately, query by query.

Our efficient private learner for parity **(2)** dispels the similarity between learning with noise and private learning. First, SQ algorithms provably require exponentially many (in the dimension) queries to learn parity [39]. More compellingly, learning parity with noise is thought to be computationally hard, and has been used as the basis of several cryptographic primitives (*e.g.,* [13, 35, 4, 49]).

Limitations of local ("randomized response") algorithms. *Local* algorithms (also referred to as *randomized response*, *input perturbation*, *Post Randomization Method (PRAM)*, and *Framework for High-Accuracy Strict-Privacy Preserving Mining (FRAPP)*) have been studied extensively in the context of privacy-preserving data mining, both in statistics and computer science (*e.g.,* [53, 2, 1, 3, 52, 29, 45, 36]). Roughly, a local algorithm accesses each individual's data via independent randomization operators. See Figure 1, p. 4.

Local algorithms were introduced to encourage truthfulness in surveys: respondents who know that their data will be randomized are more likely to answer honestly. For example, Warner [53] famously considered a survey technique in which respondents are asked to give the correct answer to a sensitive (true/false) question with probability $2/3$ and the incorrect answer with probability $1/3$, in the hopes that the added uncertainty would encourage them to answer honestly. The proportion of "true" answers in the population is then estimated using a standard, non-private deconvolution. The accepted privacy requirement for local algorithms is equivalent to imposing differential privacy on each randomization operator [29]. Local algorithms are popular because they are easy to understand and implement. In the extreme case, users can retain their data and apply the randomization operator themselves, using a physical device [53, 46] or a cryptographic protocol [5].

The equivalence between local and SQ algorithms **(3)** is a powerful tool that allows us to apply results

from learning theory. In particular, since parity is not learnable with a small number of SQ queries [39] but is PAC learnable privately **(2)**, we get that local algorithms require exponentially more data for some learning tasks than do general private algorithms. Our results also imply that local algorithms are strictly *less* powerful than (non-private) algorithms for learning with classification noise because subexponential (non-private) algorithms can learn parity with noise [13].

Adaptivity in SQ algorithms is important. Just as local algorithms can be interactive, SQ algorithms can be *adaptive*, that is, the averaging queries they make may depend on answers to previous queries. The equivalence of SQ and local algorithms **(3)** preserves interaction/adaptivity: a concept class is nonadaptively SQ learnable if and only if it is noninteractively locally learnable. The masked parity class **(4)** shows that interaction (resp., adaptivity) adds considerable power to local (resp., SQ) algorithms.

Most of the reasons that local algorithms are so attractive in practice, and have received such attention, apply only to noninteractive algorithms (interaction can be costly, complicated, or even impossible—for instance, when statistical information is collected by an interviewer, or at a polling booth).

This suggests that further investigating the power of nonadaptive SQ learners is an important problem. For example, the SQ algorithm for learning conjunctions [42] is nonadaptive, but SQ formulations of the perceptron and k-means algorithms [11] seem to rely heavily on adaptivity.

Understanding the "price" of privacy for learning problems. The SQ result of Blum *et al.* [11] and our learner for parity **(2)** provide efficient (i.e., polynomial time) private learners for essentially all the concept classes known (by us) to have efficient non-private distribution-free learners. Finding a concept class that can be learned efficiently, but not privately and efficiently, remains an interesting and important question.

Our results also lead to questions of optimal sample complexity for learning problems of practical importance. The private simulation of SQ algorithms due to Blum *et al.* [11] uses a factor of approximately $\sqrt{t}/\epsilon$ more data points than the naïve non-private implementation, where t is the number of SQ queries and ϵ is the parameter of differential privacy (typically a small constant). In contrast, the generic agnostic learner **(1)** uses a factor of at most $1/\epsilon$ more samples than the corresponding non-private learner. For parity, our private learner uses a factor of roughly $1/\epsilon$ more samples than, and about the same computation time as, the non-private learner. What, then, is the additional cost of privacy when learning practical concept classes (half-planes, low-dimensional curves, etc)? Can the theoretical sample bounds of **(1)** be matched by (more) efficient learners?

1.1.2 Techniques

Our generic private learner **(1)** adapts the exponential sampling technique of McSherry and Talwar [44], developed in the context of auction design. Our use of the exponential mechanism inspired an elegant *subsequent* result of Blum, Liggett, and Roth [14] (BLR) on simultaneously approximating many different functions.

The efficient private learner for parity **(2)** uses a very different technique, based on sampling, running a non-private learner, and occasionally refusing to answer based on delicately calibrated probabilities. Running a non-private learner on a random subset of examples is a very intuitive approach to building private algorithms, but it is not private in general. The private learner for parity illustrates both why this technique can leak private information and how it can sometimes be repaired based on special (in this case, algebraic) structure.

The interesting direction of the equivalence between SQ and local learners **(3)** is proved via a simulation of any local algorithm by a corresponding SQ algorithm. We found this simulation surprising since local protocols can, in general, have very complex structure (see, *e.g.,* [29]). The SQ algorithm proceeds by a direct simulation of the output of the randomization operators. For a given input distribution $\mathcal{D}$ and any operator R, one can sample from the corresponding output distribution $R(\mathcal{D})$ via rejection sampling. We show that if R is differentially private, the rejection probabilities can be approximated via low-accuracy SQ queries to $\mathcal{D}$.

Finally, the separation between adaptive and nonadaptive SQ **(4)** uses a Fourier analytic argument inspired by Kearns' SQ lower bound for parity [39].

1.1.3 Classes of Private Learning Algorithms

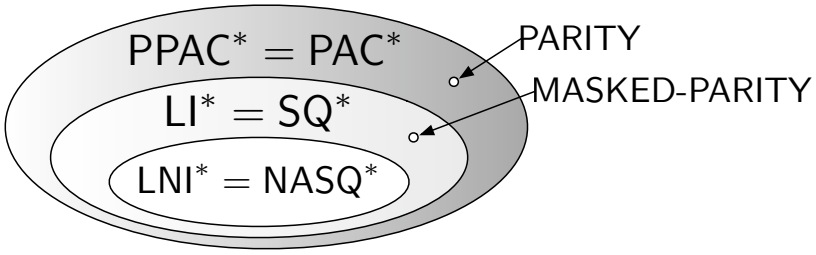

Figure 2: Relationships among learning classes taking into account sample complexity, but not computational efficiency.

We can summarize our results via a complexity-theoretic picture of learnable and privately learnable concept classes (more precisely, the members of the classes are pairs of concept classes and example distributions). In order to make asymptotic statements, we measure complexity in terms of the length d of the binary description of examples.

We first consider learners that use a polynomial (in d) number of samples and output a hypothesis that is described using a polynomial number of bits, but have unlimited computation time. Let PAC^* denote the set of concept classes that are learnable by such algorithms ignoring privacy, and let PPAC^* denote the subset of PAC^* learnable by differentially private[2] algorithms.

Since we restrict the learner's output to a polynomial number of bits, the hypothesis classes of the algorithms are *de facto* limited to have size at most $\exp(poly(d))$. Thus, the generic private learner (point **(1)** in the introduction) will use a polynomial number of samples, and $\mathsf{PAC}^* = \mathsf{PPAC}^*$.

We can similarly interpret the other results above. Within PAC^*, we can consider subsets of concepts learnable by SQ algorithms (SQ^*), nonadaptive SQ algorithms (NASQ^*), local interactive algorithms (LI^*) and local noninteractive algorithms (LNI^*). We obtain the following picture (see page 6):

$$\mathsf{LNI}^* = \mathsf{NASQ}^* \quad \subsetneq \quad \mathsf{LI}^* = \mathsf{SQ}^* \quad \subsetneq \quad \mathsf{PPAC}^* = \mathsf{PAC}^*.$$

The equality of LI^* and SQ^*, and of LNI^* and NASQ^*, follow from the SQ simulation of local algorithms (Theorem 5.14). The parity and masked-parity concept classes separate PPAC^* from SQ^* and SQ^* from NASQ^*, respectively (Corollaries 5.15 and 5.17). (*Note:* The separation of PPAC^* from SQ^* holds even for distribution-free learning; in contrast, the separation of SQ^* from NASQ^* holds for learnability under a

[2]Differential privacy is quantified by a real parameter $\epsilon > 0$. To make qualitative statements, we look at algorithms where $\epsilon \to 0$ as $d \to \infty$. Taking $\epsilon = 1/d^c$ for any constant $c > 0$ would yield the same class.

specific distribution on examples, since the adaptive SQ learner for MASKED-PARITY requires a uniform distribution on examples.)

When we take computational efficiency into account, the picture changes. The relation between local and SQ classes remain the same modulo a technical restriction on the randomization operators (Definition 5.13). SQ remains distinct from PPAC since parity is efficiently learnable privately. However, it is an open question whether concept classes which can be efficiently learned can also be efficiently learned privately.

1.2 Related Work

Prior to this work, the literature on differential privacy studied function approximation tasks (e.g. [20, 26, 11, 24, 47, 7]), with the exception of the work of McSherry and Talwar on mechanism design [44]. Nevertheless, several of these prior results have direct implications to machine learning-related problems. Blum *et al.* [11] considered a particular class of learning algorithms (SQ), and showed that algorithms in the class could be simulated using noisy function evaluations. In an independent, unpublished work, Chaudhuri, Dwork, and Talwar considered a version of private learning in which privacy is afforded only to input labels, but not to examples. Other works considered specific machine learning problems such as mining frequent itemsets [29], k-means clustering [11, 47], learning decision trees [11], and learning mixtures of Gaussians [47].

As mentioned above, a subsequent result of Blum, Ligett and Roth [14] on approximating classes of low-VC-dimension functions was inspired by our generic agnostic learner. We discuss their result further in Section 3.1. Since the original version of our work, there have also been several results connecting differential privacy to more "statistical" notions of utility, such as consistency of point estimation and density estimation [50, 23, 54, 56].

Our separation of interactive and noninteractive protocols in the *local* model **(3)** also has a precedent: Dwork *et al.* [24] separated interactive and noninteractive private protocols in the *centralized* model, where the user accesses the data via a server that runs differentially private algorithms on the database and sends back the answers. That separation has a very different flavor from the one in this work: any example of a computation that cannot be performed noninteractively in the centralized model must rely on the fact that the computational task is not defined until after the first answer from the server is received. (Otherwise, the user can send an algorithm for that task to the server holding the data, thus obviating the need for interaction.) In contrast, we present a computational task that is hard for noninteractive local algorithms – learning *masked parity* – yet is defined in advance.

In the machine learning literature, several notions similar to differential privacy have been explored under the rubric of "algorithmic stability" [19, 40, 16, 43, 28, 9]. The most closely related notion is *change-one error stability*, which measures how much the generalization error changes when an input is changed (see the survey [43]). In contrast, differential privacy measures how the distribution over the entire output changes—a more complex measure of stability (in particular, differential privacy implies change-one error stability). A different notion, stability under resampling of the data from a given distribution [10, 9], is connected to the sample-and-aggregate method of [47] but is not directly relevant to the techniques considered here. Finally, in a different vein, Freund, Mansour and Schapire [31] used a weighted averaging technique with the same weights as the sampler in our generic learner to reduce generalization error (see Section 3.1).

2 Preliminaries

We use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. Logarithms base 2 and base e are denoted by $\log$ and $\ln$, respectively. $\Pr[\cdot]$ and $\mathbb{E}[\cdot]$ denote probability and expectation, respectively. $\mathcal{A}(x)$ is the probability distribution over outputs of a randomized algorithm $\mathcal{A}$ on input x. The *statistical difference* between distributions $\mathbb{P}$ and $\mathbb{Q}$ on a discrete space D is defined as $\max_{S \subset D} |\mathbb{P}(S) - \mathbb{Q}(S)|$.

2.1 Differential Privacy

A statistical database is a vector $\mathrm{z} = (z_1, \ldots, z_n)$ over a domain D, where each entry $z_i \in D$ represents information contributed by one individual. Databases z and z′ are *neighbors* if $z_i \neq z'_i$ for exactly one $i \in [n]$ (i.e., the Hamming distance between z and z′ is 1). All our algorithms are symmetric, that is, they do not depend on the order of entries in the database z. Thus, we could define a database as a multi-set in D, and use symmetric difference instead of the Hamming metric to measure distance. We adhere to the vector formulation for consistency with the previous works.

A (randomized) algorithm (in our context, this will usually be a learning algorithm) is private if neighboring databases induce nearby distributions on its outcomes:

Definition 2.1 (ϵ-differential privacy [24])**.** *A randomized algorithm $\mathcal{A}$ is ϵ-differentially private if for all neighboring databases z, z', and for all sets $\mathcal{S}$ of outputs,*

$$\Pr[\mathcal{A}(\mathrm{z}) \in \mathcal{S}] \leq \exp(\epsilon) \cdot \Pr[\mathcal{A}(\mathrm{z}') \in \mathcal{S}].$$

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

In [24], the notion above was called "indistinguishability". The name "differential privacy" was suggested by Mike Schroeder, and first appeared in Dwork [21].

Differential privacy composes well (see, *e.g.,* [22, 47, 44, 38]):

Claim 2.2 (Composition and Post-processing)**.** *If a randomized algorithm $\mathcal{A}$ runs k algorithms $\mathcal{A}_1, ..., \mathcal{A}_k$, where each $\mathcal{A}_i$ is ϵ_i-differentially private, and outputs a function of the results (that is, $\mathcal{A}(\mathrm{z}) = g(\mathcal{A}_1(\mathrm{z}), \mathcal{A}_2(\mathrm{z}), ..., \mathcal{A}_k(\mathrm{z}))$ for some probabilistic algorithm g), then $\mathcal{A}$ is $(\sum_{i=1}^{k} \epsilon_i)$-differentially private.*

One method for obtaining efficient differentially private algorithms for approximating real-valued functions is based on adding Laplacian noise to the true answer. Let $\mathrm{Lap}(\lambda)$ denote the Laplace probability distribution with mean 0, standard deviation $\sqrt{2}\lambda$, and p.d.f. $f(x) = \frac{1}{2\lambda} e^{-|x|/\lambda}$.

Theorem 2.3 (Dwork *et al.* [24])**.** *For a function $f : D^n \to \mathbb{R}$, define its global sensitivity $GS_f = \max_{\mathrm{z},\mathrm{z}'} |f(\mathrm{z}) - f(\mathrm{z}')|$ where the maximum is over all neighboring databases z, z'. Then, an algorithm that on input z returns $f(\mathrm{z}) + \eta$ where $\eta \sim \mathrm{Lap}(GS_f/\epsilon)$ is ϵ-differentially private.*

2.2 Preliminaries from Learning Theory

A concept is a function that labels *examples* taken from the domain X by the elements of the range Y. A *concept class* $\mathcal{C}$ is a set of concepts. It comes implicitly with a way to represent concepts; $size(c)$ is the size of the (smallest) representation of c under the given representation scheme. The domain and the range of the concepts in $\mathcal{C}$ are understood to be ensembles $X = \{X_d\}_{d \in \mathbb{N}}$ and $Y = \{Y_d\}_{d \in \mathbb{N}}$, where the representation of elements in X_d, Y_d is of size at most d. We focus on binary classification problems, in which the label space Y_d is $\{0, 1\}$ or $\{+1, -1\}$; the parameter d thus measures the size of the examples

in X_d. (We use the parameter d to formulate asymptotic complexity notions.) The concept classes are ensembles $\mathcal{C} = \{\mathcal{C}_d\}_{d\in\mathbb{N}}$ where $\mathcal{C}_d$ is the class of concepts from X_d to Y_d. When the size parameter is clear from the context or not important, we omit the subscript in $X_d, Y_d, \mathcal{C}_d$.

