Individuals and Privacy in the Eye of Data Analysis

Thesis submitted in partial fulfillment
of the requirements for the degree of
"DOCTOR OF PHILOSOPHY"

by

Uri Stemmer

Submitted to the Senate of
Ben-Gurion University of the Negev

Approved by

Prof. Amos Beimel	Prof. Kobbi Nissim	Prof. Michal Shapira
Thesis advisor	Thesis advisor	Dean

March 2017
Beer-Sheva

This work was carried out under the supervision of

Prof. Amos Beimel and Prof. Kobbi Nissim

In the Department of Computer Science

Faculty of Natural Sciences

Acknowledgments

I could not have asked for better advisors. I will be forever grateful for their close guidance, their constant encouragement, and the warm shelter they provided. Without them, this thesis could have never begun.

I have been fortunate to work with Raef Bassily, Amos Beimel, Mark Bun, Kobbi Nissim, Adam Smith, Thomas Steinke, Jonathan Ullman, and Salil Vadhan. I enjoyed working with them all, and I thank them for everything they have taught me.

Contents

List of Figures

Abstract

Many modern data analysts face the challenge of performing computations on sensitive individual data, while at the same time protecting the privacy of those individuals. In this thesis we will be interested in *differential privacy* [Dwork et al., TCC 2006], a mathematical definition for privacy providing rigorous guarantees. Our research theme is driven by the following question:

> **Question:** *When can we construct privacy preserving analogues to existing data analysis algorithms, and what price do we pay?*

We focus on the task of *private learning* [Kasiviswanathan et al., FOCS 2008], which generalizes many of the analyses applied to large collections of data. Informally, a private learner is applied to a database of labeled individual information and outputs a hypothesis while preserving the privacy of each individual. Utility wise, this hypothesis should be able to predict the labels corresponding to unseen individuals.

A natural measure for the price of privacy is the amount of data necessary in order to *privately* identify a good hypothesis – a.k.a. *the sample complexity*. The sample complexity of private learners is important, as it determines the amount of individual data that must be collected before starting the analysis. We show new results about the possibility and impossibility of learning tasks with differential privacy, and study the incurred price to the sample complexity. In particular:

- We present a combinatorial characterization of the sample complexity of private learners preserving *pure* ε-differential privacy, in terms of a new combinatorial measure we call *the representation dimension*.
- We compare the sample complexity of private learning under *pure* ε-differential privacy and its relaxation *approximate* (ε, δ)-differential privacy [Dwork et al., Eurocrypt 2006]. We show that the sample complexity of private learning under approximate

differential privacy can be significantly lower than that under pure differential privacy. To that end, we introduce a new tool for privately conducting binary searches.

- We also present new lower bounds separating (under some conditions) the sample complexity of approximate differentially private learners from that of non-private learners. These are the first non-trivial lower bounds on the sample complexity of approximate differentially private learners.

We also study the reverse connection between privacy and learning, namely:

> **Question:** *Can differential privacy be used as a tool to construct* new *(non-private) learning algorithms?*

Recall that learning algorithms are aimed at identifying properties of the underlying population, rather than properties of the given input data. Dwork et al. [STOC 2015] showed that if a property of the given sample is identified by a differentially private computation, then this property is in fact a property of the underlying population. This connection can be used to construct new algorithms, in settings which are not directly focused on privacy. Specifically, Dwork et al. used this connection for the task of answering *adaptively chosen* statistical queries on the underlying population using a sample, showing that if the answers are computed with (ε, δ)-differential privacy then $O(\varepsilon)$ accuracy is guaranteed with probability $1 - O(\delta^{\varepsilon})$. We greatly simplify and improve the results of Dwork et al. In particular:

- We show that (ε, δ)-differential privacy guarantees $O(\varepsilon)$ accuracy with probability $1 - O(\delta/\varepsilon)$. We show that our bound is tight.
- Due to its simplicity, our proof extends beyond statistical queries, to the task of answering a more general family of queries.

Keywords

Differential privacy, PAC learning, sample complexity, generalization.

Chapter 1

Introduction

Data analysis – the process of extracting knowledge from data – is an extremely beneficial technology, helping us improve upon nearly all aspects of life: Better medicine, better science, education, business intelligence, and much more. Knowledge is power. However, while the benefits of data analysis are rather self-evident, it is not without risks, and those are often overlooked. The risks come from the fact that a lot of the analyzed data contains private personal information, which, if revealed, might lead to embarrassment, loss of social status, job loss, monetary loss, or even mortal danger in some cases.

Privacy concerns exist wherever personal information is involved, such as healthcare records, financial records, web surfing behavior, and location-based services. At a high level, we can identify *two* sources for a potential privacy breach. The first, and more obvious, is *security*: If the (sensitive) data is leaked, stolen, or hacked, then privacy is clearly violated. The second (less obvious) source is *information released about the data*: At times, organizations and government agencies might want to publicly release (anonymized) personal data, in the name of science, in order to enable valuable research to be undertaken. In what sense should the data be anonymized? Is our privacy protected?

> **Question 1:** *How can we extract useful information from personal individual data, while* ensuring *the privacy of those individuals?*

To further illustrate this point, let us imagine a health insurance company, in which premiums are (partially) based on the medical status of insured family members (whose data are available to the company). Indeed, if your family members have suffered a serious medical condition, then there is, unfortunately, a greater chance that you will suffer the same condition. Now assume that both you and your cousin are insured by this company, and suppose that, out of the blue, the company bumps up your premiums. What does this tell you about your cousin?

In this thesis we study systems whose privacy guarantees are *mathematically proven*. Such systems are provably resilient to any kind of (known or unknown) attacks. To that end we must have a clear *mathematical definition* of *privacy*, as otherwise we could not claim that privacy is preserved. The definition that will perform a leading role throughout this thesis is that of *differential privacy* [44], a rigorous definition placing private data analysis on firm foundations.

1.1 Differential Privacy

Consider a database S consisting of n rows from a data universe X, where each row holds the information of one individual. We would like to analyze this database while ensuring the privacy of each individual. Specifically, we would like to apply some data analysis procedure $\mathcal{M}$ onto the database, and to obtain an outcome y such that: (i) the outcome y is useful for the analysis task at hand; and (ii) *privacy* is preserved for each individual whose data is in the database, even if y is publicly released. Under *differential privacy*, we say that privacy is preserved if the outcome y does not reveal information that is specific to any individual in the database. More formally, differential privacy requires that no individual's data has a significant effect on the distribution of the output y. Intuitively, this guarantees that whatever is learned about an individual could also be learned with her data arbitrarily modified (or without her data).

Definition 1.1 ([44, 38, 42])**.** *A randomized algorithm $\mathcal{M} : X^n \to Y$ is* (ε, δ) differentially private *if for every two databases $S, S' \in X^n$ that differ on one row, and every set $T \subseteq Y$, we have*

$$\Pr[\mathcal{M}(S) \in T] \leq e^{\varepsilon} \cdot \Pr[\mathcal{M}(S') \in T] + \delta.$$

Observe that differential privacy is a property of the *process*, rather than its outcome. The original definition from [44] had $\delta = 0$, and is sometimes referred to as *pure* differential privacy. However, a number of subsequent works have shown that allowing a small (but negligible) value of δ, referred to as *approximate differential privacy* [42], can provide substantial accuracy gains over pure differential privacy [43, 60, 48, 33, 11].

Differential privacy emerged from a line of work, called *private data analysis* [35, 46,

15, 38, 44, 42], that has placed data privacy on firm theoretical foundations. This line of research was initiated by Dinur, Dwork, and Nissim [35, 46], who demonstrated that a mathematically rigorous treatment for privacy is possible. They considered a setting in which a trusted data curator provides noisy answers to queries given by data analysts, and proved, along positive results, that a substantial amount of noise must be added in order to avoid a privacy breach. Indeed, privacy is achieved in differentially private algorithms through randomization and the introduction of noise to obscure the effect of each individual, and thus differentially private algorithms can be less accurate than their non-private analogues. Nevertheless, by now a rich literature has shown that many data analysis tasks of interest are compatible with differential privacy, and generally the loss in accuracy vanishes as the number n of individuals tends to infinity. However, in many cases, there is still is a price of privacy hidden in these asymptotics – in the rate at which the loss in accuracy vanishes, and in how large n needs to be to start getting accurate results at all (the "sample complexity").

1.2 The Sample Complexity of Private Learning

We study the price of privacy for several tasks in data analysis. Due to space limitations, we focus the presentation on one very important type of data analysis: *PAC learning*. Motivated by the observation that learning generalizes many of the analyses applied to large collections of data, Kasiviswanathan et al. [67] defined *private PAC learning* as a combination of probably approximately correct (PAC) learning [90] and differential privacy. For now, we can think of a PAC learner as an algorithm that operates on a set of classified random examples, and outputs a hypothesis h that misclassifies fresh examples with probability at most (say) $\frac{1}{10}$. It is assumed that all of the examples in the given set were classified by the same (unknown) classification rule c, called the *target concept*, taken from a (known) class of possible target concepts C. Importantly, the learner should *generalize* the input data into a hypothesis h that accurately predicts the classification of *unseen* random examples. In a case where h accurately classifies the input data, but fails to do so on fresh random examples, we say that the learner (or the hypothesis) does not *generalize*.

Intuitively, learning becomes "harder" as the class C becomes "richer". A natural problem is to characterize the *sample complexity* – the minimum number of examples necessary in order to identify a good hypothesis – as a function of the target class C. This important measure determines the amount of data that must be collected before starting the analysis.

Without privacy, it is well-known that the sample complexity of PAC learning is proportional to the Vapnik–Chervonenkis (VC) dimension of the class C [92, 19, 50]. We are interested in learning algorithms that are also differentially private. Informally, this means that the output of the learner (the chosen hypothesis) should not be significantly affected by any particular example. In the initial work on private learning, Kasiviswanathan et al. [67] proved that a private learner exists for every *finite* concept class. The proof is via a generic construction that exhibits sample complexity logarithmic in $|C|$. The VC dimension of a concept class C is always at most $\log|C|$, but is significantly lower for many interesting classes. Hence, the results of [67] left open the possibility that the sample complexity of private learning may be significantly higher than that of non-private learning.

In the case of *pure* differential privacy ($\delta = 0$), this gap in the sample complexity was shown to be unavoidable in general. Consider the task of *properly* learning a concept class C where, after consulting its sample, the learner outputs a hypothesis that is by itself in C. While non-privately this restriction has no effect on the sample complexity, Beimel, Brenner, Kasiviswanathan, and Nissim [8] showed that it can have a big impact for pure differentially private learners. Specifically, Beimel et al. considered the concept class of point functions over a data universe X, containing functions that evaluate to 1 on exactly one element of the domain X. This class has VC dimension 1 and hence can be (properly) learned without privacy with $O(1)$ samples. Under pure differential privacy, Beimel et al. presented an *improper* learner for point functions using $O(1)$ samples. In contrast, they showed that *properly* learning point functions with pure differential privacy requires sample complexity $\Omega(\log|X|)$, therefore separating the sample complexity of pure-private proper-learners from that of non-private learners (see also [26]). Feldman and Xiao [51] further showed that this separation holds even for *improper* learning, specifically, for the class of threshold functions. This is the class of all functions that evaluate to 1 on a prefix of the (totally ordered) domain X. While this class has VC dimension 1, Feldman and

Xiao showed that, under pure differential privacy, every (proper or improper) learner for it requires sample complexity $\Omega(\log|X|)$.

1.3 Our Contributions

We next describe the main contributions of this thesis. For space limitations, this thesis only includes part of our relevant results. Omitted results are described in Section 1.4.

1.3.1 Characterizing the Sample Complexity of Pure-Private Learners

In analogy to the characterization of the sample complexity of (non-private) PAC learners via the VC-dimension, in Chapter 5 we give a combinatorial characterization of the sample size sufficient and necessary for pure-private PAC (improper) learners. Towards obtaining this characterization, we introduce the notion of *probabilistic representation* of a concept class.

Beimel, Brenner, Kasiviswanathan, and Nissim [8] defined the notion of a *deterministic representation* for a concept class C. Informally, a deterministic representation for a concept class C is a hypotheses class H such that for every target concept c from C there exists a hypothesis $h \in H$ with small error w.r.t. c. Beimel et al. [8] showed that if $|H| < |C|$ then the deterministic representation can be used to improve the sample complexity of privately learning C: Given an input sample, use the generic construction of Kasiviswanathan et al. [67] to choose a good hypothesis out of H (instead from C). The generic construction of [67] requires sample complexity logarithmic in the size of the hypotheses class, and hence, the sample complexity is reduced to $\log|H|$. While for some classes this can dramatically improve the sample complexity, Beimel et al. showed that this technique is not optimal. For example, while the class C of point functions can be improperly learned (with pure privacy) using a constant sample complexity, they showed that every deterministic representation for it must be of size at least $|H| = \Omega(\log|C|)$.