Let $\mathcal{D}$ be a distribution over labeled examples in $X_d \times Y_d$. A learning algorithm is given access to $\mathcal{D}$ (the method for accessing $\mathcal{D}$ depends on the type of learning algorithm). It outputs a hypothesis $h : X_d \to Y_d$ from a hypothesis class $\mathcal{H} = \{\mathcal{H}_d\}_{d\in\mathbb{N}}$. The goal is to minimize the *misclassification error* of h on $\mathcal{D}$, defined as

$$\mathit{err}(h) = \Pr_{(x,y)\sim\mathcal{D}}[h(x) \neq y] .$$

The success of a learning algorithm is quantified by parameters α and β, where α is the desired error and β bounds the probability of failure to output a hypothesis with this error. Error measures other than misclassification are considered in supervised learning (*e.g.,* L_2^2). We study only misclassification error here, since for binary labels it is equivalent to the other common error measures.

A learning algorithm is usually given access to an oracle that produces i.i.d. samples from $\mathcal{D}$. Equivalently, one can view the learning algorithm's input as a list of n labeled examples, i.e., $\mathrm{z} \in D^n$ where $D = X_d \times Y_d$. PAC learning and agnostic learning are described in Definitions 2.4 and 2.5. Another common method of access to $\mathcal{D}$ is via "statistical queries", which return the approximate average of a function over the distribution. Algorithms that work in this model can be simulated given i.i.d. examples. See Section 5.

PAC learning algorithms are frequently designed assuming a promise that the examples are labeled consistently with some *target* concept c from a class $\mathcal{C}$: namely, $c \in \mathcal{C}_d$ and $y = c(x)$ for all (x, y) in the support of $\mathcal{D}$. In that case, we can think of $\mathcal{D}$ as a distribution only over examples X_d. To avoid ambiguity, we use $\mathcal{X}$ to denote a distribution over X_d. In the PAC setting, $\mathit{err}(h) = \Pr_{x\sim\mathcal{X}}[h(x) \neq c(x)]$.

Definition 2.4 (PAC Learning)**.** *A concept class $\mathcal{C}$ over X is* PAC learnable using hypothesis class $\mathcal{H}$ *if there exist an algorithm $\mathcal{A}$ and a polynomial $poly(\cdot,\cdot,\cdot)$ such that for all $d \in \mathbb{N}$, all concepts $c \in \mathcal{C}_d$, all distributions $\mathcal{X}$ on X_d, and all $\alpha, \beta \in (0, 1/2)$, given inputs α, β and $\mathrm{z} = (z_1, \cdots, z_n)$, where $n = poly(d, 1/\alpha, \log(1/\beta))$, $z_i = (x_i, c(x_i))$ and x_i are drawn i.i.d. from $\mathcal{X}$ for $i \in [n]$, algorithm $\mathcal{A}$ outputs a hypothesis $h \in \mathcal{H}$ satisfying*

$$\Pr[\mathit{err}(h) \leq \alpha] \geq 1 - \beta. \tag{1}$$

The probability is taken over the random choice of the examples z *and the coin tosses of $\mathcal{A}$.*

Class $\mathcal{C}$ is (inefficiently) PAC learnable *if there exists some hypothesis class $\mathcal{H}$ and a PAC learner $\mathcal{A}$ such that $\mathcal{A}$ PAC learns $\mathcal{C}$ using $\mathcal{H}$. Class $\mathcal{C}$ is* efficiently *PAC learnable if $\mathcal{A}$ runs it time polynomial in $d, 1/\alpha$, and* $\log(1/\beta)$.

Remark: Our definition deviates slightly from the standard one (see, *e.g.,* [42]) in that we do not take into consideration the size of the concept c. This choice allows us to treat PAC learners and agnostic learners identically. One can change Definition 2.4 so that the number of samples depends polynomially also on the size of c without affecting any of our results significantly.

Agnostic learning [32, 41] is an extension of PAC learning that removes assumptions about the target concept. Roughly speaking, the goal of an agnostic learner for a concept class $\mathcal{C}$ is to output a hypothesis $h \in \mathcal{H}$ whose error with respect to the distribution is close to the optimal possible by a function from $\mathcal{C}$. In the agnostic setting, $\mathit{err}(h) = \Pr_{(x,y)\sim\mathcal{D}}[h(x) \neq y]$.

Definition 2.5 (Agnostic Learning)**.** (Efficiently) agnostically learnable *is defined identically to (efficiently) PAC learnable with two exceptions:* (i) *the data are drawn from an arbitrary distribution $\mathcal{D}$ on $X_d \times Y_d$;* (ii) *instead of Equation 1, the output of $\mathcal{A}$ has to satisfy:*

$$\Pr[err(h) \leq OPT + \alpha] \geq 1 - \beta,$$

where $OPT = \min_{f \in \mathcal{C}_d} \{err(f)\}$. As before, the probability is taken over the random choice of z*, and the coin tosses of $\mathcal{A}$.*

Definitions 2.4 and 2.5 capture *distribution-free* learning, in that they do not assume a particular form for the distributions $\mathcal{X}$ or $\mathcal{D}$. In Section 5.3, we also consider learning algorithms that assume a *specific* distribution $\mathcal{D}$ on examples (but make no assumption on which concept in $\mathcal{C}$ labels the examples). When we discuss such algorithms, we specify $\mathcal{D}$ explicitly; without qualification, "learning" refers to distribution-free learning.

Efficiency Measures. The definitions above are sufficiently detailed to allow for exact complexity statements (*e.g.,* "$\mathcal{A}$ learns $\mathcal{C}$ using $n(\alpha, \beta)$ examples and time $O(t)$"), and the upper and lower bounds in this paper are all stated in this language. However, we also focus on two broader measures to allow for qualitative statements: *(a) polynomial sample complexity* is the default notion in our definitions. With the novel restriction of privacy, it is not *a priori* clear which concept classes can be learned using few examples even if we ignore computation time. *(b)* We use the term *efficient* private learning to impose the additional restriction of *polynomial computation time* (which implies polynomial sample complexity).

3 Private PAC and Agnostic Learning

We define private PAC learners as algorithms that satisfy definitions of both differential privacy and PAC learning. We emphasize that these are qualitatively different requirements. Learning must succeed on average over a set of examples drawn i.i.d. from $\mathcal{D}$ (often under the additional promise that $\mathcal{D}$ is consistent with a concept from a target class). Differential privacy, in contrast, must hold in the worst case, with no assumptions on consistency.

Definition 3.1 (Private PAC Learning)**.** *Let d, α, β be as in Definition 2.4 and $\epsilon > 0$. Concept class $\mathcal{C}$ is (inefficiently)* privately PAC learnable *using hypothesis class $\mathcal{H}$ if there exists an algorithm $\mathcal{A}$ that takes inputs $\epsilon, \alpha, \beta,$* z*, where n, the number of labeled examples in* z*, is polynomial in $1/\epsilon, d, 1/\alpha, \log(1/\beta)$, and satisfies*

a. [Privacy] *For all $\epsilon > 0$, algorithm $\mathcal{A}(\epsilon, \cdot, \cdot, \cdot)$ is ϵ-differentially private (Definition 2.1);*

b. [Utility] *Algorithm $\mathcal{A}$ PAC learns $\mathcal{C}$ using $\mathcal{H}$ (Definition 2.4).*

$\mathcal{C}$ is efficiently *privately PAC learnable if $\mathcal{A}$ runs in time polynomial in $d, 1/\epsilon, 1/\alpha,$ and $\log(1/\beta)$.*

Definition 3.2 (Private Agnostic Learning)**.** *(Efficient)* private agnostic learning *is defined analogously to (efficient) private PAC learning with Definition 2.5 replacing Definition 2.4 in the utility condition.*

Evaluating the quality of a particular hypothesis is easy: one can privately compute the fraction of the data it classifies correctly (enabling cross-validation) using the sum query framework of [11]. The difficulty of constructing private learners lies in finding a good hypothesis in what is typically an exponentially large space.

3.1 A Generic Private Agnostic Learner

In this section, we present a private analogue of a basic consistent learning result, often called the cardinality version of Occam's razor[3]. This classical result shows that a PAC learner can weed out all *bad* hypotheses given a number of labeled examples that is logarithmic in the size of the hypothesis class (see [42, p. 35]). Our generic private learner is based on the exponential mechanism of McSherry and Talwar [44].

Let $q : D^n \times \mathcal{H}_d \to \mathbb{R}$ take a database z and a candidate hypothesis h, and assign it a score $q(\mathrm{z}, h) = -|\{i \ : \ x_i \text{ is misclassified by } h, \textit{ i.e., } y_i \neq h(x_i)\}|$. That is, the score is minus the number of points in z misclassified by h. The classic Occam's razor argument assumes a learner that selects a hypothesis with maximum score (that is, minimum empirical error). Instead, our private learner $\mathcal{A}_q^\epsilon$ is defined to sample a random hypothesis with probability dependent on its score:

$$\mathcal{A}_q^\epsilon(\mathrm{z}) \quad : \quad \text{Output hypothesis } h \in \mathcal{H}_d \text{ with probability proportional to } \exp\left(\tfrac{\epsilon q(\mathrm{z},h)}{2}\right).$$

Since the score ranges from $-n$ to 0, hypotheses with low empirical error are exponentially more likely to be selected than ones with high error.

Algorithm $\mathcal{A}_q^\epsilon$ fits the framework of McSherry and Talwar, and so is ϵ-differentially private. This follows from the fact that changing one entry z_i in the database z can change the score by at most 1.

Lemma 3.3 (following [44])**.** *The algorithm $\mathcal{A}_q^\epsilon$ is ϵ-differentially private.*

A similar exponential weighting algorithm was considered by Freund, Mansour and Schapire [31] for constructing binary classifiers with good generalization error bounds. We are not aware of any direct connection between the two results. Also note that, except for the case where $|\mathcal{H}_d|$ is polynomial, the exponential mechanism $\mathcal{A}_q^\epsilon(\mathrm{z})$ does not necessarily yield a polynomial time algorithm.

Theorem 3.4 (Generic Private Learner)**.** *For all $d \in \mathbb{N}$, any concept class $\mathcal{C}_d$ whose cardinality is at most $\exp(\mathrm{poly}(d))$ is privately agnostically learnable using $\mathcal{H}_d = \mathcal{C}_d$. More precisely, the learner uses $n = O((\ln|\mathcal{H}_d| + \ln\frac{1}{\beta}) \cdot \max\{\frac{1}{\epsilon\alpha}, \frac{1}{\alpha^2}\})$ labeled examples from $\mathcal{D}$, where ϵ, α, and β are parameters of the private learner. (The learner might not be efficient.)*

Proof. Let $\mathcal{A}_q^\epsilon$ be as defined above. The privacy condition in Definition 3.1 is satisfied by Lemma 3.3.

We now show that the utility condition is also satisfied. Consider the event $E = \{\mathcal{A}_q^\epsilon(\mathrm{z}) = h \text{ with } \textit{err}(h) > \alpha + OPT\}$. We want to prove that $\Pr[E] \leq \beta$. Define the training error of h as

$$\textit{err}_T(h) = \big|\{i \in [n] \,|\, h(x_i) \neq y_i\}\big|/n = -q(\mathrm{z}, h)/n\,.$$

By Chernoff-Hoeffding bounds (see Theorem A.2 in Appendix A),

$$\Pr\left[|\textit{err}(h) - \textit{err}_T(h)| \geq \rho\right] \leq 2\exp(-2n\rho^2)$$

for all hypotheses $h \in \mathcal{H}_d$. Hence,

$$\Pr\left[|\textit{err}(h) - \textit{err}_T(h)| \geq \rho \text{ for some } h \in \mathcal{H}_d\right] \leq 2|\mathcal{H}_d|\exp(-2n\rho^2).$$

[3] We discuss the relationship to the "compression version" of Occam's razor at the end of this section.

We now analyze $\mathcal{A}_q^\epsilon(\mathrm{z})$ conditioned on the event that for all $h \in \mathcal{H}_d$, $|err(h) - err_T(h)| < \rho$. For every $h \in \mathcal{H}_d$, the probability that $\mathcal{A}_q^\epsilon(\mathrm{z}) = h$ is

$$\begin{aligned}
\frac{\exp(-\frac{\epsilon}{2} \cdot n \cdot err_T(h))}{\sum_{h' \in \mathcal{H}_d} \exp(-\frac{\epsilon}{2} \cdot n \cdot err_T(h'))} &\leq \frac{\exp\left(-\frac{\epsilon}{2} \cdot n \cdot err_T(h)\right)}{\max_{h' \in \mathcal{H}_d} \exp(-\frac{\epsilon}{2} \cdot n \cdot err_T(h'))} \\
&= \exp\left(-\frac{\epsilon}{2} \cdot n \cdot (err_T(h) - \min_{h' \in \mathcal{H}_d} err_T(h'))\right) \\
&\leq \exp\left(-\frac{\epsilon}{2} \cdot n \cdot (err_T(h) - (OPT + \rho))\right).
\end{aligned}$$

Hence, the probability that $\mathcal{A}_q^\epsilon(\mathrm{z})$ outputs a hypothesis $h \in \mathcal{H}_d$ such that $err_T(h) \geq OPT + 2\rho$ is at most $|\mathcal{H}_d| \exp(-\epsilon n \rho/2)$.

Now set $\rho = \alpha/3$. If $err(h) \geq OPT + \alpha$ then $|err(h) - err_T(h)| \geq \alpha/3$ or $err_T(h) \geq OPT + 2\alpha/3$. Thus $\Pr[E] \leq |\mathcal{H}_d|(2\exp(-2n\alpha^2/9) + \exp(-\epsilon n \alpha/6)) \leq \beta$ where the last inequality holds for $n \geq 6\left((\ln|\mathcal{H}_d| + \ln\frac{1}{\beta}) \cdot \max\{\frac{1}{\epsilon\alpha}, \frac{1}{\alpha^2}\}\right)$. □

Remark: In the non-private agnostic case, the standard Occam's razor bound guarantees that $O((\log|\mathcal{C}_d| + \log(1/\beta))/\alpha^2)$ labeled examples suffice to agnostically learn a concept class $\mathcal{C}_d$. The bound of Theorem 3.4 differs by a factor of $O(\frac{\alpha}{\epsilon})$ if $\alpha > \epsilon$, and does not differ at all otherwise. For (non-agnostic) PAC learning, the dependence on α in the sample size for both the private and non-private versions improves to $1/\alpha$. In that case the upper bounds for private and non-private learners differ by a factor of $O(1/\epsilon)$. Finally, the theorem can be extended to settings where $\mathcal{H}_d \neq \mathcal{C}_d$, but in this case using the same sample complexity the learner outputs a hypothesis whose error is close to the best error attainable by a function in $\mathcal{H}_d$.

Implications of the Private Agnostic Learner The private agnostic learner has the following important consequence: If some concept class $\mathcal{C}_d$ is learnable by any algorithm $\mathcal{A}$, not necessarily a private one, and $\mathcal{A}$'s output length in bits is polynomially bounded, then there is a (possibly exponential time) private algorithm that learns $\mathcal{C}_d$ using a polynomial number of samples. Since $\mathcal{A}$'s output is polynomially long, $\mathcal{A}$'s hypothesis class $\mathcal{H}_d$ must have size at most $2^{\mathrm{poly}(d)}$. Since $\mathcal{A}$ learns $\mathcal{C}_d$ using $\mathcal{H}_d$, class $\mathcal{H}_d$ must contain a good hypothesis. Thus, our private learner will learn $\mathcal{C}_d$ using $\mathcal{H}_d$ with sample complexity linear in $\log|\mathcal{H}_d|$.

The "compression version" of Occam's razor It is most natural to state our result as an analogue of the cardinality version of Occam's razor, which bounds generalization error in terms of the size of the hypothesis class. However, our result can be extended to the compression version, which captures the general relationship between compression and learning (we borrow the "cardinality version" terminology from [42]). This latter version states that any algorithm which "compresses" the data set, in the sense that it finds a consistent hypothesis which has a short description relative to the number of samples seen so far, is a good learner (see [15] and [42, p. 34]).