We make an additional step in improving the sample complexity by considering a *probabilistic* representation of a concept class C. Instead of one collection H representing C, we consider a list of collections $H_1, \ldots, H_r$ such that for every $c \in C$ and every distribution on the examples, if we sample a collection H_i from the list, then with high probability there

is a hypothesis $h \in H_i$ that is close to c. To privately learn C, the learning algorithm first samples $i \in \{1,\ldots,r\}$ and then uses the generic construction of Kasiviswanathan et al. [67] to select a hypothesis from H_i. This reduces the sample complexity to $O(\max_i \log|H_i|)$; the *size* of the probabilistic representation is hence defined to be $\max_i \log|H_i|$. We show that the size of the smallest probabilistic representation of a class C, which we call the *representation dimension* and denote by $\mathrm{RepDim}(C)$, characterizes (up to constants) the sample size necessary and sufficient for privately learning the class C. The work in this chapter is joint with Amos Beimel and Kobbi Nissim (ITCS 2013) [10].

Following our work, Feldman and Xiao [51] showed an equivalence between the representation dimension of a concept C and the randomized one-way communication complexity of the evaluation problem for concepts from C. Using this equivalence, they separated the sample complexity of pure-private learners from that of non-private ones. For example, they showed a lower bound of $\Omega(\log|X|)$ on the sample complexity of every pure-private (proper or improper) learner for the class of threshold functions over a domain X. This is a strong separation from the non-private sample complexity, which is $O(1)$ (as the VC dimension of this class is constant). Thus, the sample complexity of pure-private learners (proper or improper) can generally be much higher than what is required for non-private learning.

1.3.2 Private Learning: Pure vs. Approximate Differential Privacy

In Chapter 6, we show that relaxing the privacy requirement from pure to approximate differential privacy can drastically reduce the sample complexity of private learners. In particular, we show that threshold functions can be learned with approximate-privacy using $2^{O(\log^*|X|)}$ examples, a dramatic improvement over the $\Omega(\log|X|)$ lower bound on the sample complexity of every pure-private learner for this class [51]. For point functions, we construct an approximate-private *proper*-learner with $O(1)$ sample complexity, again circumventing the lower bound of $\Omega(\log|X|)$ for pure-private proper-learners.

Private Binary Search. Towards constructing our learners, we introduce a useful tool for performing binary searches while preserving privacy: Consider a set F of possible solutions, and let $q : X^n \times F \to \mathbb{R}$ be a function that, given an input database $S \in X^n$, assigns

a number to every possible solution in F. Our goal is to (privately) choose a solution $f \in F$ such that $q(S, f)$ is as close as possible to a given parameter t.

Under some assumptions on the function q, this task could be solved using the *exponential mechanism* of McSherry and Talwar [74] (to be surveyed in Chapter 3). This generic mechanism is capable of privately identifying a solution f s.t. $q(S, f) \approx t$, provided that $|S| \geq O(\log|F|)$. By relaxing the privacy guarantees from pure to approximate differential privacy, we show that it is possible to significantly reduce the needed database size to $|S| \geq 2^{O(\log^* |F|)}$, whenever the solution set F is totally ordered and $q(S, \cdot)$ is monotone. As we will see, this abstraction captures the task of (privately) learning threshold functions, and hence, our tool yields an improved upper bound on the sample complexity of learning threshold functions under approximate differential privacy. The work in this chapter is joint with Amos Beimel and Kobbi Nissim (RANDOM 2013 and Theory of Computing) [11, 13].

1.3.3 Private Learning of Threshold Functions

Our positive results for approximate-private learners with very low sample complexity raise the possibility that the sample complexity of approximate-private proper-learners is actually of the same order as that of non-private learners. In Chapter 7, however, we show that the sample complexity of proper learning with approximate differential privacy can be asymptotically larger than the VC dimension. Specifically, we show for the first time that private proper-learning of threshold functions is *impossible* when the data universe is infinite (e.g., $\mathbb{N}$ or $[0, 1]$) and in fact that the sample complexity must grow with the size $|X|$ of the data universe: $n = \Omega(\log^* |X|)$, which is tantalizingly close to the previously mentioned upper bound of $n = 2^{O(\log^* |X|)}$.

This is the first non-trivial lower bound on the sample complexity of approximate-private learners. As the VC dimension of the class of thresholds is one, this result also separates the sample complexity of approximate-private proper-learners from that of non-private learners. Our lower bound extends to the concept class of ℓ-dimensional thresholds. An ℓ-dimensional threshold function, defined over the domain X^ℓ, is a conjunction of ℓ threshold functions, each defined on one component of the domain. While the VC-dimension of this class is ℓ, our lower bound shows that $\Omega(\ell \cdot \log^* |X|)$ samples are required

for every approximate-private proper-learner for this class. This shows that our separation between the sample complexity of private and non-private learning applies to concept classes of every VC dimension.

Inspired by the techniques used to prove our lower bounds, we give an algorithm for learning thresholds with $n \leq 2^{(1+o(1))\log^*|X|}$ samples. This improves our previous upper bound of $8^{(1+o(1))\log^*|X|}$. Based on these results, it would be interesting to fully characterize the difference between the sample complexity of non-private learners and of proper learners with (approximate) differential privacy. Even showing a stronger separation (compared to $\log^*|X|$) would be interesting. Furthermore, our results still leave open the possibility that *improper* PAC learning with (approximate) differential privacy has sample complexity $O(\mathrm{VC}(C))$. We consider this to be an important question for future work. The work in this chapter is joint with Mark Bun, Kobbi Nissim, and Salil Vadhan (FOCS 2015) [22].

1.3.4 The Generalization Properties of Differential Privacy

As supported by our discussion above, many learning tasks of interest are compatible with differential privacy, i.e., private analogues exist for many learning tasks. In Chapter 4, our first technical chapter, we start our exploration of intersection points between differential privacy and learning theory by showing that, essentially, all you can do with differential privacy is to learn. Namely, differential privacy *implies* learning. Recall that the general task of computational learning is to identify properties of the underlying distribution, rather than properties of a given sample. Roughly speaking, we will show that if a property of a given sample is identified by a differentially private computation, then this property is in fact a property of the underlying distribution. In other words, differential privacy guarantees generalization.

This connection between differential privacy and generalization was first suggested by McSherry.[1] At a high level, McSherry observed that if a predicate $h : X \to \{0,1\}$ is the result of an (ε, δ)-differentially private computation on a sample S containing i.i.d. elements from a distribution $\mathcal{D}$, then the empirical average $h(S) = \frac{1}{|S|}\sum_{x\in S} h(x)$ and the expectation $h(\mathcal{D}) = \mathbb{E}_{x\sim\mathcal{D}}[h(x)]$ are close *in expectation*.

[1] See, e.g., [73], although the observation itself dates back at least to 2008.

This connection has the potential of using differential privacy *as a mathematical tool* in order to guarantee generalization, even in settings which are not focused on privacy. Indeed, Dwork et al. [40] used this connection to develop techniques for preventing overfitting and false discoveries in empirical research. Consider a data analyst interested in testing a specific research hypothesis. The analyst acquires relevant data, evaluates the hypothesis, and (say) learns that it is false. Based on her discoveries, the analyst now decides on a second hypothesis to be tested, and evaluates it *on the same data* (acquiring fresh data might be too expensive or even impossible). That is, the analyst chooses her hypotheses *adaptively*, where this choice depends on previous interactions with the data. As a result, her findings are no longer supported by classical statistical theory, which assumes that the tested hypotheses are fixed before the data is gathered, and the analyst is at risk of overfitting to the data. This problem has been identified as one of the primary explanations of why research findings are frequently false (see, e.g., [64, 55]).

Dwork et al. model the problem as follows. The analyst, who is interested in learning properties of an unknown distribution $\mathcal{D}$, interacts with a *data curator* $\mathcal{A}$ holding a database S containing i.i.d. samples from $\mathcal{D}$. The interaction is adaptive, where at every round the analyst specifies a *statistical query* $q : X \to \{0,1\}$ and receives an answer $a_q(S)$ that (hopefully) approximates $q(\mathcal{D}) = \mathbb{E}_{x\sim\mathcal{D}}[q(x)]$. Dwork et al. focused on the question of how many samples S should contain to enable $\mathcal{A}$ to answer accurately.

Let k be the number of queries the analyst makes. Using the Hoeffding bound, it is easy to see that if all k queries are fixed before interacting with $\mathcal{A}$, then $\mathcal{A}$ could simply answer every query q with its empirical average on S, i.e., letting $a_q(S) = \frac{1}{|S|}\sum_{x\in S} q(x)$. Having $|S| = O\left(\frac{\log(k)}{\alpha^2}\right)$ would then suffice for making *all* k answers α-approximate. This approach would however fail for the case where queries are chosen adaptively. Surprisingly, using the connection between differential privacy and generalization, Dwork et al. [40] showed that it is possible to answer k adaptively chosen queries when $|S|$ grows logarithmically with k.

In a nutshell, the idea of [40] is as follows. Suppose that $\mathcal{A}$ is a differentially private algorithm, capable of approximating the *empirical average* of k adaptively chosen queries (such mechanisms were intensively studied and constructed by the differential privacy community). Then, by the inherent generalization properties of differential privacy, $\mathcal{A}$'s

answers must also be accurate w.r.t. the underlying distribution!

Of course, if differential privacy only guaranteed generalization in expectation, then the answers would only be accurate in expectation, while we are typically interested in high probability guarantees. To that end, Dwork et al. [40] substantially strengthened this connection and showed that differential privacy implies generalization *with high probability*. Specifically, they showed that every (ε, δ)-differentially private computation guarantees $O(\varepsilon)$ accuracy with probability $1 - O(\delta^{\varepsilon})$. This enabled Dwork et al. to obtain algorithms that accurately answer adaptively chosen statistical queries, where accuracy is guaranteed with high probability.

In Chapter 4 we strengthen the connection between differential privacy and generalization, and show that every (ε, δ)-differentially private computation guarantees $O(\varepsilon)$ accuracy with probability $1 - O(\delta/\varepsilon)$. Observe that the failure probability in our bound is significantly improved over the previous result of Dwork et al. This improvement enables us to show improved upper bounds on the number of samples n that are needed to answer adaptively chosen statistical queries. We also show that our bound is tight, i.e., there exists an (ε, δ)-differentially private computation that achieves error ε with probability $\Theta(\delta/\varepsilon)$. The work in this chapter is joint with Raef Bassily, Kobbi Nissim, Adam Smith, Thomas Steinke, and Jonathan Ullman (STOC 2016) [6].

1.4 Additional Contributions

We next describe additional results, which are omitted from this thesis due to space limitations.

1.4.1 Semi-Supervised Private Learners

Recall that a PAC learner operates on a set of *classified* random examples, and outputs a hypothesis that should be able to predict the classification of unseen examples. As mentioned above, in Chapter 7 we show that, even with approximate-privacy, the sample complexity of private learners can be asymptotically higher than that of non-private learners. In [12] we study an alternative approach for reducing the costs of private learning. Our work is inspired by the (non-private) models of *semi-supervised learning* [91] and *active*

learning [72], where the focus is on the sample complexity of *classified* examples whereas *unclassified* examples are of a significantly lower cost. Consider, for example, a hospital conducting a study on a new disease. The hospital may already possess background information about individuals and hence can access a large pool of unclassified examples, but in order to classify an example, an actual medical test is needed. In such scenarios it makes a lot of sense to try and use a combination of both classified and unclassified examples in order to reduce the required amount of classified data.

Using two different constructions, we show that the classified sample complexity of private learners is characterized by the VC dimension. Our first construction is of learners where the classified sample complexity is proportional to the VC dimension of the concept class C; however, the unclassified sample complexity of the algorithm is as big as $\log|C|$. Our second construction presents a new technique for decreasing the classified sample complexity of a given private learner, while roughly maintaining its unclassified sample complexity. In addition, we show that in some settings the classified sample complexity does not depend on the privacy parameters of the learner. This work is joint with Amos Beimel and Kobbi Nissim (SODA 2015) [12].

1.4.2 Simultaneous Private Learning of Multiple Concepts

Currently, no lower bounds are known on the sample complexity of approximate-private *improper*-learners. On the other hand, there are no positive results for approximate-private improper-learners that improve on the sample complexity of approximate-private proper-learners. In [21] we explore a setting in which the sample complexity of approximate-private improper-learners is provably higher than that of non-private learners. Specifically, we investigate the *direct-sum* problem in the context of differentially private PAC learning: What is the sample complexity of solving k learning tasks *simultaneously* under differential privacy, and how does this cost compare to that of solving k learning tasks without privacy?

The direct-sum problem has its roots in complexity theory, and is a basic problem for many algorithmic tasks. It also has implications for the practical use of differential privacy. Consider, for instance, a hospital that collects information about its patients and wishes to use this information for medical research. The hospital records for each patient a collection of attributes such as age, sex, and the results of various diagnostic tests, and, for

each of k diseases, whether the patient suffers from the disease. Based on this collection of data, the hospital researchers wish to learn good predictors for the k diseases. One option for the researchers is to perform each of the learning tasks on a fresh sample of patients, hence enlarging the number of patient examples needed (i.e., the sample complexity) by a factor of k, which can be very costly.