Compression by itself does not imply privacy, because the compression algorithm's output might encode a few examples in the clear (for example, the hyperplane output by a support vector machine is defined via a small number of actual data points). However, Theorem 3.4 can be extended to provide a private analogue of the compression version of Occam's razor. If there exists an algorithm that compresses, in the sense above, then there also exists a private PAC learner which does not have fixed sample complexity, but uses an expected number of samples similar to that of the compression algorithm. The private learner proceeds in rounds: at each round it requests twice as many examples as in the previous round, and uses a

restricted hypothesis class consisting of sufficiently concise hypotheses from the original class $\mathcal{H}$. We omit the straightforward details.

3.2 Private Learning with VC dimension Sample Bounds

In the non-private case one can also bound the sample size of a PAC learner in terms of the Vapnik-Chervonenkis (VC) dimension of the concept class.

Definition 3.5 (VC dimension)**.** *A set $S \subseteq X_d$ is* shattered *by a concept class $\mathcal{C}_d$ if $\mathcal{C}_d$ restricted to S contains all $2^{|S|}$ possible functions from S to $\{0,1\}$. The* VC dimension *of $\mathcal{C}_d$, denoted $VCDIM(\mathcal{C}_d)$, is the cardinality of a largest set S shattered by $\mathcal{C}_d$.*

We can extend Theorem 3.4 to classes with finite VC dimension, but the resulting sample complexity also depends logarithmically on the size of the domain from which examples are drawn. Recent results of Beimel *et al.* [8] show that for "proper" learning, the dependency is in fact necessary; that is, the VC dimension alone is not sufficient to bound the sample complexity of proper private learning. It is unclear if the dependency is necessary in general.

Corollary 3.6. *Every concept class $\mathcal{C}_d$ is privately agnostically learnable using hypothesis class $\mathcal{H}_d = \mathcal{C}_d$ with $n = O((VCDIM(\mathcal{C}_d) \cdot \ln|X_d| + \ln\frac{1}{\beta}) \cdot \max\{\frac{1}{\epsilon\alpha}, \frac{1}{\alpha^2}\})$ labeled examples from $\mathcal{D}$. Here, ϵ, α, and β are parameters of the private agnostic learner, and $VCDIM(\mathcal{C}_d)$ is the VC dimension of $\mathcal{C}_d$. (The learner is not necessarily efficient.)*

Proof. Sauer's lemma (see, *e.g.*, [42]) implies that there are $O(|X_d|^{VCDIM(\mathcal{C}_d)})$ different labelings of X_d by functions in $\mathcal{C}_d$. We can thus run the generic learner of the previous section with a hypothesis class of size $|\mathcal{H}_d| = O(|X_d|^{VCDIM(\mathcal{C}_d)})$. The statement follows directly. □

Our original proof of the corollary used a result of Blum, Ligget and Roth [14] (which was inspired, in turn, by our generic learning algorithm) on generating synthetic data. The simpler proof above was pointed out to us by an anonymous reviewer.

Remark: Computability Issues with Generic Learners In their full generality, the generic learning results of the previous sections (Theorems 3.4 and 3.6) produce well-defined randomized maps, but not necessarily "algorithms" in the sense of "functions uniformly computable by Turing machines". This is because the concept class and example domain may themselves not be computable (nor even recognizable) uniformly (imagine, for example, a concept class indexed by elements of the halting problem). It is commonly assumed in the learning literature that elements of the concept class and domain can be computed/recognized by a Turing machine and some bound on the length of their binary representations is known. In this case, the generic learners can be implemented by randomized Turing machines with finite expected running time.

4 An Efficient Private Learner for PARITY

Let PARITY be the class of parity functions $c_r : \{0,1\}^d \to \{0,1\}$ indexed by $r \in \{0,1\}^d$, where $c_r(x) = r \odot x$ denotes the inner product modulo 2. In this section, we present an efficient private PAC learning algorithm for PARITY. The main result is stated in Theorem 4.4.

The standard (non-private) PAC learner for PARITY [33, 30] looks for the hidden vector r by solving a system of linear equations imposed by examples $(x_i, c_r(x_i))$ that the algorithm sees. It outputs an arbitrary

vector consistent with the examples, i.e., in the solution space of the system of linear equations. We want to design a private algorithm that emulates this behavior. A major difficulty is that the private learner's behavior must be specified on *all* databases z, even those which are not consistent with any single parity function. The standard PAC learner would simply fail in such a situation (we denote failure by the output $\perp$). In contrast, the probability that a private algorithm fails must be similar for all neighbors z and z$'$.

We first present a private algorithm $\mathcal{A}$ for learning PARITY that succeeds only with constant probability. Later we amplify its success probability and get a private PAC learner $\mathcal{A}^*$ for PARITY. Intuitively, the reason PARITY can be learned privately is that when a new example (corresponding to a new linear constraint) is added, the space of consistent hypotheses shrinks by at most a factor of 2. This holds unless the new constraint is inconsistent with previous constraints. In the latter case, the size of the space of consistent hypotheses goes to 0. Thus, the solution space changes drastically on neighboring inputs only when the algorithm fails (outputs $\perp$). The fact that algorithm outputs $\perp$ on a database z and a valid (non $\perp$) hypothesis on a neighboring database z$'$ might lead to privacy violations. To avoid this, our algorithm always outputs $\perp$ with probability at least $1/2$ on any input (Step 1).

A PRIVATE LEARNER FOR PARITY, $\mathcal{A}(\mathrm{z}, \epsilon)$

1. With probability $1/2$, output $\perp$ and terminate.
2. Construct a set S by picking each element of $[n]$ independently with probability $p = \epsilon/4$.
3. Use Gaussian elimination to solve the system of equations imposed by examples, indexed by S: namely, $\{x_i \odot r = c_r(x_i) \,:\, i \in S\}$. Let V_S denote the resulting affine subspace.
4. Pick $r^* \in V_S$ uniformly at random and output c_{r^*}; if $V_S = \emptyset$, output $\perp$.

The proof of $\mathcal{A}$'s utility follows by considering all the possible situations in which the algorithm fails to satisfy the error bound, and by bounding the probabilities with which these situations occur.

Lemma 4.1 (Utility of $\mathcal{A}$)**.** *Let $\mathcal{X}$ be a distribution over $X = \{0,1\}^d$. Let $\mathrm{z} = (z_1, \ldots, z_n)$, where for all $i \in [n]$, the entry $z_i = (x_i, c(x_i))$ with x_i drawn i.i.d. from $\mathcal{X}$ and $c \in$ PARITY. If $n \geq \frac{8}{\epsilon\alpha}(d \ln 2 + \ln 4)$ then*

$$\Pr[\mathcal{A}(\mathrm{z}, \epsilon) = h \text{ with } error(h) \leq \alpha] \geq \frac{1}{4}\,.$$

Proof. By standard arguments in learning theory [42], $|S| \geq \frac{1}{\alpha}\left(d \ln 2 + \ln \frac{1}{\beta}\right)$ labeled examples are sufficient for learning PARITY with error α and failure probability β. Since $\mathcal{A}$ adds each element of $[n]$ to S independently with probability $p = \epsilon/4$, the expected size of S is $pn = \epsilon n/4$. By the Chernoff bound (Theorem A.1), $|S| \geq \epsilon n/8$ with probability at least $1 - e^{-\epsilon n/16}$. We set $\beta = \frac{1}{4}$ and pick n such that $\epsilon n/8 \geq \frac{1}{\alpha}(d \ln 2 + \ln 4)$.

We now bound the overall success probability. $\mathcal{A}(\mathrm{z}, \epsilon) = h$ with $err(h) \leq \alpha$ unless one of the following bad events happens: (i) $\mathcal{A}$ terminates in Step 1, (ii) $\mathcal{A}$ proceeds to Step 2, but does not get enough examples: $|S| < \frac{1}{\alpha}(d \ln 2 + \ln 4)$), (iii) $\mathcal{A}$ gets enough examples, but outputs a hypothesis with error greater than α. The first bad event occurs with probability 1/2. If the lower bound on the database size n is satisfied then the second bad event occurs with probability at most $e^{-\epsilon n/16}/2 \leq 1/8$. The last inequality follows from the bound on n and the fact that $\alpha \leq 1/2$. Finally, by our choice of parameters, the last bad event occurs with probability at most $\beta/2 = 1/8$. The claimed bound on the success probability follows. □

Lemma 4.2 (Privacy of $\mathcal{A}$)**.** *Algorithm $\mathcal{A}$ is ϵ-differentially private.*

As mentioned above, the key observation in the following proof is that including of any single point in the sample set S increases the probability of a hypothesis being output by at most 2.

Proof. To show that $\mathcal{A}$ is ϵ-differentially private, it suffices to prove that any output of $\mathcal{A}$, either a valid hypothesis or $\perp$, appears with roughly the same probability on neighboring databases z and z'. In the remainder of the proof we fix ϵ, and write $\mathcal{A}(\mathrm{z})$ as shorthand for $\mathcal{A}(\mathrm{z}, \epsilon)$. We have to show that

$$\Pr[\mathcal{A}(\mathrm{z}) = h] \leq e^{\epsilon} \cdot \Pr[\mathcal{A}(\mathrm{z}') = h] \quad \text{for all neighbors } \mathrm{z}, \mathrm{z}' \in D^n \text{ and all hypotheses } h \in \mathsf{PARITY}; \tag{2}$$

$$\Pr[\mathcal{A}(\mathrm{z}) = \perp] \leq e^{\epsilon} \cdot \Pr[\mathcal{A}(\mathrm{z}') = \perp] \quad \text{for all neighbors } \mathrm{z}, \mathrm{z}' \in D^n. \tag{3}$$

We prove the correctness of Eqn. (2) first. Let z and z' be neighboring databases, and let i denote the entry on which they differ. Recall that $\mathcal{A}$ adds i to S with probability p. Since z and z' differ only in the i^{th} entry, $\Pr[\mathcal{A}(\mathrm{z}) = h \mid i \notin S] = \Pr[\mathcal{A}(\mathrm{z}') = h \mid i \notin S]$.

Note that if $\Pr[\mathcal{A}(\mathrm{z}') = h \mid i \notin S] = 0$, then also $\Pr[\mathcal{A}(\mathrm{z}) = h \mid i \notin S] = 0$, and hence $\Pr[\mathcal{A}(\mathrm{z}) = h] = 0$ because adding a constraint does not add new vectors to the space of solutions. Otherwise, $\Pr[\mathcal{A}(\mathrm{z}') = h \mid i \notin S] > 0$. In this case, we rewrite the probability on z as follows:

$$\Pr[\mathcal{A}(\mathrm{z}) = h] = p \cdot \Pr[\mathcal{A}(\mathrm{z}) = h \mid i \in S] + (1-p) \cdot \Pr[\mathcal{A}(\mathrm{z}) = h \mid i \notin S],$$

and apply the same transformation to the probability on z'. Then

$$\begin{aligned}
\frac{\Pr[\mathcal{A}(\mathrm{z}) = h]}{\Pr[\mathcal{A}(\mathrm{z}') = h]} &= \frac{p \cdot \Pr[\mathcal{A}(\mathrm{z}) = h \mid i \in S] + (1-p) \cdot \Pr[\mathcal{A}(\mathrm{z}) = h \mid i \notin S]}{p \cdot \Pr[\mathcal{A}(\mathrm{z}') = h \mid i \in S] + (1-p) \cdot \Pr[\mathcal{A}(\mathrm{z}') = h \mid i \notin S]} \\
&\leq \frac{p \cdot \Pr[\mathcal{A}(\mathrm{z}) = h \mid i \in S] + (1-p) \cdot \Pr[\mathcal{A}(\mathrm{z}) = h \mid i \notin S]}{p \cdot 0 + (1-p) \cdot \Pr[\mathcal{A}(\mathrm{z}') = h \mid i \notin S]} \\
&= \frac{p}{1-p} \cdot \frac{\Pr[\mathcal{A}(\mathrm{z}) = h \mid i \in S]}{\Pr[\mathcal{A}(\mathrm{z}) = h \mid i \notin S]} + 1
\end{aligned} \tag{4}$$

We need the following claim:

Claim 4.3. $\dfrac{\Pr[\mathcal{A}(\mathrm{z}) = h \mid i \in S]}{\Pr[\mathcal{A}(\mathrm{z}) = h \mid i \notin S]} \leq 2$*, for all* $\mathrm{z} \in D^n$ *and all hypotheses* $h \in \mathsf{PARITY}$.

This claim is proved below. For now, we can plug it into Eqn. (4) to get

$$\frac{\Pr[\mathcal{A}(\mathrm{z}) = h]}{\Pr[\mathcal{A}(\mathrm{z}') = h]} \leq \frac{2p}{1-p} + 1 \leq \epsilon + 1 \leq e^{\epsilon}\,.$$

The first inequality holds since $p = \epsilon/4$ and $\epsilon \leq 1/2$. This establishes Eqn. (2). The proof of Eqn. (3) is similar:

$$\begin{aligned}
\frac{\Pr[\mathcal{A}(\mathrm{z}) = \perp]}{\Pr[\mathcal{A}(\mathrm{z}') = \perp]} &= \frac{p \cdot \Pr[\mathcal{A}(\mathrm{z}) = \perp \mid i \in S] + (1-p) \cdot \Pr[\mathcal{A}(\mathrm{z}) = \perp \mid i \notin S]}{p \cdot \Pr[\mathcal{A}(\mathrm{z}') = \perp \mid i \in S] + (1-p) \cdot \Pr[\mathcal{A}(\mathrm{z}') = \perp \mid i \notin S]} \\
&\leq \frac{p \cdot 1 + (1-p) \cdot \Pr[\mathcal{A}(\mathrm{z}) = \perp \mid i \notin S]}{p \cdot 0 + (1-p) \cdot \Pr[\mathcal{A}(\mathrm{z}') = \perp \mid i \notin S]} \\
&= \frac{p}{(1-p) \cdot \Pr[\mathcal{A}(\mathrm{z}') = \perp \mid i \notin S]} + 1 \leq \frac{2p}{1-p} + 1 \leq \epsilon + 1 \leq e^{\epsilon}.
\end{aligned}$$

In the last line, the first inequality follows from the fact that on any input, $\mathcal{A}$ outputs $\perp$ with probability at least $1/2$. This completes the proof of the lemma. □

We now prove Claim 4.3.

Proof of Claim 4.3. The left hand side

$$\frac{\Pr[\mathcal{A}(\mathrm{z}) = h \mid i \in S]}{\Pr[\mathcal{A}(\mathrm{z}) = h \mid i \notin S]} = \frac{\sum_{T \subseteq [n] \setminus \{i\}} \Pr[\mathcal{A}(\mathrm{z}) = h \mid S = T \cup \{i\}] \cdot \Pr[\mathcal{A} \text{ selects } T \text{ from } [n] \setminus \{i\}]}{\sum_{T \subseteq [n] \setminus \{i\}} \Pr[\mathcal{A}(\mathrm{z}) = h \mid S = T] \cdot \Pr[\mathcal{A} \text{ selects } T \text{ from } [n] \setminus \{i\}]}.$$

To prove the claim, it is enough to show that $\frac{\Pr[\mathcal{A}(\mathrm{z}) = h \mid S = T \cup \{i\}]}{\Pr[\mathcal{A}(\mathrm{z}) = h \mid S = T]} \leq 2$ for each $T \subseteq [n] \setminus \{i\}$. Recall that V_S is the space of solutions to the system of linear equations $\{\langle x_i, r \rangle = c_r(x_i) \,:\, i \in S\}$. Recall also that $\mathcal{A}$ picks $r^* \in V_S$ uniformly at random and outputs $h = c_{r^*}$. Therefore,

$$\Pr[\mathcal{A}(\mathrm{z}) = c_{r^*} \mid S] = \begin{cases} 1/|V_S| & \text{if } r^* \in V_S, \\ 0 & \text{otherwise.} \end{cases}$$

If $\Pr[\mathcal{A}(\mathrm{z}) = h \mid S = T] = 0$ then $\Pr[\mathcal{A}(\mathrm{z}) = h \mid S = T \cup \{i\}] = 0$ because a new constraint does not add new vectors to the space of solutions. If $\Pr[\mathcal{A}(\mathrm{z}) = h \mid S = T \cup \{i\}] = 0$, the required inequality holds. If neither of the two probabilities is 0,

$$\frac{\Pr[\mathcal{A}(\mathrm{z}) = h \mid S = T \cup \{i\}]}{\Pr[\mathcal{A}(\mathrm{z}) = h \mid S = T]} = \frac{1/|V_{T \cup \{i\}}|}{1/|V_T|} = \frac{|V_T|}{|V_{T \cup \{i\}}|} \leq 2.$$

The last inequality holds because in $\mathbb{Z}_2$ (the finite field with 2 elements where arithmetic is performed modulo 2), adding a consistent linear constraint either reduces the space of solutions by a factor of 2 (if the constraint is linearly independent from V_T) or does not change the solutions space (if it is linearly dependent on the previous constraints). The constraint indexed by i has to be consistent with constraints indexed by T, since both probabilities are not 0. □

It remains to amplify the success probability of $\mathcal{A}$. To do so, we construct a private version of the standard (non-private) algorithm for amplifying a learner's success probability. The standard amplification algorithm generates a set of hypotheses by invoking $\mathcal{A}$ multiple times on independent examples, and then outputs a hypothesis from the set with the least training error as evaluated on a fresh test set (see [42] for details). Our private amplification algorithm differs from the standard algorithm only in the last step: it adds Laplacian noise to the training error to obtain a private version of the error, and then uses the perturbed training error instead of the true training error to select the best hypothesis from the set. [4] Recall that $\mathrm{Lap}(\lambda)$ denotes the Laplace probability distribution with mean 0, standard deviation $\sqrt{2}\lambda$, and p.d.f. $f(x) = \frac{1}{2\lambda} e^{-|x|/\lambda}$.

[4] Alternatively, we could use the generic learner from Theorem 3.4 to select among the candidate hypotheses; the resulting algorithm has the same asymptotic behavior as the algorithm we discuss here. We chose the algorithm that we felt was simplest.

AMPLIFIED PRIVATE PAC LEARNER FOR PARITY, $\mathcal{A}^*(\mathrm{z}, \epsilon, \alpha, \beta)$

1. $\beta' \leftarrow \frac{\beta}{2}$; $\alpha' \leftarrow \frac{\alpha}{5}$; $k \leftarrow \left\lceil \log_{\frac{3}{4}}\left(\frac{1}{\beta'}\right)\right\rceil$; $n' \leftarrow \frac{cd}{\epsilon\alpha'}$; $s \leftarrow \frac{c'k}{\alpha'\epsilon}\log\left(\frac{k}{\beta'}\right)$ (where c, c' are constants).
2. If $n \leq kn' + s$, stop and return "insufficient samples".
3. Divide $\mathrm{z} = (z_1, \ldots, z_n)$ into two parts, training set $\bar{\mathrm{z}} = (z_1, \ldots, z_{kn'})$ and test set $\hat{\mathrm{z}} = (z_{kn'+1}, \ldots, z_{kn'+s})$.
4. Divide $\bar{\mathrm{z}}$ into k equal parts each of size n', let $\bar{\mathrm{z}}_j = (z_{(j-1)n'+1}, \ldots, z_{jn'})$ for $j \in [k]$.
5. For $j \leftarrow 1$ to k
   $h_j \leftarrow \mathcal{A}(\bar{\mathrm{z}}_j, \epsilon)$;
   set perturbed training error of h_j to $\widehat{err}_T(h_j) = \dfrac{\left|\{z_i \in \hat{\mathrm{z}} : h_j(x_i) \neq c(x_i)\}\right|}{s} + \mathrm{Lap}\left(\dfrac{k}{s\epsilon}\right)$.
6. Output $h^* = h_{j^*}$ where $j^* = \mathrm{argmin}_{j\in[k]}\{\widehat{err}_T(h_j)\}$.

Theorem 4.4. *Algorithm $\mathcal{A}^*$ efficiently and privately PAC learns* PARITY *(according to Definition 3.1) with $O\left(\frac{d\log(1/\beta)}{\epsilon\alpha}\right)$ samples.*

The theorem follows from Lemmas 4.5 and 4.6 that, respectively, prove privacy and utility of $\mathcal{A}^*$.

Lemma 4.5 (Privacy of $\mathcal{A}^*$)**.** *Algorithm $\mathcal{A}^*$ is ϵ-differentially private.*

Proof. We prove that even if $\mathcal{A}^*$ released all hypotheses h_j, computed in Step 5, together with the corresponding perturbed error estimates $\widehat{err}_T(h_j)$, it would still be ϵ-differentially private. Since the output of $\mathcal{A}^*$ can be computed solely from this information, Claim 2.2 implies that $\mathcal{A}^*$ is ϵ-differentially private.

By Lemma 4.2, algorithm $\mathcal{A}$ is ϵ-differentially private. Since $\mathcal{A}$ is invoked on disjoint parts of z to compute hypotheses h_j, releasing all these hypotheses would also be ϵ-differentially private.

Define the training error of hypothesis h_j on $\hat{\mathrm{z}}$ as $err_T(h_j) = |\{z_i \in \hat{\mathrm{z}} : h_j(x_i) \neq c(x_i)\}|/s$. The global sensitivity of the err_T function is $1/s$ because $|err_T(\mathrm{z}) - err_T(\mathrm{z}')| \leq 1/s$ for every pair of neighboring databases z, z'. Therefore, by Theorem 2.3, releasing $\widehat{err}_T(h_j)$ for one j, would be ϵ/k-differentially private, and by Claim 2.2, releasing all k of them would be ϵ-differentially private. Since hypotheses h_j and their perturbed errors $\widehat{err}_T(h_j)$ are computed on disjoint parts of the database z, releasing all that information would still be ϵ-differentially private. □

Lemma 4.6 (Utility of $\mathcal{A}^*$)**.** *$\mathcal{A}^*(\cdot, \epsilon, \cdot, \cdot)$ PAC learns* PARITY *with sample complexity $n = O(\frac{d\log(1/\beta)}{\epsilon\alpha})$.*

Proof. Let $\mathcal{X}$ be a distribution over $X = \{0,1\}^d$. Recall that $\mathrm{z} = (z_1, \ldots, z_n)$, where for all $i \in [n]$, the entry $z_i = (x_i, c(x_i))$ with x_i drawn i.i.d. from $\mathcal{X}$ and $c \in$ PARITY. Assume that $\beta < 1/4$, and $n \geq C\frac{d\log(1/\beta)}{\epsilon\alpha}$ for a constant C to be determined. We wish to prove that $\Pr[err(h^*) \leq \alpha] \geq 1 - \beta$, where h^* is the hypothesis output by $\mathcal{A}^*$.

Consider the set of candidate hypotheses $\{h_1, ..., h_k\}$ output by the invocations of $\mathcal{A}$ inside of $\mathcal{A}^*$. We call a hypothesis h *good* if $err(h) \leq \frac{\alpha}{5} = \alpha'$. We call a hypothesis h *bad* if $err(h) \geq \alpha = 5\alpha'$. Note that good and bad refer to a hypothesis' true error rate on the underlying distribution.

We will show:

1. With probability at least $1 - \beta'$, one of the invocations of $\mathcal{A}$ outputs a *good* hypothesis.

2. Conditioned on any particular outcome $\{h_1, ..., h_k\}$ of the invocations of $\mathcal{A}$, with probability at least $1 - \beta'$, both:

   (a) Every *good* hypothesis h_j in $\{h_1, ..., h_k\}$ has training error $err_T(h_j) \leq 2\alpha'$.

   (b) Every *bad* hypothesis h_j in $\{h_1, ..., h_k\}$ has training error $err_T(h_j) \geq 4\alpha'$.

3. Conditioned on any particular hypotheses $\{h_1, ..., h_k\}$ and training errors $err_T(h_1), ..., err_T(h_k)$, with probability at least $1 - \beta'$, for all j simultaneously, $|\widehat{err}_T(h_j) - err_T(h_j)| < \alpha'$.

Suppose the events described in the three claims above all occur. Then some good hypothesis has perturbed training error less than $3\alpha'$, yet all bad hypotheses have perturbed training error greater than $3\alpha'$. Thus, the hypothesis h_{j^*} with minimal perturbed error $\widehat{err}_T(h_{j^*})$ is *not bad*, that is, has true error at most α. By the claims above, the probability that all three events occur is at least $1 - 3\beta' = 1 - \beta$, and so the lemma holds. We now prove the claims.

First, by the utility guarantee of $\mathcal{A}$, each invocation of $\mathcal{A}$ inside $\mathcal{A}^*$ outputs a *good* hypothesis with probability at least $\frac{1}{4}$ as long as the constant $c > 8(\ln 2 + \ln 4)$ (since in that case n', the size of each $\bar{z}_j$, is large enough to apply Lemma 4.1). The k invocations of the algorithm $\mathcal{A}$ are on independent samples, so the probability that none of $h_1, \ldots, h_k$ is good is at most $\left(\frac{3}{4}\right)^k$. Setting $k \geq \log_{\frac{3}{4}} \frac{1}{\beta'}$ ensures that with probability at least $1 - \beta'$, at least one of $h_1, \ldots, h_k$ has error at most α'.

Second, fix a particular sequence of candidate hypotheses $h_1, ..., h_k$. For each j, the training error $err_T(h_j)$ is the average of s Bernouilli trials, each with success probability $err(h_j)$. (Crucially, the training set $\hat{z}$ is independent of the data $\bar{z}$ used to find the candidate hypotheses). To bound the training error, we apply the multiplicative Chernoff bound (Theorem A.1) with $n = s$ and $p = err(h_j)$. Here, $p \leq \alpha'$ if h_j is *good*, and $p \geq 5\alpha'$ if h_j is *bad*.

By the multiplicative Chernoff bound (Theorem A.1) if $s \geq \frac{c_1}{\alpha'} \ln \frac{k}{\beta'}$ (for appropriate constant c_1), then

$$\Pr\left[err_T(h_j) \geq 2\alpha' \,\middle|\, h_j \text{ is good}\right] \leq \Pr[\text{Binomial}(s, \alpha') \geq 2\alpha' s] \leq \frac{\beta'}{k}, \text{ and}$$
$$\Pr\left[err_T(h_j) \leq 4\alpha' \,\middle|\, h_j \text{ is bad}\right] \leq \Pr[\text{Binomial}(s, 5\alpha') \leq 4\alpha' s] \leq \frac{\beta'}{k}.$$

By a union bound, all the training errors are (simultaneously) approximately correct, with probability at least $1 - k \cdot \frac{\beta'}{k} = 1 - \beta'$.

Finally, we prove the third claim. Consider a particular candidate hypothesis h_j. If $s \geq \frac{c_2 k}{\alpha' \epsilon} \ln \frac{k}{\beta'}$ (for appropriate constant c_2), then (by using the c.d.f.[5] of the Laplacian distribution)

$$\Pr\left[|err_T(h_j) - \widehat{err}_T(h_j)| < \alpha'\right] = \Pr\left[\text{Lap}\left(\frac{k}{s\epsilon}\right) \geq \alpha'\right] \leq \frac{\beta'}{k}.$$

By a union bound, all k perturbed estimates are within α' of their correct value with probability at least $1 - k \cdot \frac{\beta'}{k} = 1 - \beta'$. This probability is taken over the choice of Laplacian noise, and so the bound holds independently of the particular hypotheses or their training error estimates. □

Remark: In the non-private case $O((d+\ln(1/\beta))/\alpha)$ labels are sufficient for learning PARITY. Theorem 4.4 shows that the upper bounds on the sample size of private and non-private learners differ only by a factor of $O(\ln(1/\beta)/\epsilon)$.

[5] The cumulative distribution function of the Laplacian distribution $\text{Lap}(\lambda)$ is $F(x) = \frac{1}{2}\exp\left(\frac{x}{\lambda}\right)$ if $x < 0$ and $1 - \frac{1}{2}\exp\left(-\frac{x}{\lambda}\right)$ if $x \geq 0$.

5 Local Protocols and SQ learning

In this section, we relate private learning in the local model to the SQ model of Kearns [39]. We first define the two models precisely. We then prove their equivalence (Section 5.1), and discuss the implications for learning (Section 5.2). Finally, we define the concept class MASKED-PARITY and prove that it separates interactive from noninteractive local learning (Section 5.3).

Local Model. We start by describing private computation in the local model. Informally, each individual holds her private information locally, and hands it to the learner after randomizing it. This is modeled by letting the local algorithm access each entry z_i in the input database $\mathrm{z} = (z_1, \ldots, z_n) \in D^n$ only via *local randomizers*.

Definition 5.1 (Local Randomizer)**.** *An* ϵ-local randomizer $R : D \to W$ *is an* ϵ*-differentially private algorithm that takes a database of size* $n = 1$*. That is,* $\Pr[R(u) = w] \leq e^{\epsilon} \Pr[R(u') = w]$ *for all* $u, u' \in D$ *and all* $w \in W$*. The probability is taken over the coins of* R *(but* not *over the choice of the input).*

Note that since a local randomizer works on a data set of size 1, u and u' are neighbors for all $u, u' \in D$. Thus, this definition is consistent with our previous definition of differential privacy.

Definition 5.2 (LR Oracle)**.** *Let* $\mathrm{z} = (z_1, \ldots, z_n) \in D^n$ *be a database. An* LR oracle $LR_{\mathrm{z}}(\cdot, \cdot)$ *gets an index* $i \in [n]$ *and an* ϵ*-local randomizer* R*, and outputs a random value* $w \in W$ *chosen according to the distribution* $R(z_i)$*. The distribution* $R(z_i)$ *depends only on the entry* z_i *in* z*.*

Definition 5.3 (Local algorithm)**.** *An algorithm is* ϵ-local *if it accesses the database* z *via the oracle* LR_{z} *with the following restriction: for all* $i \in [n]$*, if* $LR_{\mathrm{z}}(i, R_1), \ldots, LR_{\mathrm{z}}(i, R_k)$ *are the algorithm's invocations of* LR_{z} *on index* i*, where each* R_j *is an* ϵ_j*-local randomizer, then* $\epsilon_1 + \cdots + \epsilon_k \leq \epsilon$*.*

Local algorithms that prepare all their queries to LR_{z} *before receiving any answers are called* noninteractive*; otherwise, they are* interactive*.*

By Claim 2.2, ϵ-local algorithms are ϵ-differentially private.

SQ Model. In the statistical query (SQ) model, algorithms access statistical properties of a distribution rather than individual examples.

Definition 5.4 (SQ Oracle)**.** *Let* $\mathcal{D}$ *be a distribution over a domain* D*. An* SQ oracle $SQ_{\mathcal{D}}$ *takes as input a function* $g : D \to \{+1, -1\}$ *and a tolerance parameter* $\tau \in (0, 1)$*; it outputs* v *such that:*

$$|v - \mathop{\mathbb{E}}_{u \sim \mathcal{D}}[g(u)]| \leq \tau.$$

The query function g does not have to be Boolean. Bshouty and Feldman [17] showed that given access to an SQ oracle which accepts only boolean query functions, one can simulate an oracle that accepts real-valued functions $g : D \to [-b, b]$, and outputs $\mathbb{E}_{u \sim \mathcal{D}}[g(u)] \pm \tau$ using $O(\log(b/\tau))$ nonadaptive queries to the SQ oracle and similar processing time.

Definition 5.5 (SQ algorithm)**.** *An* SQ algorithm *accesses the distribution* $\mathcal{D}$ *via the SQ oracle* $SQ_{\mathcal{D}}$*. SQ algorithms that prepare all their queries to* $SQ_{\mathcal{D}}$ *before receiving any answers are called* nonadaptive*; otherwise, they are called* adaptive*.*

Note that we do not restrict $g()$ to be efficiently computable. We will distinguish later those algorithms that only make queries to efficiently computable functions $g()$.