Without concern for privacy, the sample complexity that is necessary and sufficient for performing the k learning tasks is actually fully characterized by the VC dimension of the target concept class – it is independent of the number of learning tasks k. In [21], we examine if the situation is similar when the learning is performed with differential privacy. Interestingly, we see that with differential privacy the picture is quite different, and in particular, the required number of examples can grow polynomially in k (even for improper-learning under approximate-differential privacy). This work is joint with Mark Bun and Kobbi Nissim (ITCS 2016) [21].

1.4.3 Locating a Small Cluster Privately

Our exploration of the price of privacy in data analysis is not limited to PAC learners. Clustering – the task of grouping data points by their similarity – is one of the most commonly used techniques for exploring data, for identifying structure in uncategorized data, and for performing a variety of machine learning and optimization tasks. In [78], we present a new differentially private tool for a clustering-related task: Given a collection S of n points in the ℓ-dimensional Euclidean space $\mathbb{R}^\ell$ and a parameter t reflecting a target number of points, our goal is to find a smallest ball containing at least t of the input points, while preserving differential privacy.

Finding an optimal solution for this problem is NP-hard even regardless of privacy requirements [82], but good approximation solutions exist. In fact, without privacy, there is a PTAS for computing a ball of radius $(1+\alpha)r_{opt}$ containing t points, where r_{opt} is the radius of a smallest ball enclosing t points, and α is the approximation parameter [1]. In [78], we present a differentially-private algorithm capable of identifying a ball of radius $O\left(\sqrt{\log(n)} \cdot r_{opt}\right)$, enclosing (almost) t points. Our algorithm has implications to private data exploration, clustering, and removal of outliers. Furthermore, we use it to significantly relax the requirements of the sample and aggregate technique [76], which

allows compiling of "off the shelf" (non-private) analyses into analyses that preserve differential privacy. Interestingly, an important ingredient in our construction is the tool for conducting a (sample efficient) private binary search, which we developed towards constructing an approximate-private proper-learner for the class of threshold functions (see Section 1.3.2 of this introduction, or Chapter 6 for the technical details). This work is joint with Kobbi Nissim and Salil Vadhan (PODS 2016) [78].

1.4.4 Private Query Release

Another very important type of data analysis that we consider is *private query release*. Given a set Q of queries $q : X^n \to \mathbb{R}$, the *query release* problem for Q is to output accurate answers to all queries in Q. That is, we want a differentially private algorithm $\mathcal{M} : X^n \to \mathbb{R}^{|Q|}$ such that for every database $S \in X^n$, with high probability over $y \leftarrow \mathcal{M}(S)$, we have that $|y_q - q(S)|$ is at most (say) $\frac{1}{10}$ for all $q \in Q$.

A special case of interest is the case where Q consists of *counting queries*. In this case, we are given a set Q of predicates $q : X \to \{0,1\}$ on individual rows, and then extend them to databases by averaging. That is, $q(S) = \frac{1}{n}\sum_{x \in S} q(x)$ counts the fraction of individuals in the database that satisfy predicate q.

The query release problem for counting queries is one of the most widely studied problems in differential privacy. Early work on differential privacy implies that for every family of counting queries Q, the query release problem for Q has "sample complexity" at most $\tilde{O}(\sqrt{|Q|})$ [35, 46, 15, 44]. That is, there is an $n_0 = \tilde{O}(\sqrt{|Q|})$ such that for all $n \geq n_0$, there is a differentially private mechanism $\mathcal{M} : X^n \to \mathbb{R}^Q$ that solves the query release problem for Q.

Remarkably, Blum, Ligett, and Roth [17] showed that if the data universe X is finite, then the sample complexity grows much more slowly with $|Q|$ — indeed the query release problem for Q has sample complexity at most $O(\log|Q| \cdot \log|X|)$. Hardt and Rothblum [59] improved this bound to $\tilde{O}(\log|Q| \cdot \sqrt{\log|X|})$, which was recently shown to be optimal for some families Q [24].

However, for specific query families of interest, the sample complexity can be significantly smaller. In particular, consider the family of *point functions* over a domain X, and the family of *threshold functions* over a totally ordered domain X. The query release

problems for these families correspond to the very natural tasks of producing ℓ_∞ approximations to the histogram and to the cumulative distribution function of the empirical data distribution, respectively. Recall that in Chapter 6 (see also Section 1.3.2 of this introduction) we develop techniques for constructing approximate-private proper-learners with low sample complexity. Those techniques can also be used to construct sample efficient algorithms for answering all point and threshold functions. We show that for point functions the sample complexity has no dependence on $|X|$ (or $|Q|$, since $|Q| = |X|$ for these families), and for threshold functions, it has at most a very mild dependence, namely $2^{O(\log^* |X|)}$.

As with proper-learning of threshold functions, our results from Chapter 7 (see also Section 1.3.3 in this introduction) can be used to show that the sample complexity of releasing threshold functions over a data universe X with approximate differential privacy is at least $\Omega(\log^* |X|)$. In particular, there is no differentially private algorithm for releasing threshold functions over an infinite data universe.

Chapter 2

Related Literature

There is now an extensive amount of related literature on differential privacy, which we cannot hope to cover here. Part of that literature, which is most relevant to this thesis, is described throughout the introduction. Next we survey some additional related results, and refer the reader to the excellent surveys in [47] and [89].

2.1 The Computational Price of Differential Privacy

In this thesis we study the effects of privacy on the sample complexity of data analysis. Another very basic measure for the price of privacy in data analysis is *computational complexity*, i.e., the required increase in *running time* in order to guarantee privacy while analyzing data.

2.1.1 Private Learning

Recall that a PAC learner is given a set of random examples, all of which are classified by some *fixed* target concept, and aims at identifying a good hypothesis w.r.t. this fixed target concept (and the underlying distribution). A basic problem in learning theory, however, is to construct learning algorithms that are able to cope with incorrectly classified examples. In the *random classification noise* model of Angluin and Laird [2], the label of each example is flipped with some fixed probability η, called the *noise rate*. A powerful framework for constructing computationally-efficient noise-tolerant learning algorithms is the *statistical queries (SQ)* learning model of Kearns [68]. In the SQ model, instead of accessing examples directly, the learner can specify some properties on the examples, for which he is given an estimate of the probability that a random example satisfies the property. As Kearns

showed, any class of functions that is learnable from statistical queries is learnable with random classification noise. Moreover, the SQ model is known to be very powerful, and in fact, most existing PAC learners have SQ analogues.

Intuitively, noise-tolerant learning seems to be related to private learning as both kinds of learners are required to be robust to small changes in the input. Indeed, Blum et al. [15] showed that any learner in the SQ model can be transformed to preserve differential privacy, while maintaining computational efficiency. As the SQ model captures most of the efficiently learnable classes, this implies that many computational learning tasks that are efficiently learnable non-privately can be learned privately and efficiently. In addition, Kasiviswanathan et al. [67] showed an example of a concept class – the class of *parity functions* – that is not learnable in the statistical queries model but can be learned privately and efficiently (learning parity with noise is conjectured to be computationally hard). However, as was recently shown by Bun and Zhandry [25], there are concept classes that are efficiently PAC learnable, but for which every efficient learner fails to be differentially private (under plausible computational hardness assumptions).

2.1.2 Private Query Release

Consider again the query release problem for counting queries, where we seek a mechanism that, given a database $S \in X^n$, releases approximate answers to all queries in some family of counting queries Q. When $|Q|$ is large, an efficient algorithm cannot output an answer for every $q \in Q$ directly. Instead, it is required to produce a *data structure* capable of approximating $q(S)$ for every $q \in Q$. It is especially desirable to construct mechanisms whose output is defined in terms of an alternative database $\hat{S} \in X^*$ satisfying $q(S) \approx q(\hat{S})$ for every $q \in Q$. Such an alternative database $\hat{S}$ is called a *synthetic database*.

Without privacy, computing a "synthetic database" is trivial, as it is possible to simply output the input database S. With privacy, computing a synthetic database is known to be computationally feasible for some families Q of counting queries. For example, our techniques from Chapter 6 can be used to construct time efficient algorithms producing a synthetic database for all point and threshold functions. In general, however, this is not the case, and hardness results were given in a number of works, originating with the work of Dwork, Naor, Reingold, Rothblum, and Vadhan [45]. Interestingly, those hardness results

are based on a connection to *traitor-tracing schemes* – a cryptographic tool for preventing illegal distribution of digital content – introduced in 1994 by Chor et al. [29].

Indeed, under standard hardness assumptions, Ullman and Vadhan [88] proved the existence of a "natural" family Q of counting queries on a data universe X, where $|Q| = O(\log^2|X|)$, such that there is no $n = \text{poly}(\log|X|)$ and a polynomial-time differentially private algorithm that takes a database $S \in X^n$ and outputs a synthetic database $\hat{S} \in X^*$ providing accurate answers to every query in Q. That is, producing synthetic databases is hard even for simple families of counting queries of size $o(n)$.

One might hope that by allowing our mechanisms to output an arbitrary data-structure, rather than a synthetic database, private query release would become feasible for large families of queries, of size exponential in n. However, under plausible hardness assumptions, a recent result by Kowalczyk et al. [71] shows that this is not the case. Specifically, for every n, there is a family Q of $\tilde{O}(n^7)$ counting queries on a data universe X such that there is no polynomial-time differentially private algorithm that takes a database $S \in X^n$ and outputs accurate answers to every query in Q. See also [45].

The best known *efficient* mechanism for answering an arbitrary set of counting queries is the Laplace mechanism [44] (to be surveyed in Chapter 3), capable of approximating $\tilde{O}(n^2)$ queries. Seemingly, therefore, there is a gap between the best general construction, answering $\tilde{O}(n^2)$ arbitrary counting queries, and the hardness results for families of $\tilde{O}(n^7)$ counting queries. Actually, the Laplace mechanism is *interactive* (or *universal*) in the sense that the queries are not fixed but are instead given as input to the mechanism. The above mentioned hardness results are for the non-interactive setting, stating that for every n there is a family of size $\tilde{O}(n^7)$ that is hard to approximate by *any* differentially private mechanism operating on databases of size n. As Ullman [87] showed, under standard hardness assumptions, there is no private and efficient universal algorithm that accurately answers more than $\tilde{O}(n^2)$ arbitrary counting queries, and the Laplace mechanism is (in general) optimal for the interactive setting.

2.2 Interactive Query Release

In Chapters 6 and 7 we study the sample complexity of privately learning the class of threshold functions. As we mentioned, our techniques also apply to the task of private query release. Specifically, we show that there is an efficient mechanism that privately releases approximate answers to all of the threshold functions over a domain X, using a database of size $n = 2^{O(\log^* |X|)}$. Moreover, every such mechanism requires a database of size $n = \Omega(\log^* |X|)$.

In [23], Bun, Steinke, and Ullman considered the problem of privately releasing approximate answers to *adaptively* given threshold queries. Consider, for example, the class of threshold functions over an *infinite* domain X. As our results in Chapter 7 show, it is impossible to privately release approximate answers to *all* of those (infinitely many) threshold functions. Nevertheless, if we are only interested in getting accurate answers for k of the threshold functions, which are fixed in advance, then a database of size $n = 2^{O(\log^* k)}$ suffices to produce accurate answers to those k threshold functions (by simply ignoring all of the other functions).

What can we do if the queries are given one by one, in an on-line fashion? The trivial solution would be to use the Laplace mechanism capable of privately approximating k adaptively chosen (arbitrary) counting queries (not limited to thresholds). This solution requires a database of size $n \approx \sqrt{K}$. Bun et al. [23] presented an efficient mechanism for privately approximating k adaptively chosen thresholds using a database of size logarithmic in k.

Recall that in the general setting – where the adaptive queries are not restricted to a sub family of counting queries – privately answering more than $\tilde{O}(n^2)$ queries is computationally hard [87]. Hence, by restricting the adaptive queries to come from a specific sub family of counting queries, it is possible to answer *exponentially* more queries efficiently. See also [18].

Another interesting approach is the following. Consider a large number of arbitrary counting queries $f_1, f_2, \ldots$, which are given (one by one) to a data curator (holding a database $S \in X^n$). In every round, we would like to receive, in a differentially private manner, an approximation to $f_i(S)$. In some cases, however, we might only be interested in obtaining

answers for the queries f_i whose answers are "meaningful". More specifically, we have some threshold T in mind, and we only care to receive answers to queries s.t. $f_i(S) \geq T$. Dwork, Naor, Reingold, Rothblum, and Vadhan [45] presented a simple (and elegant) tool for such a scenario – Algorithm AboveThreshold. Loosely speaking, Algorithm AboveThreshold is capable of answering c "meaningful" queries (out of a stream of k queries) using a database of size $n = \tilde{O}(\log(k) \cdot \sqrt{c})$. Moreover, it is computationally efficient.

The interactive query release problem was also studied from an information-theoretic perspective, showing that (ignoring running time) it is possible to privately answer an exponential number of arbitrary counting queries, chosen adaptively. This was first done by Roth and Roughgarden [80] who presented a mechanism that answers k such queries using a database of size $\approx \log|X| \cdot \log^3(k)$. Their work was improved by Hardt and Rothblum [59], who presented a mechanism with both improved accuracy and running time, capable of answering k adaptively chosen queries using a database of size $\approx \sqrt{\log|X|} \cdot \log(k)$. The results of [80] and [59] were later unified by Gupta, Roth, and Ullman [57], who presented framework that generalizes both mechanisms.

Recall that (interactive) query release mechanisms should provide answers which are accurate w.r.t. their input database S. If the input database S was sampled i.i.d. from some underlying distribution $\mathcal{D}$, then such mechanisms are capable of privately approximating the *empirical average* of k adaptively chosen predicates (i.e., counting queries). As we mentioned in Section 1.3.4, such mechanisms will be used as an important building block in Chapter 4, where we construct of algorithms that answer adaptively chosen statistical queries *w.r.t. the underlying distribution*.

2.3 Answering Adaptively Chosen Statistical Queries

Multiple hypothesis testing is a ubiquitous task in empirical research. A finite sample of data is drawn from some unknown population, and several analyses are performed on that sample. The outcome of an analysis is deemed significant if it is unlikely to have occurred by chance alone, and a "false discovery" occurs if the analyst incorrectly declares an outcome to be significant. False discovery has been identified as a substantial problem in the scientific community (see, e.g., [64, 55]). This problem persists despite decades of

research by statisticians on methods for preventing false discovery, such as the widely used Bonferroni Correction [20, 36] and the Benjamini-Hochberg Procedure [14].

False discovery is often attributed to misuse of statistics. An alternative explanation is that the prevalence of false discovery arises from the inherent *adaptivity* in the data analysis process – the fact that the choice of analyses to perform depends on previous interactions with the data (see, e.g., [55]). Adaptivity is essentially unavoidable when a sequence of research groups publish research papers based on overlapping data sets. Adaptivity also arises naturally in other settings, for example: in multistage inference algorithms where data are preprocessed (say, to select features or restrict to a principal subspace) before the main analysis is performed; in scoring data-based competitions [16]; and in the re-use of holdout or test data [41, 39].

The general problem of adaptive data analysis was formally modeled and studied in recent papers by Dwork, Feldman, Hardt, Pitassi, Reingold, and Roth [40] and by Hardt and Ullman [61] as follows: Suppose there is an unknown distribution $\mathcal{D}$ and a set of n independent samples S is drawn from $\mathcal{D}$. We seek an algorithm that, given S as input, accurately answers a sequence of *adaptively chosen* statistical queries about the unknown distribution $\mathcal{D}$. How many samples n must we draw from the distribution, as a function of the number of queries k, and the desired level of accuracy α?

The textbook solution to this problem would be to split the sample S into k chunks, and then to answer every given query using its empirical average on a "fresh" chunk of the data. In order to answer k queries, this solution requires a sample of size $n = \Omega(k/\alpha^2)$. In the non-adaptive setting, where the k queries are fixed before the data is gathered, it is possible to simply answer every query using its empirical average on the sample S, i.e., letting $a_q(S) = \frac{1}{|S|}\sum_{x \in X} q(s)$. By the Hoeffding bound, a sample of size $n = O(\log(k)/\alpha^2)$ would then suffice for making *all* k answers accurate to within error α. This approach would however fail for the case where queries are chosen adaptively, and it can easily be shown that a sample of size linear in k is necessary in order for the empirical average to maintain accuracy.

The striking results of Dwork et al. [40] gave the first nontrivial algorithms for provably ensuring statistical validity in adaptive data analysis, allowing for even an *exponential* number of tests against the same sample. However, their construction is computationally

inefficient. They also presented an efficient algorithm allowing to answer k adaptively chosen statistical queries to within error α using a sample containing $n = \tilde{O}(\sqrt{k}/\alpha^{2.5})$. The dependency of this upper bound in k was shown to be tight by Hardt and Ullman [61] and Steinke and Ullman [85]. Under standard hardness assumptions, they showed that a sample of size $n = \Omega(\sqrt{k}/\alpha)$ is necessary in order to answer k adaptively chosen statistical queries to within accuracy α. In Chapter 4 we present an improved upper bound, showing that a sample of size $n = \tilde{O}(\sqrt{k}/\alpha^2)$ suffices.

2.4 Other Related Work

A line of research (started by Schapire [81]) that is very relevant to this thesis is boosting learning algorithms, that is, taking a learning algorithm that has a big classification error and producing a learning algorithm with small error. The construction of boosting algorithms has received a lot of attention after the seminal work of [81] (see, e.g., [54, 52, 34]); perhaps the most significant construction is the AdaBoost algorithm of Schapire and Freund [53] (who won the Gödel Prize for their work).

In 2010, Dwork, Rothblum, and Vadhan [48] presented a differentially private variant of the AdaBoost algorithm, where given a *private* learning algorithm with large classification error, they produce a *private* learning algorithm with small error. In Chapter 5, we give an alternative technique for boosting the accuracy of pure-private learners, whose proof is simpler. However, it is (generally) not computationally efficient.

This research theme of constructing private variants to influential learning algorithms is not limited to boosting algorithms. A lot of the works in this vein, however, does not exactly fit the PAC learning model, and are thus less relevant to this thesis. For example, Chaudhuri et al. [27] gave a general technique for producing privacy-preserving variants for convex *empirical risk minimization (ERM)* algorithms. Their results were extended by Kifer et al. [70] and Bassily et al. [7], who produced algorithms with improved error rates. In particular, those results yield private variants to logistic regression and support vector machine classifiers.

Chapter 3

Background and Preliminaries

In this chapter we introduce the necessary preliminaries for the main tasks discussed in this thesis. Additional preliminaries will be given throughout when needed. We use X to denote an arbitrary domain, and use X^n for the cartesian n^{th} power of X, i.e., $X^n = (X)^n$.

3.1 Differential privacy

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

Definition 3.1. *Databases $S_1 \in X^n$ and $S_2 \in X^n$ over a domain X are called* neighboring *if they differ in exactly one entry.*

Definition 3.2 (Differential Privacy [44, 38, 42])**.** *A randomized algorithm $\mathcal{A} : X^n \to Y$ is (ε, δ)-differentially private if for all neighboring databases $S_1, S_2 \in X^n$, and for all sets $F \subseteq Y$ of outputs,*

$$\Pr[\mathcal{A}(S_1) \in F] \leq \exp(\varepsilon) \cdot \Pr[\mathcal{A}(S_2) \in F] + \delta. \tag{3.1}$$

The probability is taken over the random coins of $\mathcal{A}$. When $\delta = 0$ we omit it and say that $\mathcal{A}$ preserves ε-differential privacy.

We use the term *pure* differential privacy when $\delta = 0$ and the term *approximate* differential privacy when $\delta > 0$, in which case δ is typically a negligible function of the database size n.

3.1.1 Properties of Differential privacy

Suppose that a differentially private algorithm $\mathcal{A}$ is executed on a (sensitive) database S, and that the outcome y is publicly released. Once released, arbitrary additional analyses might be applied to the outcome y. It is out of our hands. An important property of differential privacy is its resiliency to *post-processing*, ensuring that the privacy guarantees could not be weakened by any such additional analyses.

Theorem 3.3 (post-processing)**.** *If $\mathcal{A} : X^n \to Y$ is (ε, δ)-differentially private, and $\mathcal{B} : Y \to Z$ is any randomized function, then $\mathcal{B}(\mathcal{A}(\cdot)) : X^n \to Z$ is (ε, δ)-differentially private.*

We will later present algorithms that access their input database using (several) differentially private mechanisms. For example, let $\mathcal{A}_1 : X^n \to Y$ and $\mathcal{A}_2 : X^n \to Y$ be two (ε, δ)-differentially private mechanisms. Now suppose we construct an algorithm $\mathcal{B} : X^{2n} \to Z$ that takes a database of size $2n$, and applies $\mathcal{A}_1$ to the first n elements, and $\mathcal{A}_2$ to the last n elements (and does not access the database otherwise). Theorem 3.3 immediately implies that algorithm $\mathcal{B}$ is (ε, δ)-differentially private. To see this, observe that for every fixture of the first n elements, algorithm $\mathcal{B}$ is a post-processing of the outcome of $\mathcal{A}_2$. Similarly, for every fixture of the last n elements, algorithm $\mathcal{B}$ is a post-processing of the outcome of $\mathcal{A}_1$. Privacy is therefore preserved, as two neighboring databases of size $2n$ either differ on one of the first n elements, or on one of the last n elements, but not on both.

So, an algorithm that applies several (ε, δ)-differentially private mechanisms to *different* portions of its database (and does not access the database otherwise), remains (ε, δ)-differentially private. What about algorithms that apply several private mechanisms onto the *same* portion of the database? As the following composition theorems show, privacy is still preserved, but the privacy guarantees (gracefully) deteriorate.

Theorem 3.4 ([42])**.** *If $\mathcal{A}_1$ and $\mathcal{A}_2$ satisfy $(\varepsilon_1, \delta_1)$ and $(\varepsilon_2, \delta_2)$ differential privacy, respectively, then their concatenation $\mathcal{A}(S) = \langle \mathcal{A}_1(S), \mathcal{A}_2(S) \rangle$ satisfies $(\varepsilon_1 + \varepsilon_2, \delta_1 + \delta_2)$-differential privacy.*

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

Theorem 3.5 ([42, 43])**.** *A mechanism that permits k adaptive interactions with mechanisms that preserve (ε, δ)-differential privacy (and does not access the database otherwise) ensures $(k\varepsilon, k\delta)$-differential privacy.*

Note that the privacy guaranties in the theorem deteriorates linearly with the number of interactions. By bounding the *expected* privacy loss in each interaction (as opposed to worst-case), Dwork et al. [48] showed the following stronger composition theorem, where privacy deteriorates (roughly) as $\sqrt{k}\varepsilon + k\varepsilon^2$ (rather than $k\varepsilon$).

Theorem 3.6 ([48], restated)**.** *Let $0 < \varepsilon, \delta' \leq 1$, and let $\delta \in [0, 1]$. A mechanism that permits k adaptive interactions with mechanisms that preserve (ε, δ)-differential privacy (and does not access the database otherwise) ensures $(\varepsilon', k\delta + \delta')$-differential privacy, for $\varepsilon' = \sqrt{2k \ln(1/\delta')} \cdot \varepsilon + 2k\varepsilon^2$.*

This composition theorem was subsequently improved by [65], who presented an *optimal* composition bound for differential privacy. Unfortunately, this bound is quite complex, and is #P-hard to compute exactly [75].

Another important property of differential privacy is *group privacy*. Recall that differential privacy ensures that every *single* individual does not have a significant effect on the outcome (distribution) of the computation. A similar guarantee also holds for every small *group* of individuals:

Theorem 3.7 (group privacy)**.** *If $\mathcal{A} : X^n \to Y$ is (ε, δ)-differentially private, then for all pairs of databases $S_1, S_2 \in X^n$ that differ in at most k entries, and for every set of outputs $F \subseteq Y$,*

$$\Pr[\mathcal{A}(S_1) \in F] \leq e^{k\varepsilon} \cdot \Pr[\mathcal{A}(S_2) \in F] + k \cdot e^{k\varepsilon} \cdot \delta.$$

3.2 Preliminaries from Learning Theory

A concept $c : X \to \{0, 1\}$ is a predicate that labels *examples* taken from the domain X by either 0 or 1. A *concept class* C over X is a set of concepts (predicates) mapping X to $\{0, 1\}$.

We now present two simple concept classes that will reappear throughout this thesis, and serve as running examples throughout this chapter.

Definition 3.8 (Point Functions)**.** *Let X be any domain. The class of* point functions *is the set of all predicates that evaluate to 1 on exactly one element of X, i.e.,*

$$\texttt{POINT}_X = \{c_x : x \in X\} \quad \textit{where} \quad c_x(y) = 1 \textit{ iff } y = x.$$

Definition 3.9 (Threshold Functions)**.** *Let X be any totally ordered domain. The class of* threshold functions *is the set of all predicates that evaluate to 1 on a prefix of X, i.e.,*

$$\texttt{THRESH}_X = \{c_x : x \in X\} \quad \textit{where} \quad c_x(y) = 1 \textit{ iff } y \leq x.$$

3.2.1 The PAC Model

Let C be a concept class over a domain X. A learning algorithm for the class C is given examples sampled according to an unknown probability distribution $\mathcal{D}$ over X, and labeled according to an unknown *target* concept $c \in C$. The learning algorithm is successful when it outputs a hypothesis h that approximates the target concept over samples from $\mathcal{D}$. More formally:

Definition 3.10 (Generalization Error)**.** *Let $h : X \to \{0,1\}$ be a hypothesis. The* generalization error *of h w.r.t. a concept $c : X \to \{0,1\}$ and a distribution $\mathcal{D}$ on X is defined as*

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

More generally, we define the generalization error *of h w.r.t. a distribution $\mathcal{P}$ over $X \times \{0,1\}$ as*

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

*We say that h is α-*good *whenever its generalization error is at most α.*

Definition 3.11 (PAC Learning [90])**.** *Algorithm $\mathcal{A}$ is an (α, β)-*PAC learner *with sample size n for a concept class C over X using hypothesis class H if for all concepts $c \in C$, all distributions*

$\mathcal{D}$ on X, given an input of n samples $S = (z_1, \ldots, z_n) \in (X \times \{0,1\})^n$, where $z_i = (x_i, c(x_i))$ and each x_i is drawn i.i.d. from $\mathcal{D}$, algorithm $\mathcal{A}$ outputs a hypothesis $h \in H$ satisfying

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

The probability is taken over the random choice of the examples in S according to $\mathcal{D}$ and the coin tosses of the learner $\mathcal{A}$. If $H \subseteq C$ then $\mathcal{A}$ is called a proper *learner; otherwise, it is called an* improper *learner.*

That is, a PAC learner takes a finite labeled sample S and outputs a hypothesis that should accurately label fresh examples taken from the underlying distribution. A common technique for constructing such learners is to identify a hypothesis h with small *empirical error* on the sample S, and then argue that this h also has small generalization error.