5.1 Equivalence of Local and SQ Models

Both the SQ and local models restrict algorithms to access inputs in a particular manner. There is a significant difference though: an SQ oracle sees a distribution $\mathcal{D}$, whereas a local algorithm takes as input a fixed (arbitrary) database z. Nevertheless, we show that if the entries of z are chosen i.i.d. according to $\mathcal{D}$, then the models are equivalent. Specifically, an algorithm in one model can *simulate* an algorithm in the other model. Moreover, the expected query complexity is preserved up to polynomial factors. We first present the simulation of SQ algorithms by local algorithms (Section 5.1.1). The simulation in the other direction is more delicate and is presented in Section 5.1.2.

5.1.1 Simulation of SQ Algorithms by Local Algorithms

Blum *et al.* [11] used the fact that sum queries can be answered privately with little noise to show that any efficient SQ algorithm can be simulated privately and efficiently. We show that it can be simulated efficiently even by a local algorithm, albeit with slightly worse parameters.

Let $g : D \to [-b, b]$ be the SQ query we want to simulate. By Theorem 2.3, since the global sensitivity of g is $2b$, the algorithm $R_g(u) = g(u) + \eta$ where $\eta \sim \mathrm{Lap}(2b/\epsilon)$ is an ϵ-local randomizer. We construct a local algorithm $\mathcal{A}_g$ that, given n and ϵ, and access to a database z via oracle LR_{z}, invokes LR_{z} for every $i \in [n]$ with the randomizer R_g and outputs the average of the responses:

A LOCAL ALGORITHM $\mathcal{A}_g(n, \epsilon, LR_{\mathrm{z}})$ THAT SIMULATES AN SQ QUERY $g : D \to [-b, b]$

1. Output $\frac{1}{n}\sum_{i=1}^{n} LR_{\mathrm{z}}(i, R_g)$ where $R_g(u) = g(u) + \eta$ and $\eta \sim Lap\left(\frac{2b}{\epsilon}\right)$.

Note that $\mathcal{A}_g$ outputs $\left(\frac{1}{n}\sum_{i=1}^{n} g(z_i)\right) + \left(\frac{1}{n}\sum_{i=1}^{n} \eta_i\right)$, where the η_i are i.i.d. from $\mathrm{Lap}\left(\frac{2b}{\epsilon}\right)$. This algorithm is ϵ-local (since it applies a single ϵ-local randomized to each entry of z), and therefore ϵ-differentially private. The following lemma shows that when the input database z is large enough, $\mathcal{A}_g$ simulates the desired SQ query g with small error probability.

Lemma 5.6. *If, for sufficiently large constant c, database z has $n \geq c \cdot \frac{\log(1/\beta)b^2}{\epsilon^2\tau^2}$ entries sampled i.i.d. from a distribution $\mathcal{D}$ on D then algorithm $\mathcal{A}_g$ approximates $\mathbb{E}_{u\sim\mathcal{D}}[g(u)]$ within additive error $\pm\tau$ with probability at least $1 - \beta$.*

Proof. Let $v = \mathbb{E}_{u\sim\mathcal{D}}[g(u)]$ denote the true mean. By the Chernoff-Hoeffding bound for real-valued variables (Theorem A.2),

$$\Pr\left[\left|\frac{1}{n}\sum_{i=1}^{n} g(u_i) - v\right| \geq \frac{\tau}{2}\right] \leq 2\exp\left(-\frac{\tau^2 n}{8b^2}\right).$$

Therefore, in the absence of additive Laplacian random noise, $O\left(\frac{\ln(1/\beta)b^2}{\tau^2}\right)$ examples are enough to approximate $\mathbb{E}_{u\sim\mathcal{D}}[g(u)]$ within additive error $\pm\frac{\tau}{2}$ with probability at least $1 - \frac{\beta}{2}$. (Note that the number of examples is smaller than the lower bound on n in the lemma by a factor of $O(\epsilon^{-2})$).

The effect of the Laplace noise can also be bounded via a standard tail inequality: setting $\lambda = \frac{2b}{\epsilon}$ in Lemma A.3, we get that $O\left(\frac{\ln(1/\beta)b^2}{\epsilon^2\tau^2}\right)$ samples are sufficient to ensure that the average of η_i's lies outside $[-\frac{\tau}{2}, \frac{\tau}{2}]$ with probability at most $\frac{\beta}{2}$. It follows that $\mathcal{A}_g$ estimates $\mathbb{E}_{u\sim\mathcal{D}}[g(u)]$ within additive error $\pm\tau$ with probability at least $1 - \beta$. □

Simulation. Lemma 5.6 suggests a simple simulation of a nonadaptive (resp. adaptive) SQ algorithm by a noninteractive (resp. interactive) local algorithm as follows. Assume the SQ algorithm makes at most t queries to an SQ oracle $SQ_{\mathcal{D}}$. The local algorithm simulates each query (g, τ) by running $\mathcal{A}_g(n', \epsilon, LR_{\mathrm{z}})$ with parameters $\beta' = \frac{\beta}{t}$ and $n' = c \cdot \frac{\log(1/\beta')b^2}{\epsilon^2\tau^2}$ on a previously unused portion of the database z containing n' entries.

Theorem 5.7 (Local simulation of SQ)**.** *Let $\mathcal{A}_{SQ}$ be an SQ algorithm that makes at most t queries to an SQ oracle $SQ_{\mathcal{D}}$, each with tolerance at least τ. The simulation above is ϵ-differentially private. If, for sufficiently large constant c, database z has $n \geq c \cdot \frac{t\log(t/\beta)b^2}{\epsilon^2\tau^2}$ entries sampled i.i.d. from the distribution $\mathcal{D}$ then the simulation above gives the same output as $\mathcal{A}_{SQ}$ with probability at least $1 - \beta$.*

Furthermore, the simulation is noninteractive if the original SQ algorithm $\mathcal{A}_{SQ}$ is nonadaptive. The simulation is efficient if $\mathcal{A}_{SQ}$ is efficient.

Proof. Each query is simulated with a fresh portion of z, and hence privacy is preserved as each entry is subjected to a single application of the ϵ-local randomizer R. By the union bound, the probability of any of the queries not being approximated within additive error τ is bounded by β. If $\mathcal{A}_{\mathrm{SQ}}$ is nonadaptive, all queries to LR_{z} can be prepared in advance. □

5.1.2 Simulation of Local Algorithms by SQ Algorithms

Let z be a database containing n entries drawn i.i.d. from $\mathcal{D}$. Consider a local algorithm making t queries to LR_{z}. We show how to simulate any local randomizer invoked by this algorithm by using statistical queries to $SQ_{\mathcal{D}}$. Consider one such randomizer $R : D \to W$ applied to database entry z_i. To simulate R we need to sample $w \in W$ with probability $p(w) = \Pr_{z_i \sim \mathcal{D}}[R(z_i) = w]$ taken over choice of $z_i \sim \mathcal{D}$ and random coins of R. (For interactive algorithms, it is more complicated, as the outputs of different randomizers applied to the same entry z_i have to be correlated.)

A brief outline. The idea behind the simulation is to sample from a distribution $\widetilde{p}(\cdot)$ that is within small statistical distance of $p(\cdot)$. We start by applying R to an arbitrary input (say, $\mathbf{0}$) in the domain D and obtaining a sample $w \sim R(\mathbf{0})$. Let $q(w) = \Pr[R(\mathbf{0}) = w]$ (where the probability is taken only over randomness in R). Since R is ϵ-differentially private, $q(w)$ approximates $p(w)$ within a multiplicative factor of e^{ϵ}. To sample w from $p(\cdot)$ we use the following rejection sampling algorithm: (i) sample w according to $q(\cdot)$; (ii) with probability $\frac{p(w)}{q(w)e^{\epsilon}}$, output w; (iii) with the remaining probability, repeat from (i).

To carry out this strategy, we must be able to estimate $p(w)$, which depends on the (unknown) distribution $\mathcal{D}$, using only SQ queries. The rough idea is to express $p(w)$ as the expectation, taken over $z \sim \mathcal{D}$, of the function $h(z) = \Pr[R(z) = w]$ (where the probability is taken only over the coins of R). We can use h as the basis of an SQ query. In fact, to get a sufficiently accurate approximation, we must rescale the function h somewhat, and keep careful track of the error introduced by the SQ oracle. We present the details in the proof of the following lemma:

Lemma 5.8. *Let z be a database with entries drawn i.i.d. from a distribution $\mathcal{D}$. For every noninteractive (resp. interactive) local algorithm $\mathcal{A}$ making t queries to LR_{z}, there exists a nonadaptive (resp. adaptive) statistical query algorithm $\mathcal{B}$ that in expectation makes $O(t \cdot e^{\epsilon})$ queries to $SQ_{\mathcal{D}}$ with accuracy $\tau = \Theta(\beta/(e^{2\epsilon}t))$, such that the statistical difference between $\mathcal{B}$'s and $\mathcal{A}$'s output distributions is at most β.*

Proof. We split the simulation over Claims 5.9 and 5.10. In the first claim we simulate noninteractive local algorithms using nonadaptive SQ algorithms. In the second claim we simulate interactive local algorithms using adaptive SQ algorithms.

Claim 5.9. *For every noninteractive local algorithm $\mathcal{A}$ making t nonadaptive queries to LR_z, there exists a nonadaptive statistical query algorithm $\mathcal{B}$ that in expectation makes $t \cdot e^{\epsilon}$ queries to $SQ_{\mathcal{D}}$ with accuracy $\tau = \Theta(\beta/(e^{2\epsilon}t))$, such that the statistical difference between $\mathcal{B}$'s and $\mathcal{A}$'s output distributions is at most β.*

Proof. We show how to simulate an ϵ-local randomizer R using statistical queries to $SQ_{\mathcal{D}}$. Because the local algorithm is non-interactive, we can assume without loss of generality that it accesses each entry z_i only once. (Otherwise, one can combine different operators, used to access z_i, by combining their answers into a vector). Given $R : D \to W$, we want to sample $w \in W$ with probability:

$$p(w) = \Pr_{z_i \sim \mathcal{D}}[R(z_i) = w].$$

Two notes regarding our notation: (i) As z_i is drawn i.i.d. from $\mathcal{D}$ we could omit the index i. We leave the index i in our notation to emphasize that we actually simulate the application of a local randomizer R to entry i. (ii) The semantics of $\Pr$ changes depending on whether it appears with the subscript $z_i \sim \mathcal{D}$ or not. $\Pr_{z_i \sim \mathcal{D}}$ denotes probability that is taken over the choice of $z_i \sim \mathcal{D}$ and the randomness in R, whereas when the subscript is dropped z_i is fixed and the probability is taken only over the randomness in R. Using this notation, $\Pr_{z_i \sim \mathcal{D}}[R(z_i) = w] = \mathbb{E}_{z_i \sim \mathcal{D}} \Pr[R(z_i) = w]$.

We construct an algorithm $\mathcal{B}_{R,\epsilon}$ that given t, β, and access to the SQ oracle, outputs $w \in W$, such that the statistical difference between the output probability distributions of $\mathcal{B}_{R,\epsilon}$ and the simulated randomizer R is at most β/t. Because the local algorithm makes t queries, the overall statistical distance between the output distribution of the local algorithm and the distribution resulting from the simulation is at most β, as desired.

AN SQ ALGORITHM $\mathcal{B}_{R,\epsilon}(t, \beta, SQ_{\mathcal{D}})$ THAT SIMULATES AN ϵ-LOCAL RANDOMIZER $R : D \to W$.

1. Sample $w \sim R(\mathbf{0})$. Let $q(w) = \Pr[R(\mathbf{0}) = w]$.
2. Define $g : D \to [-1, 1]$ by $g(z_i) = \dfrac{\Pr[R(z_i) = w] - q(w)}{q(w)(e^{\epsilon} - e^{-\epsilon})}$, and let $\tau = \frac{\beta}{3e^{2\epsilon}t}$.
3. Query the SQ oracle $v = SQ_{\mathcal{D}}(g, \tau)$, and let $\widetilde{p}(w) = vq(w)(e^{\epsilon} - e^{-\epsilon}) + q(w)$.
4. With probability $\frac{\widetilde{p}(w)}{q(w)(1+\frac{\beta}{3t})e^{\epsilon}}$, output w.
   With the remaining probability, repeat from Step 1.

We now show that the statistical distance between the output of $\mathcal{B}_{R,\epsilon}(t, \beta, SQ_{\mathcal{D}})$ and the distribution $p(\cdot)$ is at most β/t. As mentioned above, our initial approximation $\widetilde{p}(\cdot)$ of $p(\cdot)$ in Step 1 is obtained by applying R to some arbitrary input (namely, $\mathbf{0}$) in the domain D and sampling $w \sim R(\mathbf{0})$. Since R is ϵ-differentially private, $q(w) = \Pr[R(\mathbf{0}) = w]$ approximates $p(w)$ within a multiplicative factor of e^{ϵ}.

However, to carry out the rejection sampling strategy, we need to get a much better estimate of $p(w)$. Steps 2 and 3 compute such an estimate, $\widetilde{p}(w)$, satisfying (with probability 1)

$$\widetilde{p}(w) \in (1 \pm \phi)\, p(w) \quad \text{where} \quad \phi = \tfrac{\beta}{3t}\,. \tag{5}$$

We establish the inclusion (5) below. For now, assume it holds on every iteration. Step 4 is a rejection sampling step which ensures that the output will follow a distribution close to $\widetilde{p}(\cdot)$. Inclusion (5) guarantees that $\frac{\widetilde{p}(w)}{q(w)(1+\frac{\beta}{3t})e^{\epsilon}}$ is at most 1, so the probability in Step 4 is well defined. The difficulty is that the quantity

$\widetilde{p}(w)$ is not a well-defined function of w: it depends on the SQ oracle and may vary, for the same w, from iteration to iteration.

Nevertheless, $\widetilde{p}$ is fixed for any given iteration of the algorithm. In the given iteration, any particular element w gets output with probability $q(w) \times \frac{\widetilde{p}(w)}{q(w)(1+\phi)e^{\epsilon}} = \frac{\widetilde{p}(w)}{(1+\phi)e^{\epsilon}}$. The probability that the given iteration terminates (i.e., outputs some w) is then $p_{terminate} = \sum_w \frac{\widetilde{p}(w)}{(1+\phi)e^{\epsilon}}$. By (5), this probability is in $\frac{1\pm\phi}{(1+\phi)e^{\epsilon}}$. Thus, *conditioned on the iteration terminating*, element w is output with probability $\frac{\widetilde{p}(w)}{(1+\phi)\cdot e^{\epsilon}\cdot p_{teminate}} \in \frac{1\pm\phi}{1\pm\phi} \cdot p(w)$. Since $\phi \leq 1/3$, we can simplify this to get

$$\Pr\left[w \text{ output in a given iteration} \middle| \text{iteration produces output}\right] \in (1 \pm 3\phi)p(w)\,.$$

This implies that no matter which iteration produces output, the statistical difference between the distribution of w and $p(\cdot)$ will be at most $3\phi = \frac{\beta}{t}$, as desired.

Moreover, since each iteration terminates with probability at least $\frac{1-\phi}{1+\phi} \cdot e^{-\epsilon}$, the expected number of iterations is at most $\frac{1+\phi}{1-\phi} \cdot e^{\epsilon} \leq 2e^{\epsilon}$. Thus, the total expected SQ query complexity of the simulation is $O(t \cdot e^{\epsilon})$.