Definition 3.12 (Empirical Error)**.** *For a labeled sample $S = (x_i, y_i)_{i=1}^n$, the* empirical error *of h is*

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

If $\text{error}_S(h) = 0$ we say that h is consistent *with S.*

As an example, let us now construct a simple (proper) PAC learner for the class of threshold functions. The algorithm takes a labeled sample $S = (x_i, y_i)_{i=1}^n$ sampled i.i.d. from an (unknown) distribution $\mathcal{D}$ and labeled by an (unknown) target concept $c_j \in \texttt{THRESH}_X$, where c_j is s.t. $c_j(x) = 1$ iff $x \leq j$. The algorithm returns an arbitrary hypothesis $h = c_\ell \in \texttt{THRESH}_X$ s.t. $\text{error}_S(c_\ell) = 0$. Such a concept exists in $\texttt{THRESH}_X$, as, in particular, this is the case for the target concept c_j.

To see that our algorithm is indeed a PAC learner for the thresholds class, fix a distribution $\mathcal{D}$ on X and a target concept $c_j \in \texttt{THRESH}_X$. Let $L, R \subseteq X$ denote the two intervals of mass α (under $\mathcal{D}$) to the left and to the right of the point j. Observe that a hypothesis from $\texttt{THRESH}_X$ with generalization error bigger than α either errs on every point in L, or errs on every point in R. Thus, assuming that the sample S contains both a point from L and a point from R, every hypothesis with generalization error bigger than α has non-zero empirical error. As we will now see, provided that n is big enough, the probability that the sample S does not include points from L and R is small.

3.2.2 The Vapnik-Chervonenkis Dimension

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

Definition 3.13 ([92]). *Let C be a concept class over a domain X, and let $B = \{b_1, \ldots, b_\ell\} \subseteq X$. The set of all dichotomies (behaviors) on B that are realized by C is*

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

Observe that $\Pi_C(B)$ is a subset of $\{0,1\}^\ell$ (as $c \in C$ maps into $\{0,1\}$). The set of dichotomies $\Pi_C(B)$ can be viewed as the "projection" of C on B. For example, let X be a totally ordered domain, and let $B = \{b_1, \ldots, b_\ell\} \subseteq X$ where $b_1 < b_2 < \cdots < b_\ell$. For the class of threshold functions we have that $\Pi_{\mathtt{THRESH}_X}(B)$ contains all the vectors of the form $(1,1,\ldots,1,0,0,\ldots,0) \in \{0,1\}^\ell$. For the class of point functions we have that $\Pi_{\mathtt{POINT}_X}(B)$ contains all the vectors of the form $(0,0,\ldots,0,1,0,\ldots,0) \in \{0,1\}^\ell$.

Definition 3.14 ([92]). *A set $B \subseteq X$ is* shattered *by C if $\Pi_C(B) = \{0,1\}^\ell$ (where $\ell = |B|$).*

That is, $B \subseteq X$ is shattered by C if C realizes all possible dichotomies over B, i.e., $|\Pi_C(B)| = 2^\ell$. For example, observe that there is no set $B \subseteq X$ of size $|B| \geq 2$ that is shattered by $\mathtt{THRESH}_X$ or by $\mathtt{POINT}_X$.

Definition 3.15 (VC-Dimension [92]). *The* VC-Dimension *of a concept class C (over a domain X), denoted as* $\mathrm{VC}(C)$, *is the cardinality of the largest set $B \subseteq X$ shattered by C. If arbitrarily large finite sets can be shattered by C, then* $\mathrm{VC}(C) = \infty$.

So, $\mathrm{VC}(\mathtt{THRESH}_X) = \mathrm{VC}(\mathtt{POINT}_X) = 1$. In general, observe that as $\Pi_C(B) \leq |C|$, a set B can be shattered only if $|B| \leq \log|C|$ and hence $\mathrm{VC}(C) \leq \log|C|$.

3.2.3 VC Bounds

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

Theorem 3.16 ([50])**.** *Any algorithm for PAC learning a concept class C must have sample complexity $\Omega(\frac{\mathrm{VC}(C)}{\alpha})$, where α is the approximation parameter.*

Theorem 3.17 (VC-Dimension Generalization Bound [19])**.** *Let $\mathcal{D}$ and C be, respectively, a distribution and a concept class over a domain X, and let $c \in C$. For a sample $S = (x_i, c(x_i))_{i=1}^n$ where $n \geq \frac{64\,\mathrm{VC}(C)}{\alpha} \ln(\frac{512}{\alpha\beta})$ and the x_i are drawn i.i.d. from $\mathcal{D}$, it holds that*

$$\Pr\left[\exists\, h \in C:\ \ \mathrm{error}_{\mathcal{D}}(h,c) > \alpha \ \wedge\ \mathrm{error}_S(h) \leq \frac{\alpha}{2}\right] \leq \beta.$$

So, for any concept class C, any algorithm that takes a sample of $n = \Omega_{\alpha,\beta}(\mathrm{VC}(C))$ labeled examples and produces a hypothesis $h \in C$ with small empirical error is a PAC learner for C. In particular, as is common with non-private learners, it suffices to identify a *consistent* hypothesis h with empirical error 0. However, when privacy is introduced, we will be forced to identify a hypothesis with small (but non-zero) empirical error, as privately identifying a consistent hypothesis might be impossible.

Such an algorithm is a PAC learner for C using C (that is, both the target concept and the returned hypotheses are taken from the same concept class C), and, therefore, there always exist a hypothesis $h \in C$ with small empirical error (e.g., the target concept itself). The next theorem handles the agnostic case, in which a learning algorithm for a concept class C uses a hypotheses class H s.t. $C \nsubseteq H$. In particular, given a sample S (labeled by some $c \in C$), a hypothesis with small empirical error might not exist in H.

Theorem 3.18 (VC-Dimension Agnostic Generalization Bound [4, 3])**.** *Let $\mathcal{P}$ be a distribution over $(X \times \{0,1\})$, and let H be a hypothesis class. For a sample $S = (x_i, y_i)_{i=1}^n$ where $n \geq \frac{50\,\mathrm{VC}(H)}{\alpha^2} \ln(\frac{1}{\alpha\beta})$ and $\{(x_i, y_i)\}$ are drawn i.i.d. from $\mathcal{P}$, we have*

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

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

3.3 Generalization Bounds for Points and Thresholds

It turns out that we can already use the simple concept classes of point and threshold functions in order to illustrate an interesting phenomenon. Recall Theorem 3.17 stating that every hypothesis with low empirical error also has low generalization error, provided that the sample size is at least $n \geq O(\frac{1}{\alpha}\mathrm{VC}(C)\log(1/\alpha))$. For some specific cases, such as threshold functions, we can obtain an improved bound without the $\log(1/\alpha)$ factor:

Lemma 3.19 (Generalization Bound for Thresholds, e.g., [69])**.** *Let $\mathcal{D}$ be a distribution over a (totally ordered) domain X, and let $c \in \mathtt{THRESH}_X$. For a sample $S = (x_i, c(x_i))_{i=1}^n$ where $n \geq \frac{8}{\alpha}\ln(\frac{2}{\beta})$ and the x_i are drawn i.i.d. from $\mathcal{D}$, it holds that*

$$\Pr\left[\exists\, h \in \mathtt{THRESH}_X :\ \mathrm{error}_{\mathcal{D}}(h,c) > \alpha \ \wedge\ \mathrm{error}_S(h) \leq \frac{\alpha}{2}\right] \leq \beta.$$

On the other hand, for different concept classes, such as point functions, the $\log(1/\alpha)$ factor is necessary in order to ensure that *all* hypotheses with small empirical error have low generalization error:

Lemma 3.20 ([5, 83])**.** *Let $\alpha < 1$, and let X be a domain s.t. $|X| > \lceil\frac{1}{\alpha}\rceil$. Let n denote the minimal sample size s.t. for every target concept $c \in \mathtt{POINT}_X$ and every distribution $\mathcal{D}$ on X we have that*

$$\Pr\left[\exists\, h \in \mathtt{POINT}_X :\ \mathrm{error}_{\mathcal{D}}(h,c) > \alpha \ \wedge\ \mathrm{error}_S(h) = 0\right] \leq 1/2.$$

The probability is over the sample $S = (x_i, c(x_i))_{i=1}^n$ where the x_i are drawn i.i.d. from $\mathcal{D}$. Then,

$$n = \Omega\left(\frac{1}{\alpha}\log\left(\frac{1}{\alpha}\right)\right).$$

The idea is to consider a uniform distribution on $1/\alpha$ domain points and a target concept that evaluates to zero on all of them. Now, if the sample S does not contain all of those $1/\alpha$ points, then the hypothesis that evaluates to 1 a "missed" point has generalization error α but empirical error 0. Unless $n = \Omega(\frac{1}{\alpha}\log(\frac{1}{\alpha}))$, we are likely to miss at least one such point.

That said, the class $\mathtt{POINT}_X$ *can be PAC learned* using a sample of size $O(1/\alpha)$, e.g., by choosing a *random* consistent hypothesis. It can be easily shown that the number of consis-

tent hypotheses with large empirical error is at most $1/\alpha$, and hence, assuming that $|X|$ is big enough, the probability that a random consistent hypothesis has large generalization error is small. In fact, Hanneke [58] showed that for *every* concept class C there exists a PAC learner whose sample complexity grows as $1/\alpha$. As evident by Lemma 3.20, this cannot be achieved by simply choosing an arbitrary consistent hypothesis.

Theorem 3.21 ([58])**.** *Let C be a concept class. There exists an (α, β)-PAC learner for C with sample size n, where*

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

3.4 Private Learning

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

Definition 3.22 (Private PAC Learning [67])**.** *Let $\mathcal{A}$ be an algorithm that gets an input $S = \{z_1, \ldots, z_n\}$. Algorithm $\mathcal{A}$ is an* $(\alpha, \beta, \varepsilon, \delta)$-PPAC learner *for a concept class C with sample size n using hypothesis class H if*

Privacy. *Algorithm $\mathcal{A}$ is (ε, δ)-differentially private (as in Definition 3.2);*

Utility. *Algorithm $\mathcal{A}$ is an* (α, β)-PAC learner *for C with sample size n using H (as in Definition 3.11).*

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

Note that the utility requirement in the definition is an average-case requirement, as the learner is only required to do well on typical samples (i.e., samples drawn i.i.d. from a distribution $\mathcal{D}$ and correctly labeled by a target concept $c \in C$). In contrast, the privacy requirement is a worst-case requirement, and Inequality (3.1) must hold for every pair of neighboring databases (no matter how they were generated, even if they are not consistent with any concept in C). This worst case requirement is important as otherwise wrong assumptions about how the data is generated might lead to privacy breaches, which are irreversible.

3.5 Basic Differentially Private Mechanisms

3.5.1 The Laplace Mechanism

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

Definition 3.23 (The Laplace Distribution)**.** *A random variable has probability distribution* $\mathrm{Lap}(b)$ *if its probability density function is* $f(x) = \frac{1}{2b}\exp(-\frac{|x|}{b})$*, where* $x \in \mathbb{R}$*.*

Definition 3.24 (Sensitivity)**.** *The* sensitivity *of a function* $f : X^m \to \mathbb{R}^n$ *is the smallest* k *such that for every neighboring* $D, D' \in X^m$*, we have* $\|f(D) - f(D')\|_1 \leq k$*.*

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

Theorem 3.25 (The Laplace mechanism [44])**.** *Let* $f : X^n \to \mathbb{R}^\ell$ *be a* k*-sensitive function. The mechanism* $\mathcal{A}$ *that on input* $D \in X^n$ *adds independently generated noise with distribution* $\mathrm{Lap}(\frac{k}{\varepsilon})$ *to each of the* ℓ *coordinates of* $f(D)$ *preserves* ε*-differential privacy. Moreover,*

$$\Pr\left[\exists i \text{ s.t. } |\mathcal{A}_i(D) - f_i(D)| > \Delta\right] \leq \ell \cdot \exp\left(-\frac{\varepsilon\Delta}{k}\right),$$

where $\mathcal{A}_i(D)$ *and* $f_i(D)$ *are the* i^{th} *coordinates of* $\mathcal{A}(D)$ *and* $f(D)$*.*

3.5.2 The Exponential Mechanism

We next describe the exponential mechanism of McSherry and Talwar [74]. Let X be a domain and H a set of solutions. Given a database $S \in X^*$, the exponential mechanism privately chooses a "good" solution h out of the possible set of solutions H. This "goodness" is quantified using a *quality function* that matches solutions to scores.