It remains to prove the correctness of (5). To estimate $p(w)$ given w, we set up the statistical query $g(z_i)$. This is a valid query since $\Pr[R(z_i) = w]$ is a function of z_i, and furthermore $g(z_i) \in [-1, 1]$ for all z_i as $\Pr[R(z_i) = w]/\Pr[R(\mathbf{0}) = w] \in e^{\pm\epsilon}$. The SQ query result v lies within $\mathbb{E}_{z_i\sim\mathcal{D}}[g(z_i)] \pm \tau$, where τ is the tolerance parameter for the statistical query, and so

$$\mathop{\mathbb{E}}_{z_i\sim\mathcal{D}}[g(z_i)] = \frac{\mathbb{E}_{z_i\sim\mathcal{D}}\Pr[R(z_i) = w] - q(w)}{q(w)(e^{\epsilon} - e^{-\epsilon})} = \frac{p(w) - q(w)}{q(w)(e^{\epsilon} - e^{-\epsilon})}.$$

Plugging in the bounds for v and $q(w)$ we get that $\widetilde{p}(w) \in (1 \pm \tau')p(w)$ where $\tau' = e^{2\epsilon}\tau = \frac{\beta}{3t}$. This establishes (5) and concludes the proof. □

Claim 5.10. *For every interactive local algorithm $\mathcal{A}$ making t queries to LR_z, there exists an adaptive statistical query algorithm $\mathcal{B}$ that in expectation makes $O(t \cdot e^{\epsilon})$ queries $SQ_{\mathcal{D}}$ with accuracy $\tau = \Theta(\beta/(e^{2\epsilon}t))$, such that the statistical difference between $\mathcal{B}$'s and $\mathcal{A}$'s output distributions is at most β.*

Proof. As in the previous claim, we show how to simulate the output of the local randomizers during the run of the local algorithm. A difference, however, is that because an entry z_i may be accessed multiple times, we have to condition our sampling on the outcomes of previous (simulated) applications of local randomizers to z_i.

More concretely, let $R_1, R_2, ...$ be the sequence of randomizers that access the entry z_i. To simulate $R_k(z_i)$, we must take into account the answers $a_1, \ldots, a_{k-1}$ given by the simulations of $R_1(z_i), \ldots, R_{k-1}(z_i)$. We show how to do this using adaptive statistical queries to $SQ_{\mathcal{D}}$. The notation is the same as in Claim 5.9. We want to output $w \in W$ with probability

$$p(w) = \Pr_{z_i\sim\mathcal{D}}[R_k(z_i) = w \,|\, R_{k-1}(z_i) = a_{k-1}, R_{k-2}(z_i) = a_{k-2}, \ldots, R_1(z_i) = a_1],$$

where R_j $(1 \leq j \leq k-1)$ denotes the jth randomizer applied to z_i.

As before, we start by sampling $w \sim R(\mathbf{0})$. Let $q(w) = \Pr[R_k(\mathbf{0}) = w]$. Note that $q(w)$ approximates $p(w)$ within a multiplicative factor of e^{ϵ} because $R_1, \ldots, R_k$ are respectively ϵ_1-,$\ldots$, ϵ_k-differentially

private, and $\epsilon_1 + \ldots + \epsilon_k \leq \epsilon$. Hence, we can use the rejection sampling algorithm as in Claim 5.9. Rewrite $p(w)$:

$$\begin{aligned} p(w) &= \frac{\Pr_{z_i \sim \mathcal{D}}[R_k(z_i) = w \wedge R_{k-1}(z_i) = a_{k-1} \wedge \cdots \wedge R_1(z_i) = a_1]}{\Pr_{z_i \sim \mathcal{D}}[R_{k-1}(z_i) = a_{k-1} \wedge \cdots \wedge R_1(z_i) = a_1]} \\ &= \frac{\mathbb{E}_{z_i \sim \mathcal{D}}[\Pr[R_k(z_i) = w \wedge R_{k-1}(z_i) = a_{k-1} \wedge \cdots \wedge R_1(z_i) = a_1]]}{\mathbb{E}_{z_i \sim \mathcal{D}}[\Pr[R_{k-1}(z_i) = a_{k-1} \wedge \cdots \wedge R_1(z_i) = a_1]]} \end{aligned}$$

Conditioned on a particular value of z_i, the probabilities in the last expression depend only the coins of the randomizers. The outputs of the randomizers are independent conditioned on z_i, and therefore we can simplify the expression above:

$$p(w) = \frac{\mathbb{E}_{z_i \sim \mathcal{D}}\left[\Pr[R_k(z_i) = w] \cdot \prod_{j=1}^{k-1} \Pr[R_j(z_i) = a_j]\right]}{\mathbb{E}_{z_i \sim \mathcal{D}}\left[\prod_{j=1}^{k-1} \Pr[R_j(z_i) = a_j]\right]}$$

Let p_1 and p_2 denote the numerator and denominator, respectively, in the right hand side of the equation above. Let $r_1(z_i)$ and $r_2(z_i)$ denote the values inside the expectations that define p_1 and p_2, respectively. Namely,

$$r_1(z_i) = \Pr[R_k(z_i) = w] \cdot \prod_{j=1}^{k-1} \Pr[R_j(z_i) = a_j] \quad \text{and} \quad r_2(z_i) = \prod_{j=1}^{k-1} \Pr[R_j(z_i) = a_j]\,.$$

For estimating $p_1 = \mathbb{E}_{z_i \sim \mathcal{D}}[r_1(z_i)]$ we use the statistical query $g_1(z_i)$, and for estimating $p_2 = \mathbb{E}_{z_i \sim \mathcal{D}}[r_2(z_i)]$ we use the statistical query $g_2(z_i)$ defined as follows:

$$g_1(z_i) = \frac{r_1(z_i) - r_1(\mathbf{0})}{r_1(\mathbf{0})(e^\epsilon - e^{-\epsilon})} \quad \text{and} \quad g_2(z_i) = \frac{r_2(z_i) - r_2(\mathbf{0})}{r_2(\mathbf{0})(e^\epsilon - e^{-\epsilon})}.$$

As in Claim 5.9, one can estimate p_1 and p_2 to within a multiplicative factor of $(1 \pm \tau')$ where $\tau' = e^{2\epsilon}\tau$ and τ is the accuracy of the statistical queries. The ratio of the estimates for p_1 and p_2 gives an estimate $\tilde{p}(w)$ for $p(w)$ to within a multiplicative factor $(1 \pm 3\tau')$, for $\tau' \leq \frac{1}{3}$. The estimate $\tilde{p}(w)$ can then be used with rejection sampling to sample an output of the randomizer.

Let t be the number of queries made by $\mathcal{A}$. Setting $\tau' \leq \frac{\beta}{3t}$ guarantees that the statistical difference between distributions p and $\widetilde{p}$ is at most $\frac{\beta}{t}$, and hence the statistical difference between $\mathcal{B}$'s and $\mathcal{A}$'s output distributions is at most β. As in Claim 5.9, the expected number of SQ queries for rejection sampling is $O(t \cdot e^\epsilon)$. □

Claims 5.9 and 5.10 imply Lemma 5.8. □

Note that the efficiency of the constructions in Lemma 5.8 depends on the efficiency of computing the functions submitted to the SQ oracle, *e.g.,* the efficiency of computing the probability $\Pr[R(z_i) = w]$. We discuss this issue in the next section.

5.2 Implications for Local Learning

In this section, we define learning in the local and SQ models. The equivalence of the two models follows from the simulations described in the previous sections. An immediate but important corollary is that local learners are strictly less powerful than general private learners.

Definition 5.11 (Local Learning)**.** Locally learnable *is defined identically to privately PAC learnable (Definition 3.1), except for the additional requirement that for all $\epsilon > 0$, algorithm $\mathcal{A}(\epsilon, \cdot, \cdot, \cdot)$ is ϵ-local and invokes LR_{z} at most $poly(d, size(c), 1/\epsilon, 1/\alpha, \log(1/\beta))$ times. Class $\mathcal{C}$ is efficiently locally learnable if both: (i) the running time of $\mathcal{A}$ and (ii) the time to evaluate each query that $\mathcal{A}$ makes are bounded by some polynomial in $d, size(c), 1/\epsilon, 1/\alpha$, and* $\log(1/\beta)$.

Let $\mathcal{X}$ be a distribution over an input domain X. Let $SQ_{c,\mathcal{X}}$ denote the statistical query oracle that takes as input a function $g : X \times \{+1, -1\} \to \{+1, -1\}$ and a tolerance parameter $\tau \in (0, 1)$ and outputs v such that: $|v - \mathbb{E}_{x\sim\mathcal{X}}[g(x, c(x))]| \leq \tau$.

Definition 5.12 (SQ Learning[6])**.** SQ learnable *is defined identically to PAC learnable (Definition 2.4), except that instead of having access to examples z, an SQ learner $\mathcal{A}$ can make $poly(d, size(c), 1/\alpha, \log(1/\beta))$ queries to oracle $SQ_{c,\mathcal{X}}$ with tolerance $\tau \geq 1/poly(d, size(c), 1/\alpha, \log(1/\beta))$. Class $\mathcal{C}$ is efficiently SQ learnable if both: (i) the running time of $\mathcal{A}$ and (ii) the time to evaluate each query that $\mathcal{A}$ makes are bounded by some polynomial in $d, 1/\alpha$, and* $\log(1/\beta)$.

In order to state the equivalence between SQ and local learning, we require the following efficiency condition for a local randomizer.

Definition 5.13 (Transparent Local Randomizer)**.** *Let $R : D \to W$ be an ϵ-local randomizer. The randomizer is* transparent *if both: (i) for all inputs $u \in D$, the time needed to evaluate R; and (ii) for all inputs $u \in D$ and outputs $w \in W$ the time taken to compute the probability $\Pr[R(u) = w]$, are polynomially bounded in the size of the input and $1/\epsilon$.*

As stated, this definition requires *exact* computation of probabilities. This may not make sense on a finite-precision machine, since for many natural randomizers the transition probabilities are irrational. One can relax the requirement to insist that relevant probabilities are computable with additive error at most ϕ in time polynomial in $\log(\frac{1}{\phi})$.

All local protocols that have appeared in the literature [29, 3, 2, 1, 29, 45, 36] are transparent, at least in this relaxed sense.

In the equivalences of the previous sections, *transparency* of local randomizers corresponds directly to *efficient computability* of the function g in an SQ query. To see why, consider first the simulation of SQ algorithms by local algorithms: if the original SQ algorithm is efficient (that is, query g can be evaluated in polynomial time) then the local randomizer $R(u) = g(u) + \eta$ can also be evaluated in polynomial time for all $u \in D$. Furthermore, it is simple to estimate for all inputs $u \in D$ and outputs $w \in W$ the probability $\Pr[R(u) = w]$ since $R(u)$ is a Laplacian random variable with known parameters. Second, in the SQ simulation of a local algorithm, the functions $g(z_i) = \frac{\Pr[R(z_i)=w]-q(w)}{q(w)(e^{\epsilon}-e^{-\epsilon})}$ that are constructed can be evaluated efficiently precisely when the local randomizers are transparent.

We can now state the main result of this section, which follows from Lemmas 5.6 and 5.8, along with the correspondence between transparent randomizers and efficient SQ queries.

Theorem 5.14. *Let $\mathcal{C}$ be a concept class over X. Let $\mathcal{X}$ be a distribution over X. Let $z = (z_1, \ldots, z_n)$ denote a database where every $z_i = (x_i, c(x_i))$ with x_i drawn i.i.d. from $\mathcal{X}$ and $c \in \mathcal{C}$. Concept class $\mathcal{C}$ is*

[6]The standard definition of SQ learning does not allow for any probability of error in the learning algorithm (that is, $\beta = 0$). Our definition allows for a small failure probability β. This enables cleaner equivalence statements and clean modeling of randomized SQ algorithms. One can show that differentially private algorithms must have some non-zero probability of error, so a relaxation along these lines is necessary for our results.

locally learnable using $\mathcal{H}$ by an interactive local learner with inputs α, β, and with access to LR_z if and only if *$\mathcal{C}$ is SQ learnable using $\mathcal{H}$ by an adaptive SQ learner with inputs α, β, and access to $SQ_{c,\mathcal{X}}$.*

Furthermore, the simulations guarantee the following additional properties: (i) an efficient *SQ learner is simulatable by an* efficient *local learner that uses only* transparent *randomizers; (ii) an* efficient *local learner that uses only* transparent *randomizers is simulatable by an* efficient *SQ learner; (iii) a* nonadaptive *SQ (resp. noninteractive local) learner is simulatable by a* noninteractive *local (resp. nonadaptive SQ) learner.*

Now we can use lower bounds for SQ learners for PARITY (see, *e.g.,* [39, 12, 55]) to demonstrate limitations of local learners. The lower bound of [12] rules out SQ learners for PARITY that use at most $2^{d/3}$ queries of tolerance at least $2^{-d/3}$, even (a) allowing for unlimited computing time, (b) under the restriction that examples be drawn from the uniform distribution and (c) allowing a small probability of error (see Footnote 6). Since PARITY is (efficiently) privately learnable (Theorem 4.4), and since local learning is equivalent to SQ learning, we obtain:

Corollary 5.15. *Concept classes learnable by local learners are a strict subset of concept classes PAC learnable privately. This holds both with and without computational restrictions.*

5.3 The Power of Interaction in Local Protocols

To complete the picture of locally learnable concept classes, we consider how interaction changes the power of local learners (and, equivalently, how adaptivity changes SQ learning). As mentioned in the introduction, interaction is very costly in typical applications of local algorithms. We show that this cost is sometimes necessary, by giving a concept class that an interactive algorithm can learn efficiently with a polynomial number of examples drawn from the uniform distribution, but for which any noninteractive algorithm requires an exponential number of examples under the same distribution.

Let MASKED-PARITY be the class of functions $c_{r,a} : \{0,1\}^d \times \{0,1\}^{\log d} \times \{0,1\} \to \{+1,-1\}$ indexed by $r \in \{0,1\}^d$ and $a \in \{0,1\}$:

$$c_{r,a}(x,i,b) = \begin{cases} (-1)^{r \odot x + a} & \text{if } b = 0, \\ (-1)^{r_i} & \text{if } b = 1, \end{cases}$$

where $r \odot x$ denotes the inner product of r and x modulo 2, and r_i is the ith bit of r. This concept class divides the domain into two parts (according to the last bit, b). When $b = 0$, the concept $c_{r,a}$ behaves either like the PARITY concept indexed by r, or like its negation, according to the bit a (the "mask"). When $b = 1$, the concept essentially ignores the input example and outputs some bit of the parity vector r.

Below, we consider the learnability of MASKED-PARITY $= \{c_{r,a}\}$ when the examples are drawn from the uniform distribution over the domain $\{0,1\}^{d+\log d+1}$. In Section 5.3.1, we give a *adaptive* SQ learner for MASKED-PARITY under the uniform distribution. The adaptive learner uses two rounds of communication with the SQ oracle: the first, to learn r from the $b = 1$ half of the input, and the second, to retrieve the bit a from the $b = 0$ half of the input via queries that depend on r.

In Section 5.3.2, we show that no *nonadaptive* SQ learner which uses $2^{o(d)}$ examples can consistently produce a hypothesis that labels significantly more than $3/4$ of the domain correctly. The intuition is that as the queries are prepared nonadaptively, any information about r gained from the $b = 1$ half of the inputs cannot be used to prepare queries to the $b = 0$ half. Since information about a is contained only in the $b = 0$ half, in order to extract a, the SQ algorithm is forced to learn PARITY, which it cannot do with

few examples. Our separation in the SQ model directly translates to a separation in the local model (using Theorem 5.14).

The following theorem summarizes our results.

Theorem 5.16.

1. *There exists an efficient adaptive SQ learner for* MASKED-PARITY *over the uniform distribution.*
2. *No nonadaptive SQ learner can learn* MASKED-PARITY *(with a polynomial number of queries) even under the uniform distribution on examples. Specifically, there is an SQ oracle $\mathcal{O}$ such that any nonadaptive SQ learner that makes t queries to $\mathcal{O}$ over the uniform distribution, all with tolerance at least $2^{-d/3}$, satisfies the following: if the concept $c_{\bar{r},\bar{a}}$ is drawn uniformly at random from the set of* MASKED-PARITY *concepts, then, with probability at least $\frac{1}{2} - \frac{t}{2^{d/3+2}}$ over $c_{\bar{r},\bar{a}}$, the output hypothesis h of the learner has err$(c_{\bar{r},\bar{a}}, h) \geq \frac{1}{4}$.*

Corollary 5.17. *The concept classes learnable by nonadaptive SQ learners (resp. noninteractive local learners) under the uniform distribution are a strict subset of the concept classes learnable by adaptive SQ learners (resp. interactive local learners) under the uniform distribution. This holds both with and without computational restrictions.*

Weak vs. Strong Learning. The learning theory literature distinguishes between *strong* learning, in which the learning algorithm is required to produce hypotheses with arbitrarily low error (as in Definition 2.4, where the parameter α can be arbitrarily small), and *weak* learning, in which the learner is only required to produce a hypothesis with error bounded below $1/2$ by a polynomially small margin. The separation proved in this section (Theorem 5.16) applies only to *strong* learning: although no nonadaptive SQ learner can produce a hypothesis with error much better than $1/4$, it is simple to design a nonadaptive weak SQ learner for MASKED-PARITY under the uniform distribution with error exactly 1/4.

In fact, it is impossible to obtain an analogue of our separation for weak learning. The characterization of SQ learnable classes in terms of "SQ dimension" by Blum *et al.* [12] implies that adaptive and nonadaptive SQ algorithms are equivalent for weak learning. This is not explicit in [12], but follows from the fact that the weak learner constructed for classes with low SQ dimension is non-adaptive. (Roughly, the learner works by checking if the concept at hand is approximately equal to one of a polynomial number of alternatives; these alternatives depend on the input distribution and the concept class, but not on the particular concept at hand.)

Distribution-free vs Distribution-specific Learning The results of this section concern the learnability of MASKED-PARITY under the uniform distribution. The class MASKED-PARITY does not separate adaptive from nonadaptive *distribution-free* learners, since MASKED-PARITY cannot be learned by any SQ learner under the distribution which is uniform over examples with $b = 0$ (in that case, learning MASKED-PARITY is equivalent to learning PARITY under the uniform distribution). Separating adaptive from nonadaptive *distribution-free* SQ learning remains an open problem.

5.3.1 An Adaptive Strong SQ Learner for MASKED-PARITY over the Uniform Distribution

Our adaptive learner for MASKED-PARITY uses two rounds of communication with the SQ oracle: first, to learn r from the $b = 1$ half of the input, and second, to retrieve the bit a from the $b = 0$ half of the input via queries that depend on r. Theorem 5.16, part (1), follows from the proposition below.

ADAPTIVE SQ LEARNER $\mathcal{A}_{\mathsf{MP}}$ FOR MASKED-PARITY OVER THE UNIFORM DISTRIBUTION

1. For $j = 1, \ldots, d$ (in parallel)
   (a) Define $g_j : D \to \{0,1\}$ by
   $$g_j(x,i,b,y) = (i = j) \ \wedge \ (b = 1) \ \wedge \ (y = -1)\,,$$
   where $x \in \{0,1\}^d$, $i \in \{0,1\}^{\log d}$, $b \in \{0,1\}$, and $y = c_{r,a}(x,i,b) \in \{+1,-1\}$.
   (b) $answer_j \leftarrow SQ_{\mathcal{D}}(g_j, \tau)$, where $\tau = \frac{1}{4d+1}$, and $\hat{r}_j \leftarrow \begin{cases} 1 & \text{if } answer_j > \frac{1}{4d}; \\ 0 & \text{otherwise.} \end{cases}$
2. (a) $\hat{r} \leftarrow \hat{r_1} \ldots \hat{r_d} \in \{0,1\}^d$
   (b) Define $g_{d+1} : D \to \{0,1\}$ by
   $$g_{d+1}(x,i,b,y) = (b = 0) \ \wedge \ (y \neq (-1)^{\hat{r} \odot x})\,.$$
   where $x \in \{0,1\}^d$, $i \in \{0,1\}^{\log d}$, $b \in \{0,1\}$, and $y = c_{r,a}(x,i,b) \in \{+1,-1\}$.
   (c) $answer_{d+1} \leftarrow SQ_{\mathcal{D}}(g_{d+1}, \frac{1}{5})$., and $\hat{a} \leftarrow \begin{cases} 1 & \text{if } answer_{d+1} > \frac{1}{4}; \\ 0 & \text{otherwise.} \end{cases}$
   (d) Output $c_{\hat{r},\hat{a}}$.

Proposition 5.18 (Theorem 5.16, part (1), in detail)**.** *The algorithm* $\mathcal{A}_{\mathsf{MP}}$ efficiently *learns* MASKED-PARITY *(with probability 1) in 2 rounds using* $d+1$ *SQ queries computed over the uniform distribution with minimum tolerance* $\frac{1}{4d+1}$.

Proof. Consider the d queries in the first round. If $r_j = 1$, then

$$\mathop{\mathbb{E}}_{(x,i,b,y) \leftarrow \mathcal{D}}[g_j(x,i,b,y)] = \Pr_{i \in_u \{0,1\}^{\log d}, b \in_u \{0,1\}}[(i = j) \ \wedge \ (b = 1)] = \frac{1}{2d}\,.$$

If $r_j = 0$, then $\mathbb{E}[g_j(x,i,b,y)] = 0$. Since the tolerance τ is less than $\frac{1}{4d}$, each query g_j reveals the jth bit of r exactly. Thus, the estimate $\hat{r}_j$ is exactly r_j, and $\hat{r} = r$.

Given that $\hat{r}$ is correct, the second round query g_{d+1} is always 0 if $a = 0$. If $a = 1$, then g_{d+1} is 1 exactly when $b = 0$. Thus $\mathbb{E}[g_{d+1}(x,i,b,y)] = \frac{a}{2}$ (where $a \in \{0,1\}$). Since the tolerance is less than $\frac{1}{4}$, querying g_{d+1} reveals a: that is, $\hat{a} = a$, and so the algorithm outputs the target concept.

Note that the functions $g_1, \ldots, g_{d+1}$ are all computable in time $O(d)$, and the computations performed by $\mathcal{A}_{\mathsf{MP}}$ can be done in time $O(d)$, so the SQ learner is efficient. □

5.3.2 Impossibility of non-adaptive SQ learning for MASKED-PARITY

The impossibility result (Theorem 5.16, part (2)) for nonadaptive learners uses ideas from statistical query lower bounds (see, *e.g.,* [39, 12, 55]).

Proof of Theorem 5.16, part (2). Recall that the distribution $\mathcal{D}$ is uniform over $D = \{0,1\}^{d+\log(d)+1}$. For functions $f, h : \{0,1\}^{d+\log d+1} \to \{+1,-1\}$, recall that $err(f,h) = \Pr_{x \sim \mathcal{D}}[f(x) \neq h(x)]$. Define the inner

product of f and h as:

$$\langle f, h\rangle = \frac{1}{|D|}\sum_{x\in D} f(x)h(x) = \mathop{\mathbb{E}}_{x\sim\mathcal{D}}[f(x)h(x)].$$

The quantity $\langle f, h\rangle = \Pr_{x\sim\mathcal{D}}[f(x) = h(x)] - \Pr_{x\sim\mathcal{D}}[f(x) \neq h(x)] = 1 - 2\cdot \mathit{err}(f, h)$ measures the correlation between f and h when x is drawn from the uniform distribution $\mathcal{D}$.

Let the target function $c_{\bar{r},\bar{a}}$ be chosen uniformly at random from the set $\{c_{r,a}\}$. Consider a nonadaptive SQ algorithm that makes t queries $g_1, \ldots, g_t$. The queries $g_1, \ldots, g_t$ *must be independent of* $\bar{r}$ *and* $\bar{a}$ *since the learner is nonadaptive*. The only information about $\bar{a}$ is in the outputs associated with the $b = 0$ half of the inputs (recall that $c_{\bar{r},\bar{a}}(x, i, b) = (-1)^{r_i}$ when $b = 1$).

The main technical part of the proof follows the lower bound on SQ learning of PARITY. Using Fourier analysis, we split the true answer to a query into three components: a component that depends on the query g but not the pair $(\bar{r}, \bar{a})$, a component that depends on g and $\bar{r}$ (but not $\bar{a}$), and a component that depends on $g, \bar{r}$, and $\bar{a}$ (see Equation (7) below). We show that for most target concepts $c_{\bar{r},\bar{a}}$ the last component can be ignored by the SQ oracle. That is, a very close approximation to the correct output to the SQ queries made by the learner can be computed solely based on g and $\bar{r}$. Consequently, for most target concepts $c_{\bar{r},\bar{a}}$, the SQ oracle can return answers that are independent of $\bar{a}$, and hence $\bar{a}$ cannot be learned.

Consider a statistical query $g : \{0,1\}^d \times \{0,1\}^{\log d} \times \{0,1\} \times \{+1,-1\} \to \{+1,-1\}$. For some $(x, i, b) \in D$, the value of $g(x, i, b, \cdot)$ depends on the label (i.e., $(g(x, i, b, +1) \neq g(x, i, b, -1))$) and otherwise $g(x, i, b, \cdot)$ is insensitive to the label (i.e., $(g(x, i, b, +1) = g(x, i, b, -1))$). Every statistical query $g(\cdot, \cdot, \cdot, \cdot)$ can be decomposed into a label-independent and label-dependent part. This fact was first implicitly noted by Blum *et al.* [12] and made explicit by Bshouty and Feldman [17] (Lemma 30). We adapt the proof presented in [17] for our purpose.

Let

$$f_g(x, i, b) = \frac{g(x, i, b, 1) - g(x, i, b, -1)}{2} \qquad \text{and} \qquad C_g = \frac{1}{2}\,\mathbb{E}[g(x, i, b, 1) + g(x, i, b, -1)]\,.$$

We can rewrite the expectation of g on any concept $c_{\bar{r},\bar{a}}$ in terms of these quantities:

$$\mathbb{E}[g(x, i, b, c_{\bar{r},\bar{a}}(x, i, b))] = C_g + \langle f_g, c_{\bar{r},\bar{a}}\rangle\,.$$

Note that C_g depends on the statistical query g, but not on the target function. We now wish to analyze the second term, $\langle f_g, c_{\bar{r},\bar{a}}\rangle$, more precisely. To this end, we define the following functions parameterized by $s \in \{0, 1\}$:

$$c^s_{\bar{r},\bar{a}}(x, i, b) = \begin{cases} 0 & \text{if } b \neq s, \\ c_{\bar{r},\bar{a}}(x, i, b) & \text{if } b = s, \end{cases} \quad \text{and} \quad f^s_g(x, i, b) = \begin{cases} 0 & \text{if } b \neq s, \\ f_g(x, i, b) & \text{if } b = s. \end{cases} \tag{6}$$

Recall that $\langle f_g, c_{\bar{r},\bar{a}}\rangle$ is a sum over tuples (x, i, b). We can separate the sum into two pieces: one with tuples where $b = 0$ and the other with tuples where $b = 1$. Using the functions $c^s_{\bar{r},\bar{a}}, f^s_g$ just defined, we can write $\langle f_g, c_{\bar{r},\bar{a}}\rangle = \langle f^0_g, c^0_{\bar{r},\bar{a}}\rangle + \langle f^1_g, c^1_{\bar{r},\bar{a}}\rangle$. Hence,

$$\mathbb{E}[g(x, i, b, c_{\bar{r},\bar{a}}(x, i, b))] = C_g + \langle f^0_g, c^0_{\bar{r},\bar{a}}\rangle + \langle f^1_g, c^1_{\bar{r},\bar{a}}\rangle. \tag{7}$$

The inner product $\langle f^1_g, c^1_{\bar{r},\bar{a}}\rangle$ depends on the statistical query g and on $\bar{r}$, but not on $\bar{a}$. Thus only the middle term on the righthand side of (7) depends on $\bar{a}$.

Consider an SQ oracle $\mathcal{O} = \mathcal{O}_{c_{\bar{r},\bar{a}},\mathcal{D}}$ that responds to every query (g,τ) as follows (recall that $\mathcal{D}$ is the uniform distribution):

$$\mathcal{O}_{c_{\bar{r},\bar{a}},\mathcal{D}}(g,\tau) = \begin{cases} C_g + \langle f_g^1, c_{\bar{r},\bar{a}}^1 \rangle & \text{if } |\langle f_g^0, c_{\bar{r},\bar{a}}^0 \rangle| < \tau, \\ \mathbb{E}[g(x,i,b,c_{\bar{r},\bar{a}}(x,i,b))] & \text{otherwise.} \end{cases}$$

If the condition $|\langle f_g^0, c_{\bar{r},\bar{a}}^0 \rangle| < \tau$ is met for all the queries (g,τ) made by the learner, then the SQ oracle $\mathcal{O}$ never replies with a quantity that depends on $\bar{a}$. We now show that this is typically the case.

Extend the definition of $c_{\bar{r},\bar{a}}^s$ (Equation 6) to any $(r,a) \in \{0,1\}^d \times \{0,1\}$ by defining

$$c_{r,a}^0(x,i,b) = \begin{cases} 0 & \text{if } b = 1, \\ c_{r,a}(x,i,b) \left(= (-1)^{\langle r,x \rangle + a}\right) & \text{if } b = 0. \end{cases}$$

Note that for $r, r' \in \{0,1\}^d$ and $a \in \{0,1\}$,

$$\langle c_{r,a}^0, c_{r',a}^0 \rangle = \begin{cases} 1/2 & \text{if } r = r', \\ 0 & \text{if } r \neq r'. \end{cases}$$

We get that $\{c_{r,0}^0\}_{r \in \{0,1\}^d}$ is an orthogonal set of functions, and similarly with $\{c_{r,1}^0\}_{r \in \{0,1\}^d}$. The ℓ_2 norm of $c_{r,0}^0$ is $\|c_{r,0}^0\| = \sqrt{\langle c_{r,0}^0, c_{r,0}^0 \rangle} = 1/\sqrt{2}$, so the set $\{\sqrt{2} \cdot c_{r,0}^0\}_{r \in \{0,1\}^d}$ is orthonormal. A similar argument holds for $\{\sqrt{2} \cdot c_{r,1}^0\}_{r \in \{0,1\}^d}$.

Expanding the function f_g^0 in the orthonormal set $\{\sqrt{2} \cdot c_{r,0}^0\}_{r \in \{0,1\}^d}$, we get:

$$\sum_{r \in \{0,1\}^d} \langle f_g^0, \sqrt{2} \cdot c_{r,0}^0 \rangle^2 \leq \|f_g^0\|^2 = \langle f_g^0, f_g^0 \rangle \leq 1/2\,.$$

(The first inequality is loose in general because the set $\{\sqrt{2} \cdot c_{r,0}^0\}_{r \in \{0,1\}^d}$ spans a subset of dimension 2^d whereas f_g^0 is taken from a space of dimension $2^{d+\log d+1}$). Similarly,

$$\sum_{r \in \{0,1\}^d} \langle f_g^0, \sqrt{2} \cdot c_{r,1}^0 \rangle^2 \leq \|f_g^0\|^2 = \langle f_g^0, f_g^0 \rangle \leq 1/2.$$

Summing the two previous equations, we get

$$\sum_{(r,a) \in \{0,1\}^d \times \{0,1\}} 2 \cdot \langle f_g^0, c_{r,a}^0 \rangle^2 \leq 1\,.$$

Hence, at most $2^{2d/3-1}$ functions $c_{r,a}$ can have $|\langle f_g^0, c_{r,a}^0 \rangle| \geq 1/2^{d/3}$. Since $\bar{r}, \bar{a}$ was chosen uniformly at random we can restate this: for any particular query g, the probability that $c_{\bar{r},\bar{a}}^0$ has inner product more than $1/2^{d/3}$ with f_g^0 is at most $2^{2d/3-1}/2^{d+1} = 2^{-d/3}$. This is true regardless of a: since $c_{r,0}^0 = -c_{r,0}^0$, we have $|\langle f_g^0, c_{r,0}^0 \rangle| = |\langle f_g^0, c_{r,1}^0 \rangle|$, so the event that $|\langle f_g^0, c_{\bar{r},\bar{a}}^0 \rangle| \geq 1/2^{d/3}$ happens with probability at most $2^{-d/3}$ over $\bar{r}$, for $\bar{a} = 0, 1$.