Definition 3.26 (Quality function)**.** *A* quality function *is a function* $q : X^* \times H \to \mathbb{R}$ *that maps a database* $S \in X^*$ *and a solution* $h \in H$ *to a real number, identified as the score of the solution* h *w.r.t. the database* S*.*

Given a quality function q and a database S, the goal is to chooses a solution h approximately maximizing $q(S, h)$. The exponential mechanism chooses a solution probabilistically,

where the probability mass that is assigned to each solution h increases exponentially with its quality $q(S,h)$:

The Exponential Mechanism

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

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

Proposition 3.27 (Properties of the exponential mechanism)**.** *(i) The exponential mechanism is ε-differentially private. (ii) Let $\hat{e} \triangleq \max_{f \in H}\{q(S,f)\}$ and $\Delta > 0$. The exponential mechanism outputs a solution h such that $q(S,h) \leq (\hat{e} - \Delta m)$ with probability at most $|H| \cdot \exp(-\varepsilon \Delta m/2)$.*

Kasiviswanathan et al. [67] showed in 2008 that the exponential mechanism can be used as a generic private learner – when used with the quality function $q(S,h) = |\{i : h(x_i) = y_i\}|$ (i.e., the number of points on which h agrees with the sample S), the probability that the exponential mechanism outputs a hypothesis h such that $\text{error}_S(h) > \min_{f \in H}\{\text{error}_S(f)\} + \Delta$ is at most $|H| \cdot \exp(-\varepsilon \Delta m/2)$. This results in a generic private proper-learner for every finite concept class C, with sample complexity $O_{\alpha,\beta,\varepsilon}(\log|C|)$.

3.5.3 Stability and Privacy – $\mathcal{A}_{\text{dist}}$

We restate a simplified variant of algorithm $\mathcal{A}_{\text{dist}}$ by Smith and Thakurta [86], which is an instantiation of the Propose-Test-Release framework [43]. Let $q : X^* \times H \to \mathbb{N}$ be a 1-sensitive quality function over a domain X and a set of solutions H. Given a database $S \in X^*$, the goal is to choose a solution $h \in H$ maximizing $q(S,h)$, under the assumption that the optimal solution h scores much better than any other solution in H.

Algorithm $\mathcal{A}_{\text{dist}}$

Input: parameters ε, δ, a database $S \in X^*$, a 1-sensitive quality function q.

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

Proposition 3.28 (Properties of $\mathcal{A}_{\text{dist}}$ [86])**.** *(i) Algorithm* $\mathcal{A}_{\text{dist}}$ *is* (ε, δ)*-differentially private. (ii) When given an input database* S *for which* $\text{gap} \geq \frac{1}{\varepsilon}\log(\frac{1}{\beta\delta})$*, algorithm* $\mathcal{A}_{\text{dist}}$ *outputs* h_1 *maximizing* $q(h, S)$ *with probability at least* $(1 - \beta)$*.*

3.6 Concentration Bounds

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

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

The first two inequalities are known as the multiplicative Chernoff bounds [28], and the last inequality is known as the Hoeffding bound [62].

Chapter 4

The Generalization Properties of Differential Privacy

By now a rich literature has shown that many learning tasks of interest are compatible with differential privacy. Indeed, one of the main goals in the following chapters is to understand under which conditions we can construct private analogues for existing non-private learning algorithms. In this chapter, however, we study a different connection between differential privacy and learning; namely, that differential privacy *implies* learning. Recall that the general task of computational learning is to identify properties of the underlying distribution, rather than properties of a given sample. Roughly speaking, we show that if a property of a given sample is identified by a differentially private computation, then this property is in fact a property of the underlying distribution. In other words – differential privacy guarantees generalization.

4.1 Main Results

Dwork et al. [40] showed that if a predicate $h : X \to \{0,1\}$ is the result of an (ε, δ)-differentially private computation on a sample S containing i.i.d. elements from a distribution $\mathcal{D}$, then the empirical average $h(S) = \frac{1}{|S|}\sum_{x\in S} h(x)$ and the expectation $h(\mathcal{D}) = \mathbb{E}_{x\sim\mathcal{D}}[h(x)]$ are close to within $O(\varepsilon)$, except with probability at most $O(\delta^{\varepsilon})$. We strengthen this connection and obtain the following theorem. As we will see, our connection between differential privacy and generalization is optimal.

Theorem 4.1. *Let $\varepsilon \in (0, 1/3)$, $\delta \in (0, \varepsilon/4)$, and $n \geq \frac{1}{\varepsilon^2}\log(\frac{2\varepsilon}{\delta})$. Let $\mathcal{A} : X^n \to 2^X$ be an (ε, δ)-differentially private algorithm that operates on a database of size n and outputs a predicate*

$h : X \to \{0,1\}$. *Let $\mathcal{D}$ be a distribution over X, let S be a database containing n i.i.d. elements from $\mathcal{D}$, and let $h \leftarrow \mathcal{A}(S)$. Then*

$$\Pr_{\substack{S \sim \mathcal{D} \\ h \leftarrow \mathcal{A}(S)}} \left[|h(S) - h(\mathcal{D})| \geq 10\varepsilon \right] < \frac{\delta}{\varepsilon}$$

where $h(S)$ is the empirical average of h on S, and $h(\mathcal{D})$ is the expectation of h over $\mathcal{D}$.

In words, if $\mathcal{A}$ is a differentially private algorithm operating on a database containing n i.i.d. samples from $\mathcal{D}$, then $\mathcal{A}$ cannot (with significant probability) identify a predicate that behaves differently on the sample S and on $\mathcal{D}$.

A matching bound. We show that Theorem 4.1 is tight, i.e., there exists an (ε, δ) differentially private computation that achieves error ε with probability $\Theta(\delta/\varepsilon)$.

4.1.1 Proof Outline

We begin with the following expectation bound. Consider an algorithm $\mathcal{B}$ operating on T *sub-databases* $\vec{S} = (S_1, S_2, \ldots, S_T)$, where every S_t contains i.i.d. samples from $\mathcal{D}$. Algorithm $\mathcal{B}$ outputs a predicate h and an index $1 \leq t \leq T$, and succeeds if $h(\mathcal{D})$ is far from $h(S_t)$. That is, algorithm $\mathcal{B}$ tries to identify a sub-database S_t and a predicate h that behaves differently on S_t and on $\mathcal{D}$. We first show that no differentially private algorithm can succeed in this task in expectation. That is, if $\mathcal{B}$ is differentially private, then the expectations $\mathbb{E}_{\vec{S},\mathcal{A}}[h(\mathcal{D})]$ and $\mathbb{E}_{\vec{S},\mathcal{A}}[h(S_t)]$ are close.

This expectation bound for algorithms that operate on T sub-databases is then transformed into a high probability bound on private algorithms that operate on a single database: Assume the existence of a differentially private algorithm $\mathcal{A}$ that operates on a database S containing i.i.d. samples from $\mathcal{D}$ and, with probability β, outputs a predicate h s.t. $h(S)$ is far from $h(\mathcal{D})$. We can use $\mathcal{A}$ to construct an algorithm $\mathcal{B}$ operating on $T \approx \frac{1}{\beta}$ sub-databases that contradicts our expectation bound. Specifically, algorithm $\mathcal{B}$ applies $\mathcal{A}$ on every sub-database, and obtains T predicates $H = \{h_1, \ldots, h_T\}$. Since $T \approx \frac{1}{\beta}$, w.h.p. at least one of the h_t's behaves differently on S_t and on $\mathcal{D}$, and $\mathcal{B}$ can identify such an $h_t \in H$ using standard differentially private tools. This will contradict our expectation bound.

4.1.2 Additional Results

As we explained in Chapter 1, following Dwork et al., generalization bounds for differentially private algorithms can be translated into upper bounds on the number of samples n needed to accurately answer adaptively chosen *statistical queries* w.r.t. to the underlying distribution. In this work we prove the first upper bounds on the number of samples required to answer more general families of queries. These include arbitrary *low-sensitivity queries* and an important class of *optimization queries* (alternatively, *risk minimization queries*). Those results are omitted from the main body of this thesis; they appear in the full version of this work [6].

4.2 From Expectation to High Probability Bounds

Given a predicate $h : X \to \{0,1\}$, a distribution $\mathcal{D}$ over X, and a sample $S \in X^n$, we denote the expectation of h over $\mathcal{D}$, and the empirical average of h on S as:

$$h(\mathcal{D}) = \mathop{\mathbb{E}}_{x\sim\mathcal{D}}[h(x)] \qquad \text{and} \qquad h(S) = \frac{1}{n}\sum_{x\in S} h(x).$$

Let $\mathcal{A}$ be an (ε,δ)-differentially private algorithm that outputs a predicate, and consider a database S sampled i.i.d. from some distribution $\mathcal{D}$. Our goal is to show that if $h \leftarrow \mathcal{A}(S)$, then *with high probability* $h(\mathcal{D})$ is close to the empirical average $h(S)$, where the probability is over the choice of S and the random coins of $\mathcal{A}$. This is in spite of h being chosen based on S.

Similarly to [40], we begin by showing that if h is the result of a differentially private computation on a database S sampled i.i.d. from a distribution $\mathcal{D}$, then $h(S)$ and $h(\mathcal{D})$ are close in expectation. In order to later transform this expectation bound into a high probability bound, we allow $\mathcal{A}$ to operate on *several* sub-databases $S_1,\ldots,S_T$ and analyze its (in)ability to identify one sub-database S_t and a predicate h that behaves differently on S_t and on $\mathcal{D}$.

Lemma 4.2 (Expectation bound)**.** *Let $\mathcal{B} : (X^n)^T \to 2^X \times \{1,2,\ldots,T\}$ be an (ε,δ)-differentially private algorithm that operates on T sub-databases and outputs a predicate $h : X \to \{0,1\}$ and*

an index $t \in \{1, 2, \ldots, T\}$. Let $\mathcal{D}$ be a distribution over X, let $\vec{S} = (S_1, \ldots, S_T)$ where every S_j is a database containing n i.i.d. elements from $\mathcal{D}$, and let $(h, t) \leftarrow \mathcal{B}\left(\vec{S}\right)$. Then

$$\left| \mathop{\mathbb{E}}_{\vec{S}\sim\mathcal{D},\ (h,t)\leftarrow\mathcal{B}\left(\vec{S}\right)} \left[h(\mathcal{D}) - h(S_t) \right] \right| \leq e^{\varepsilon} - 1 + T\delta.$$

Proof. We denote the i-th element of the j-th sample of $\vec{S}$ as $x_{j,i}$. That is, $\vec{S} = (S_1, \ldots, S_T)$, where each sample $S_j = (x_{j,1}, \ldots, x_{j,n})$. We can now calculate

$$\begin{aligned}
\mathop{\mathbb{E}}_{\vec{S}\sim\mathcal{D}} \left[\mathop{\mathbb{E}}_{(h,t)\leftarrow\mathcal{B}\left(\vec{S}\right)} [h(S_t)] \right] &= \mathop{\mathbb{E}}_{\vec{S}\sim\mathcal{D}} \left[\mathop{\mathbb{E}}_{(h,t)\leftarrow\mathcal{B}\left(\vec{S}\right)} \left[\mathop{\mathbb{E}}_{i\sim[n]} [h(x_{t,i})] \right] \right] \\
&= \mathop{\mathbb{E}}_{i\sim[n]} \left[\mathop{\mathbb{E}}_{\vec{S}\sim\mathcal{D}} \left[\mathop{\mathbb{E}}_{(h,t)\leftarrow\mathcal{B}\left(\vec{S}\right)} [h(x_{t,i})] \right] \right] \\
&= \mathop{\mathbb{E}}_{i\sim[n]} \left[\mathop{\mathbb{E}}_{\vec{S}\sim\mathcal{D}} \left[\Pr_{(h,t)\leftarrow\mathcal{B}\left(\vec{S}\right)} [h(x_{t,i}) = 1] \right] \right] \\
&= \mathop{\mathbb{E}}_{i\sim[n]} \left[\mathop{\mathbb{E}}_{\vec{S}\sim\mathcal{D}} \left[\sum_{m=1}^{T} \Pr_{(h,t)\leftarrow\mathcal{B}\left(\vec{S}\right)} [h(x_{m,i}) = 1 \text{ and } t = m] \right] \right] \\
&= \mathop{\mathbb{E}}_{i\sim[n]} \left[\mathop{\mathbb{E}}_{\vec{S}\sim\mathcal{D}} \left[\mathop{\mathbb{E}}_{z\sim\mathcal{D}} \left[\sum_{m=1}^{T} \Pr_{(h,t)\leftarrow\mathcal{B}\left(\vec{S}\right)} [h(x_{m,i}) = 1 \text{ and } t = m] \right] \right] \right]. \qquad (4.1)
\end{aligned}$$

Given a multi-sample $\vec{S}$, a single element $z \in X$, and a pair of indices $(m, i) \in [T] \times [n]$, we define $\vec{S}^{(m,i):z}$ to be the same as $\vec{S}$, except that the i-th element of the m-th sample of $\vec{S}$ is replaced with z. Note that $x_{m,i}$ is still the element from $\vec{S}$ (and not from $\vec{S}^{(m,i):z}$). With this notation, by the differential privacy of $\mathcal{B}$, we have that

$$\begin{aligned}
(4.1) \quad &\leq \mathop{\mathbb{E}}_{i\sim[n]} \left[\mathop{\mathbb{E}}_{\vec{S}\sim\mathcal{D}} \left[\mathop{\mathbb{E}}_{z\sim\mathcal{D}} \left[\sum_{m=1}^{T} e^{\varepsilon} \cdot \Pr_{(h,t)\leftarrow\mathcal{B}\left(\vec{S}^{(m,i):z}\right)} [h(x_{m,i}) = 1 \text{ and } t = m] + \delta \right] \right] \right] \\
&= T\delta + e^{\varepsilon} \cdot \sum_{m=1}^{T} \mathop{\mathbb{E}}_{i\sim[n]} \left[\mathop{\mathbb{E}}_{\vec{S}\sim\mathcal{D}} \left[\mathop{\mathbb{E}}_{z\sim\mathcal{D}} \left[\Pr_{(h,t)\leftarrow\mathcal{B}\left(\vec{S}^{(m,i):z}\right)} [h(x_{m,i}) = 1 \text{ and } t = m] \right] \right] \right]. \qquad (4.2)
\end{aligned}$$

Now note that every $\vec{S}^{(m,i):z}$ above contains i.i.d. samples from $\mathcal{D}$, and that $x_{m,i}$ is independent of $\vec{S}^{(m,i):z}$. The pairs $(x_{m,i}, \vec{S}^{(m,i):z})$ and $(z, \vec{S})$ are, therefore, identically distributed. Hence,

$$
\begin{aligned}
(4.2) \quad &= T\delta + e^{\varepsilon} \cdot \sum_{m=1}^{T} \mathop{\mathbb{E}}_{i\sim[n]} \left[\mathop{\mathbb{E}}_{\vec{S}\sim\mathcal{D}} \left[\mathop{\mathbb{E}}_{z\sim\mathcal{D}} \left[\Pr_{(h,t)\leftarrow\mathcal{B}\left(\vec{S}\right)} [h(z) = 1 \text{ and } t = m] \right] \right] \right] \\
&= T\delta + e^{\varepsilon} \mathop{\mathbb{E}}_{\vec{S}\sim\mathcal{D}} \left[\mathop{\mathbb{E}}_{z\sim\mathcal{D}} \left[\Pr_{(h,t)\leftarrow\mathcal{B}\left(\vec{S}\right)} [h(z) = 1] \right] \right] \\
&= T\delta + e^{\varepsilon} \mathop{\mathbb{E}}_{\vec{S}\sim\mathcal{D}} \left[\mathop{\mathbb{E}}_{z\sim\mathcal{D}} \left[\mathop{\mathbb{E}}_{(h,t)\leftarrow\mathcal{B}\left(\vec{S}\right)} [h(z)] \right] \right] \\
&= T\delta + e^{\varepsilon} \mathop{\mathbb{E}}_{\vec{S}\sim\mathcal{D}} \left[\mathop{\mathbb{E}}_{(h,t)\leftarrow\mathcal{B}\left(\vec{S}\right)} \left[\mathop{\mathbb{E}}_{z\sim\mathcal{D}} [h(z)] \right] \right] \\
&= T\delta + e^{\varepsilon} \cdot \mathop{\mathbb{E}}_{\vec{S}\sim\mathcal{D}} \left[\mathop{\mathbb{E}}_{(h,t)\leftarrow\mathcal{B}\left(\vec{S}\right)} [h(\mathcal{D})] \right] \\
&\leq T\delta + e^{\varepsilon} - 1 + \mathop{\mathbb{E}}_{\vec{S}\sim\mathcal{D}} \left[\mathop{\mathbb{E}}_{(h,t)\leftarrow\mathcal{B}\left(\vec{S}\right)} [h(\mathcal{D})] \right].
\end{aligned}
$$

(since $h(\mathcal{D}) \in [0,1]$, and since $e^{\varepsilon} \cdot y \leq e^{\varepsilon} - 1 + y$ for every $y \leq 1$ and $\varepsilon \geq 0$.)

An identical argument shows that

$$
\mathop{\mathbb{E}}_{\substack{\vec{S}\sim\mathcal{D} \\ (h,t)\leftarrow\mathcal{B}\left(\vec{S}\right)}} \left[h(S_t) \right] \geq -T\delta + e^{-\varepsilon} - 1 + \mathop{\mathbb{E}}_{\substack{\vec{S}\sim\mathcal{D} \\ (h,t)\leftarrow\mathcal{B}\left(\vec{S}\right)}} \left[h(\mathcal{D}) \right].
$$

□

We now transform the above expectation bound for private algorithms that operate on T databases into a *high probability bound* for private algorithms that operate on a single database. In fact, we will allow our private algorithm to output not 1 but k predicates, and show that w.h.p. *all* of them behave similarly on the input sample and on the underlying distribution. This relaxation will be helpful in Section 4.5, where we use the connection between differential privacy and generalization to accurately answer adaptively chosen queries.

Theorem 4.3 (High probability bound)**.** *Let $\varepsilon \in (0,1/3)$, $\delta \in (0,\varepsilon/4)$, $k \in \mathbb{N}$, and $n \geq \frac{1}{\varepsilon^2}\log(\frac{2\varepsilon k}{\delta})$. Let $\mathcal{A} : X^n \to (2^X)^k$ be an (ε,δ)-differentially private algorithm that operates on a database of size n and outputs k predicates $h_1,\ldots,h_k : X \to \{0,1\}$. Let $\mathcal{D}$ be a distribution over X, let S be a database containing n i.i.d. elements from $\mathcal{D}$, and let $(h_1,\ldots,h_k) \leftarrow \mathcal{A}(S)$. Then*

$$\Pr_{\substack{S\sim\mathcal{D}\\(h_1,\ldots,h_k)\leftarrow\mathcal{A}(S)}}\left[\max_{1\leq i\leq k}|h_i(S)-h_i(\mathcal{D})| \geq 10\varepsilon\right] < \frac{\delta}{\varepsilon}.$$

Proof. Fix a distribution $\mathcal{D}$ on X. Assume towards contradiction that with probability at least δ/ε algorithm $\mathcal{A}$ outputs k predicates $h_1,\ldots,h_k$ s.t. $\max_i |h_i(S)-h_i(\mathcal{D})| \geq 10\varepsilon$. We now use $\mathcal{A}$ and $\mathcal{D}$ to construct the following algorithm $\mathcal{B}$ that contradicts Lemma 4.2. We remark that algorithm $\mathcal{B}$ "knows" the distribution $\mathcal{D}$. This will still lead to a contradiction because the expectation bound of Lemma 4.2 holds for *every* differentially private algorithm and *every* underlying distribution.

Algorithm $\mathcal{B}$

Input: T databases of size n each: $\vec{S} = (S_1,\ldots,S_T)$, where $T \triangleq \lfloor \varepsilon/\delta \rfloor$.

1. Set $F = \emptyset$.
2. For $t = 1,\ldots,T$:
   (a) Let $(h_1^t,\ldots,h_k^t) \leftarrow \mathcal{A}(S_t)$.
   (b) Set $F = F \cup \left\{\left(h_1^t,t\right),\ldots,\left(h_k^t,t\right),\left(\overline{h_1^t},t\right),\ldots,\left(\overline{h_k^t},t\right)\right\}$, where a predicate $\overline{h}$ is defined by $\overline{h}(x) = 1 - h(x)$.
3. Sample (h^*,t^*) from F with probability proportional to $\exp\left(\frac{\varepsilon n}{2}\left(h^*(S_{t^*}) - h^*(\mathcal{D})\right)\right)$.

Output: (h^*,t^*).

Observe that $\mathcal{B}$ only accesses its input through $\mathcal{A}$ (which is (ε,δ)-differentially private) and the exponential mechanism (which is $(\varepsilon,0)$-differentially private). Thus, by composition and post-processing, $\mathcal{B}$ is $(2\varepsilon,\delta)$-differentially private.

Now consider applying $\mathcal{B}$ on databases $\vec{S} = (S_1,\ldots,S_T)$ containing i.i.d. samples from $\mathcal{D}$. By our assumption on $\mathcal{A}$, for every t we have that $\max_{1\leq i\leq k}\left|h_i^t(S_t) - h_i^t(\mathcal{D})\right| \geq 10\varepsilon$ with

probability at least δ/ε. By our choice of $T = \lfloor \varepsilon/\delta \rfloor$, we therefore get

$$\Pr_{\substack{\vec{S}\sim\mathcal{D}\\ \mathcal{B}\left(\vec{S}\right)}}\left[\max_{\substack{t\in[T]\\ i\in[k]}}\left|h_i^t(S_t) - h_i^t(\mathcal{D})\right| \geq 10\varepsilon\right] \geq 1 - \left(1 - \frac{\delta}{\varepsilon}\right)^T \geq \frac{1}{2}.$$

The probability is taken over the random choice of the examples in $\vec{S}$ according to $\mathcal{D}$ and the generation of the predicates h_i^t according to $\mathcal{B}\left(\vec{S}\right)$. Thus, by Markov's inequality,

$$\mathop{\mathbb{E}}_{\substack{\vec{S}\sim\mathcal{D}\\ \mathcal{B}\left(\vec{S}\right)}}\left[\max_{\substack{t\in[T]\\ i\in[k]}}\left|h_i^t(S_t) - h_i^t(\mathcal{D})\right|\right] \geq 5\varepsilon.$$

Recall that the set F (constructed in step 2 of algorithm $\mathcal{B}$) contains predicates and their negations, and hence,

$$\mathop{\mathbb{E}}_{\substack{\vec{S}\sim\mathcal{D}\\ \mathcal{B}\left(\vec{S}\right)}}\left[\max_{(h,t)\in F}\{h(S_t) - h(\mathcal{D})\}\right] = \mathop{\mathbb{E}}_{\substack{\vec{S}\sim\mathcal{D}\\ \mathcal{B}\left(\vec{S}\right)}}\left[\max_{\substack{t\in[T]\\ i\in[k]}}\left|h_i^t(S_t) - h_i^t(\mathcal{D})\right|\right] \geq 5\varepsilon. \tag{4.3}$$

So, in expectation, the set F contains a pair (h,t) with large difference $h(S_t) - h(\mathcal{D})$. In order to contradict the expectation bound of Lemma 4.2, we need to show that this is also the case for the pair (h^*, t^*), which is sampled from F using the exponential mechanism. To that end, we now use the following technical claim, stating that the expected quality of a solution sampled using the exponential mechanism is high.

Claim 4.4. *Let F be a finite set, $f : F \to \mathbb{R}$ a function, and $\eta > 0$. Define a random variable Y on F by $\Pr[Y = y] = \exp(\eta f(y))/C$, where $C = \sum_{y\in F}\exp(\eta f(y))$. Then $\mathbb{E}[f(Y)] \geq \max_{y\in F} f(y) - \frac{1}{\eta}\log|F|$.*

We can apply Claim 4.4 with $f(h,t) = h(S_t) - h(\mathcal{D})$ and $\eta = \frac{\varepsilon n}{2}$ to get

$$\mathop{\mathbb{E}}_{(h^*,t^*)\in_R F}\left[h^*(S_{t^*}) - h^*(\mathcal{D})\right] \geq \max_{(h,t)\in F}\{h(S_t) - h(\mathcal{D})\} - \frac{2}{\varepsilon n}\log(2Tk). \tag{4.4}$$

Taking the expectation also over $\vec{S} \sim \mathcal{D}$ and $\mathcal{B}(\vec{S})$ we get that

$$\begin{aligned}\mathop{\mathbb{E}}_{\substack{\vec{S}\sim\mathcal{D}\\ \mathcal{B}(\vec{S})}}\left[h^*(S_{t^*}) - h^*(\mathcal{D})\right] &\geq \mathop{\mathbb{E}}_{\substack{\vec{S}\sim\mathcal{D}\\ \mathcal{B}(\vec{S})}}\left[\max_{(h,t)\in F}\{h(S_t) - h(\mathcal{D})\}\right] - \frac{2}{\varepsilon n}\log(2Tk) \\ &\geq 5\varepsilon - \frac{2}{\varepsilon n}\log(2\varepsilon k/\delta).\end{aligned}$$

This contradicts Lemma 4.2 whenever $n \geq \frac{1}{\varepsilon^2}\log(2\varepsilon k/\delta)$.

It remains to prove Claim 4.4.