Recall that the learner makes t queries, $g_1, \ldots, g_t$. Let $Good$ be the event that $|\langle f_{g_i}^0, c_{\bar{r},\bar{a}} \rangle| \leq 1/2^{d/3}$ for all $i \in [t]$ (i.e., the oracle can answer each of the queries independently of $\bar{a}$). Taking a union bound over queries, we have $\Pr[Good] \geq 1 - t/2^{d/3+2}$ (where the probability is taken only over $\bar{r}$).

We argued above that there is a valid SQ oracle which, conditioned on $Good$, can be simulated using $\bar{r}$ but without knowledge of $\bar{a}$, as long as all queries are made with tolerance $\tau \geq 1/2^{d/3}$ (as in

the theorem statement). To conclude the proof, we now argue that no nonadaptive *strong* learner exists for MASKED-PARITY over the uniform distribution. For that we concentrate on the $b = 0$ half of the inputs, where the outcome of $c_{\bar{r},\bar{a}}(\cdot)$ depends on a. Let h be the output hypothesis of the learner. For any input $(x, i, 0)$ we have $c_{\bar{r},0}(x, i, 0) = -c_{\bar{r},1}(x, i, 0)$. Thus either $c_{\bar{r},0}(x, i, 0) \neq h(x, i, 0)$ or $c_{\bar{r},1}(x, i, 0) \neq h(x, i, 0)$, and so some choice of $\bar{a}$ causes the error of h to be at least $1/4$.

Let A be the event that $err(h, c_{\bar{r},\bar{a}}) \geq 1/4$. Because $Good$ depends only on $\bar{r}$, we can think of $\bar{a}$ as being selected after the learner's hypothesis h whenever $Good$ occurs. Thus, $\Pr[A \mid Good] \geq 1/2$. Using $\overline{Good}$ to denote the complement of the event $Good$, we get

$$\begin{aligned} \Pr[A] &= \Pr[A \wedge Good] + \Pr[A \wedge \overline{Good}] \\ &\geq \Pr[A \mid Good] \Pr[Good] + 0 \geq \frac{1}{2}(1 - t/2^{d/3+2}). \end{aligned}$$

Therefore, $\Pr[err(h, c_{\bar{r},\bar{a}}) \geq 1/4] \geq \frac{1}{2}(1 - t/2^{d/3+2})$, as desired. □

Acknowledgments

We thank Enav Weinreb for many discussions related to the local model, Avrim Blum and Rocco Servedio for discussions about related work in learning theory, and Katrina Ligett and Aaron Roth for discussions about [14]. We also thank an anonymous reviewer for useful comments on the paper and, in particular, for the simple proof of Theorem 3.6.

References

[1] AGRAWAL, D., AND AGGARWAL, C. C. On the design and quantification of privacy preserving data mining algorithms. In *PODS* (2001), ACM, pp. 247–255.

[2] AGRAWAL, R., AND SRIKANT, R. Privacy-preserving data mining. In *SIGMOD* (2000), vol. 29(2), ACM, pp. 439–450.

[3] AGRAWAL, S., AND HARITSA, J. R. A framework for high-accuracy privacy-preserving mining. In *ICDE* (2005), IEEE Computer Society, pp. 193–204.

[4] ALEKHNOVICH, M. More on average case vs approximation complexity. In *FOCS* (2003), IEEE, pp. 298–307.

[5] AMBAINIS, A., JAKOBSSON, M., AND LIPMAA, H. Cryptographic randomized response techniques. In *PKC* (2004), vol. 2947 of *LNCS*, Springer, pp. 425–438.

[6] ANGLUIN, D., AND VALIANT, L. G. Fast probabilistic algorithms for hamiltonian circuits and matchings. *J. Comput. Syst. Sci. 18*, 2 (1979), 155–193.

[7] BARAK, B., CHAUDHURI, K., DWORK, C., KALE, S., MCSHERRY, F., AND TALWAR, K. Privacy, accuracy, and consistency too: a holistic solution to contingency table release. In *PODS* (2007), ACM, pp. 273–282.

[8] BEIMEL, A., KASIVISWANATHAN, S. P., AND NISSIM, K. Bounds on the sample complexity for private learning and private data release. In *Theory of Cryptography Conference (TCC)* (2010), D. Micciancio, Ed., LNCS, Springer.

[9] BEN-DAVID, S., PÁL, D., AND SIMON, H.-U. Stability of k-means clustering. In *COLT* (2007), LNCS, pp. 20–34.

[10] BEN-DAVID, S., VON LUXBURG, U., AND PÁL, D. A sober look at clustering stability. In *COLT* (2006), LNCS, Springer, pp. 5–19.

[11] BLUM, A., DWORK, C., MCSHERRY, F., AND NISSIM, K. Practical privacy: The SuLQ framework. In *PODS* (2005), ACM, pp. 128–138.

[12] BLUM, A., FURST, M. L., JACKSON, J., KEARNS, M. J., MANSOUR, Y., AND RUDICH, S. Weakly learning DNF and characterizing statistical query learning using Fourier analysis. In *STOC* (1994), ACM, pp. 253–262.

[13] BLUM, A., KALAI, A., AND WASSERMAN, H. Noise-tolerant learning, the parity problem, and the statistical query model. *J. ACM 50*, 4 (2003), 506–519.

[14] BLUM, A., LIGETT, K., AND ROTH, A. A learning theory approach to non-interactive database privacy. In *STOC* (2008), ACM, pp. 609–618.

[15] BLUMER, A., EHRENFEUCHT, A., HAUSSLER, D., AND WARMUTH, M. K. Occam's razor. *Inf. Process. Lett. 24*, 6 (1987), 377–380.

[16] BOUSQUET, O., AND ELISSEEFF, A. Stability and generalization. *Journal of Machine Learning Research 2* (2002), 499 – 526.

[17] BSHOUTY, N. H., AND FELDMAN, V. On using extended statistical queries to avoid membership queries. *Journal of Machine Learning Research 2* (2002), 359–395.

[18] CHERNOFF, H. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Ann. Math. Statist. 23* (1952), 493–507.

[19] DEVROYE, L., AND WAGNER, T. Distribution-free performance bounds for potential function rules. *IEEE Transactions on Information Theory 25*, 5 (1979), 601–604.

[20] DINUR, I., AND NISSIM, K. Revealing information while preserving privacy. In *PODS* (2003), ACM, pp. 202–210.

[21] DWORK, C. Differential privacy. In *ICALP* (2006), LNCS, pp. 1–12.

[22] DWORK, C., KENTHAPADI, K., MCSHERRY, F., MIRONOV, I., AND NAOR, M. Our data, ourselves: Privacy via distributed noise generation. In *EUROCRYPT* (2006), LNCS, Springer, pp. 486–503.

[23] DWORK, C., AND LEI, J. Differential privacy and robust statistics. In *Symposium on the Theory of Computing (STOC)* (2009).

[24] DWORK, C., MCSHERRY, F., NISSIM, K., AND SMITH, A. Calibrating noise to sensitivity in private data analysis. In *TCC* (2006), LNCS, Springer, pp. 265–284.

[25] DWORK, C., MCSHERRY, F., AND TALWAR, K. The price of privacy and the limits of lp decoding. In *STOC* (2007), ACM, pp. 85–94.

[26] DWORK, C., AND NISSIM, K. Privacy-preserving datamining on vertically partitioned databases. In *CRYPTO* (2004), LNCS, Springer, pp. 528–544.

[27] DWORK, C., AND YEKAHNIN, S. On lower bounds for noise in private analysis of statistical databases. Presentation at BSF/DIMACS/DyDan Workshop on Data Privacy, February 2008.

[28] ELISSEEFF, A., EVGENIOU, T., AND PONTIL, M. Stability of randomized learning algorithms. *Journal of Machine Learning Research 6* (2005), 55–79.

[29] EVFIMIEVSKI, A., GEHRKE, J., AND SRIKANT, R. Limiting privacy breaches in privacy preserving data mining. In *PODS* (2003), ACM, pp. 211–222.

[30] FISCHER, P., AND SIMON, H.-U. On learning ring-sum-expansions. *SIAM Journal on Computing 21*, 1 (1992), 181–192.

[31] FREUND, Y., MANSOUR, Y., AND SCHAPIRE, R. E. Generalization bounds for averaged classifiers. *Annals of Statistics 32*, 4 (2004), 1698–1722.

[32] HAUSSLER, D. Decision theoretic generalizations of the PAC model for neural net and other learning applications. *Information and Computation 100*, 1 (1992), 78–150.

[33] HELMBOLD, D., SLOAN, R., AND WARMUTH, M. K. Learning integer lattices. *SIAM Journal on Computing 21*, 2 (Apr. 1992), 240–266.

[34] HOEFFDING, W. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association 58*, 301 (1963), 13–30.

[35] HOPPER, N. J., AND BLUM, M. Secure human identification protocols. In *ASIACRYPT* (2001), vol. 2248 of *LNCS*, Springer, pp. 52–66.

[36] JANK, W., AND SHMUELI, G. *Statistical Methods in eCommerce Research*. Wiley & Sons, 2008.

[37] KASIVISWANATHAN, S. P., LEE, H. K., NISSIM, K., RASKHODNIKOVA, S., AND SMITH, A. What can we learn privately? In *FOCS* (2008), pp. 559–569.

[38] KASIVISWANATHAN, S. P., AND SMITH, A. A note on differential privacy: Defining resistance to arbitrary side information. *CoRR arXiv:0803.39461 [cs.CR]* (2008).

[39] KEARNS, M. Efficient noise-tolerant learning from statistical queries. *Journal of the ACM 45*, 6 (1998), 983–1006. Preliminary version in *proceedings of STOC'93*.

[40] KEARNS, M., AND RON, D. Algorithmic stability and sanity-check bounds for leave-one-out cross-validation. *Neural Computation 11*, 6 (1999), 1427 – 1453.

[41] KEARNS, M. J., SCHAPIRE, R. E., AND SELLIE, L. M. Toward efficient agnostic learning. *Machine Learning 17*, 2-3 (1994), 115–141.

[42] KEARNS, M. J., AND VAZIRANI, U. V. *An Introduction to Computational Learning Theory*. MIT press, Cambridge, Massachusetts, 1994.

[43] KUTIN, S., AND NIYOGI, P. Almost-everywhere algorithmic stability and generalization error. In *UAI* (2002), pp. 275–282.

[44] MCSHERRY, F., AND TALWAR, K. Mechanism design via differential privacy. In *FOCS* (2007), IEEE, pp. 94–103.

[45] MISHRA, N., AND SANDLER, M. Privacy via pseudorandom sketches. In *PODS* (2006), ACM, pp. 143–152.

[46] MORAN, T., AND NAOR, M. Polling with physical envelopes: A rigorous analysis of a human-centric protocol. In *EUROCRYPT* (2006), LNCS, Springer, pp. 88–108.

[47] NISSIM, K., RASKHODNIKOVA, S., AND SMITH, A. Smooth sensitivity and sampling in private data analysis. In *STOC* (2007), ACM, pp. 75–84.

[48] RASTOGI, V., HONG, S., AND SUCIU, D. The boundary between privacy and utility in data publishing. In *VLDB* (2007), pp. 531–542.

[49] REGEV, O. On lattices, learning with errors, random linear codes, and cryptography. In *STOC* (2005), pp. 84–93.

[50] SMITH, A. Efficient, differentially private point estimators. *CoRR abs/0809.4794* (2008).

[51] VALIANT, L. G. A theory of the learnable. *Communications of the ACM 27* (1984), 1134–1142.

[52] VAN DEN HOUT, A., AND VAN DER HEIJDEN, P. Randomized response, statistical disclosure control and misclassification: A review. *International Statistical Review 70* (2002), 269–288.

[53] WARNER, S. L. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association 60*, 309 (1965), 63–69.

[54] WASSERMAN, L., AND ZHOU, S. A statistical framework for differential privacy. *ArXiv.org*, arXiv:0811.2501v1 [math.ST] (2008).

[55] YANG, K. New lower bounds for statistical query learning. *Journal of Computer and System Sciences 70*, 4 (2005), 485–509.

[56] ZHOU, S., LIGETT, K., AND WASSERMAN, L. Differential privacy with compression. *ArXiv.org*, arXiv:0901.1365v1 [stat.ML] (2009).

A Concentration Bounds

We need several standard tail bounds in this paper.

Theorem A.1 (Multiplicative Chernoff Bounds (e.g. [18, 6]))**.** *Let $X_1, \ldots, X_n$ be i.i.d. Bernoulli random variables with $\Pr[X_i = 1] = \mu$. Then for every $\phi \in (0, 1]$,*

$$\Pr\left[\frac{\sum_i X_i}{n} \geq (1+\phi)\mu\right] \leq \exp\left(-\frac{\phi^2 \mu n}{3}\right)$$

and

$$Pr\left[\frac{\sum_i X_i}{n} \leq (1-\phi)\mu\right] \leq \exp\left(-\frac{\phi^2 \mu n}{2}\right).$$

Theorem A.2 (Real-valued Additive Chernoff-Hoeffding Bound [34]). *Let $X_1, \ldots, X_n$ be i.i.d. random variables with $\mathbb{E}[X_i] = \mu$ and $a \leq X_i \leq b$ for all i. Then for every $\delta > 0$,*

$$\Pr\left[\left|\frac{\sum_i X_i}{n} - \mu\right| \geq \delta\right] \leq 2\exp\left(\frac{-2\delta^2 n}{(b-a)^2}\right).$$

Lemma A.3 (Sums of Laplace Random Variables). *Let $X_1, ..., X_n$ be i.i.d. random variables drawn from $\mathrm{Lap}(\lambda)$ (i.e., with probability density $h(x) = \frac{1}{2\lambda}\exp\left(-\frac{|x|}{\lambda}\right)$). Then for every $\delta > 0$,*

$$\Pr\left[\left|\frac{\sum_{i=1}^n X_i}{n}\right| \geq \delta\right] = \exp\left(-\frac{\delta^2 n}{4\lambda^2}\right).$$

The proof of this lemma is standard; we include it here since we were unable to find an appropriate reference.

Proof. Let $S = \sum_{i=1}^n X_i$. By the Markov inequality, for all $t > 0$,

$$\Pr[S > \delta n] = \Pr[e^{tS} > e^{t\delta n}] \leq \frac{\mathbb{E}[e^{tS}]}{e^{t\delta n}} = \frac{m_S(t)}{e^{t\delta n}},$$

where $m_S(t) = \mathbb{E}[e^{tS}]$ is the moment generating function of S. To compute $m_S(t)$, note that the moment generating function of $X \sim \mathrm{Lap}(\lambda)$ is $m_X(t) = \mathbb{E}[e^{tX}] = \frac{1}{1-(\lambda t)^2}$, defined for $0 < t < \frac{1}{\lambda}$. Hence $m_S(t) = (m_X(t))^n = (1-(\lambda t)^2)^{-n} < \exp(n(\lambda t)^2)$, where the last inequality holds for $(\lambda t)^2 < \frac{1}{2}$. We get that $\Pr[S > \delta n] \leq \exp(n((\lambda t)^2 - t\delta))$. To complete the proof, set $t = \frac{\delta}{2}\lambda^2$ (note that if $\delta < 1$ and $\lambda > 1$ then $(\lambda t)^2 = (\frac{\delta}{2}\lambda)^2 < \frac{1}{2}$). We get that $\Pr[S > \delta n] \leq \exp\left(n\left(\left(\frac{\delta}{2}\lambda\right)^2 - \frac{\delta^2}{2}\lambda\right)\right) = \exp\left(-n\frac{\delta^2}{4}\lambda^2\right)$, as desired. □