Proof of Claim 4.4. We have

$$f(x) = \frac{1}{\eta}\Big(\log C + \log \Pr[X = x]\Big).$$

Thus

$$\begin{aligned}\mathbb{E}[f(X)] &= \sum_{x\in F} \Pr[X = x] f(x) \\ &= \sum_{x\in F} \Pr[X = x] \frac{1}{\eta}\Big(\log C + \log \Pr[X = x]\Big) \\ &= \frac{1}{\eta}\left(\log C - H(X)\right),\end{aligned}$$

where $H(X)$ is the Shannon entropy of the distribution of X. In particular,

$$H(X) \leq \log|\text{support}(X)| = \log|F|,$$

as the uniform distribution maximizes entropy. Moreover, $C \geq \max_{x\in F} e^{\eta f(x)}$, whence $\frac{1}{\eta}\log C \geq \max_{x\in F} f(x)$. Claim 4.4 now follows from these two inequalities. □

This completes the proof of Theorem 4.3. □

4.3 Tightness of Our Results

In the previous section we showed that (ε, δ)-differential privacy guarantees $O(\varepsilon)$ accuracy with probability $1 - O(\delta/\varepsilon)$. It would be tempting to guess that (ε, δ)-differential privacy should guarantee $O(\varepsilon)$ accuracy with probability $1 - O(\delta)$. As we will now see, this is not the case, and our results are tight.

Theorem 4.5. *Let $\mathcal{U}$ be the uniform distribution over $[0,1]$. For every $\alpha > \delta$ there exists a $(0,\delta)$-differentially private algorithm $\mathcal{A}$ such that the following holds. If S is a database containing $n \geq \frac{1}{\alpha}$ i.i.d. samples from $\mathcal{U}$, and if $h \leftarrow \mathcal{A}(S)$ then*

$$\Pr[h(S) \geq h(\mathcal{U}) + \alpha] \geq \frac{\delta}{2\alpha}.$$

Proof. Consider the following simple algorithm, denoted as $\mathcal{B}$. On input a database S, output S with probability δ, and otherwise output the empty database. Clearly, $\mathcal{B}$ is $(0,\delta)$-differentially private. Now construct the following algorithm $\mathcal{A}$.

Algorithm $\mathcal{A}$

Input: $\frac{1}{\alpha}$ databases of size αn each: $\vec{S} = (S_1, \ldots, S_{1/\alpha})$.

1. For $1 \leq i \leq \frac{1}{\alpha}$ let $\hat{S}_i = \mathcal{B}(S_i)$.
2. Return $h : [0,1] \to \{0,1\}$ where $h(x) = 1$ iff $\exists i$ s.t. $x \in \hat{S}_i$.

As $\mathcal{B}$ is $(0,\delta)$-differentially private, and as $\mathcal{A}$ only applies $\mathcal{B}$ on disjoint databases, we get that $\mathcal{A}$ is also $(0,\delta)$-differentially private.

Suppose $\vec{S} = (S_1, \ldots, S_{1/\alpha})$ contains i.i.d. samples from $\mathcal{U}$, and let $h \leftarrow \mathcal{A}(\vec{S})$. Observe that h evaluates to 1 only on a finite number of points from $[0,1]$, and hence, we have that $h(\mathcal{U}) = \mathbb{E}_{x \sim \mathcal{U}}[h(x)] = 0$. Next note that $h(\vec{S}) = \alpha \cdot |\{i : \hat{S}_i = S_i\}|$. Therefore, if there exists an i s.t. $\hat{S}_i = S_i$ then $h(\vec{S}) \geq h(\mathcal{U}) + \alpha$. The probability that this is not the case is at most

$$(1-\delta)^{1/\alpha} \leq e^{-\delta/\alpha} \leq 1 - \frac{\delta}{2\alpha},$$

and thus, with probability at least $\frac{\delta}{2\alpha}$, algorithm $\mathcal{A}$ outputs a predicate h s.t. $h(S) \geq h(\mathcal{U}) + \alpha$. □

In particular, using Theorem 4.5 with $\alpha = \varepsilon$ shows that the confidence parameter in Theorem 4.3 is tight.

4.4 Beyond Binary Functions

Recall that the task of learning algorithms is to predict the classification of unseen examples. Our work, as well as most of the research on private learning, is mainly focused on *binary classification*, where examples are labeled by either 0 or 1. In practice, however, there might be more than two possible labels. For example, in medical diagnosis, we might be interested in mapping patients to one out of a hundred possible diseases.

In this section we show that the generalization properties of differential privacy are not limited to binary functions, and extend to multi-class classification:

Definition 4.6. *Let X be a data universe, let Y be a class of possible labels, and let $\mathcal{P}$ be a distribution over $X \times Y$. The* generalization error *of a hypothesis $h : X \to Y$ w.r.t. $\mathcal{P}$ is defined by* $\text{error}_{\mathcal{P}}(h) = \Pr_{(x,y)\sim\mathcal{P}}[h(x) \neq y]$.

The empirical error *of a hypothesis $h : X \to Y$ on a labeled sample $S = ((x_1, y_1), \ldots, (x_n, y_n)) \in (X \times Y)^n$ is* $\text{error}_S(h) = \frac{1}{n}|\{i : h(x_i) \neq y_i\}|$.

We now restate Theorem 4.3 for algorithms that output functions mapping X to Y.

Theorem 4.7. *Let $\varepsilon \in (0, 1/3)$, $\delta \in (0, \varepsilon/4)$, and $n \geq \frac{1}{\varepsilon^2}\log(\frac{2\varepsilon}{\delta})$. Let $\mathcal{A} : (X \times Y)^n \to Y^X$ be an (ε, δ)-differentially private algorithm that operates on a labeled database of size n and outputs a function $h : X \to Y$. Let $\mathcal{P}$ be a distribution over $X \times Y$, let S be a database containing n i.i.d. elements from $\mathcal{P}$, and let $h \leftarrow \mathcal{A}(S)$. Then*

$$\Pr_{\substack{S\sim\mathcal{P} \\ h\leftarrow\mathcal{A}(S)}}[|\text{error}_S(h) - \text{error}_{\mathcal{P}}(h)| \geq 10\varepsilon] < \frac{\delta}{\varepsilon}.$$

Observe that since algorithm $\mathcal{A}$ in the above theorem outputs a non-binary function, we cannot apply Theorem 4.3 directly to obtain generalization guarantees for it. We address this by post-processing $\mathcal{A}$'s output, and transforming the returned hypothesis h into the *error function* associated with it, which is a binary function. If we assume that h is accurate on the sample but not on the underlying distribution, then the error function would be a

privately computed binary function, that behaves very differently on the sample and the distribution. This will contradict Theorem 4.3

Proof of Theorem 4.7. Fix a distribution $\mathcal{P}$ on $X \times Y$. Assume towards contradiction that with probability at least δ/ε algorithm $\mathcal{A}$ outputs a predicate h s.t. $|\text{error}_S(h) - \text{error}_{\mathcal{P}}(h)| \geq 10\varepsilon$. Consider the following algorithm, denoted as $\mathcal{B}$. Given a labeled database $S \in (X \times Y)^n$, apply $\mathcal{A}(S)$ to obtain a hypothesis $h : X \to Y$, and output $f : (X \times Y) \to \{0,1\}$ where $f(x,y) = 1$ iff $h(x) \neq y$. Observe that $\mathcal{B}$ only post-processes the output of $\mathcal{A}$, and thus, $\mathcal{B}$ is (ε, δ)-differentially private. Now consider applying $\mathcal{B}$ on a database $S \in (X \times Y)^n$ sampled i.i.d. from $\mathcal{P}$. By our assumption on $\mathcal{A}$, with probability at least δ/ε we have that

$$\begin{aligned} 10\varepsilon &\leq |\text{error}_S(h) - \text{error}_{\mathcal{P}}(h)| \\ &= \left| \frac{1}{n} \sum_{i=1}^{n} \mathbb{1}[h(x_i) \neq y_i] - \Pr_{(x,y)\sim\mathcal{P}}[h(x) \neq y] \right| \\ &= |f(S) - f(\mathcal{P})|. \end{aligned}$$

This contradicts Theorem 4.3. □

4.5 Answering Adaptively Chosen Statistical Queries

Following Dwork et al. [40], our high probability generalization bound (Theorem 4.3) yields new upper bounds on the number of samples n needed to accurately answer adaptively chosen statistical queries w.r.t. to the underlying distribution. We now give the details.

4.5.1 Definitions

Given a distribution $\mathcal{D}$ over a data universe X, or a sample $S = (x_1, \ldots, x_n) \in X^n$, we would like to answer *statistical queries* about $\mathcal{D}$ or about S. These queries are specified by a function $q : X \to \{0,1\}$, and the error of an answer a to a statistical query q with respect to $\mathcal{D}$ or S is defined to be

$$\text{err}_S(q,a) = a - q(S) \qquad \text{and} \qquad \text{err}_{\mathcal{D}}(q,a) = a - q(\mathcal{D}).$$

Our goal is to design a *mechanism* $\mathcal{M}$ that answers queries on $\mathcal{D}$ using only independent samples $x_1,\ldots,x_n$ from $\mathcal{D}$. Our focus is the case where the queries are chosen adaptively and adversarially.

Specifically, $\mathcal{M}$ is a stateful algorithm that holds a collection of samples $x_1,\ldots,x_n \in X$, takes a statistical query q as input, and returns an answer a. We require that when $x_1,\ldots,x_n$ are independent samples from $\mathcal{D}$, the answer a is close to $q(\mathcal{D})$. Moreover we require that this condition holds for every query in an adaptively chosen sequence $q_1,\ldots,q_k$. Formally, we define an accuracy game $\mathrm{Acc}_{n,k}[\mathcal{M},\mathbb{A}]$ between a mechanism $\mathcal{M}$ and a stateful *data analyst* $\mathbb{A}$ in Figure 4.1.

$\mathbb{A}$ chooses a distribution $\mathcal{D}$ over X.
Sample $x_1,\ldots,x_n$ i.i.d. from $\mathcal{D}$, and let $S = (x_1,\ldots,x_n)$. (Note that $\mathcal{A}$ does not know S.)

For $j = 1,\ldots,k$
  $\mathbb{A}$ outputs a statistical query $q_j : X \to \{0,1\}$.
  $\mathcal{M}(S, q_j)$ outputs a_j.
  (As $\mathbb{A}$ and $\mathcal{M}$ are stateful, q_j and a_j may depend on the history $q_1, a_1, \ldots, q_{j-1}, a_{j-1}$.)

Figure 4.1: The Accuracy Game $\mathrm{Acc}_{n,k}[\mathcal{M},\mathbb{A}]$.

Definition 4.8 (Distribution Accuracy)**.** *A mechanism $\mathcal{M}$ is (α,β)-*accurate with respect to the underlying distribution for k adaptively chosen statistical queries given n samples in X *if for every adversary $\mathbb{A}$,*

$$\Pr_{\mathrm{Acc}_{n,k}[\mathcal{M},\mathbb{A}]}\left[\max_{j\in[k]} \left|\mathrm{err}_{\mathcal{D}}\left(q_j, a_j\right)\right| \le \alpha\right] \ge 1-\beta.$$

We will also use a definition of accuracy relative to the sample given to the mechanism, described in Figure 4.2.

$\mathbb{A}$ chooses $S = (x_1, \ldots, x_n) \in X^n$.

For $j = 1, \ldots, k$

$\mathbb{A}$ outputs a statistical query q_j.

$\mathcal{M}(S, q_j)$ outputs a_j.

(q_j and a_j may depend on the history $q_1, a_1, \ldots, q_{j-1}, a_{j-1}$ and on S.)

Figure 4.2: The Sample Accuracy Game $\mathsf{SampAcc}_{n,k}[\mathcal{M}, \mathbb{A}]$.

Definition 4.9 (Sample Accuracy)**.** *A mechanism $\mathcal{M}$ is* (α, β)-accurate with respect to samples of size n from X for k adaptively chosen statistical queries *if for every adversary* $\mathbb{A}$,

$$\Pr_{\mathsf{SampAcc}_{n,k}[\mathcal{M},\mathbb{A}]}\left[\max_{j\in[k]} \quad \left|\mathrm{err}_S\left(q_j, a_j\right)\right| \leq \alpha\right] \geq 1 - \beta.$$

4.5.2 From Differential Privacy and Sample-Accuracy to Distribution-Accuracy

Let $\mathcal{M}$ be an (ε, δ)-differentially private mechanism that is (α, β)-accurate with respect to its sample for k adaptively chosen statistical queries. Assume that $\mathcal{M}$ holds a database S sampled i.i.d. from a distribution $\mathcal{D}$, and consider an interaction between $\mathcal{M}$ and a data analyst $\mathbb{A}$. For k rounds, the analyst specifies a statistical query q_i and receives an answer a_i. By the sample-accuracy of $\mathcal{M}$, with probability $(1 - \beta)$, for every i we have that

$$|a_i - q_i(S)| \leq \alpha.$$

Dwork et al. [40] observed that as the analyst only interacts with S through $\mathcal{M}$, the analyst $\mathbb{A}$ can be described as a post-processing of $\mathcal{M}$'s answers. As differential privacy is immune to post-processing, the *queries* chosen by $\mathbb{A}$ are the result of an (ε, δ)-differentially private computation on S. We can therefore think of the mechanism $\mathcal{M}$ and the analyst $\mathbb{A}$ as a *single* differentially private algorithm $\mathcal{A}$ that operates on a random sample S and outputs k queries.

We can now apply our results from Section 4.2 to claim that w.h.p. the empirical average of those queries must be close to their expectation. Specifically, by Theorem 4.3,

with probability $(1-\delta/\varepsilon)$ for every query q_i it holds that

$$|q_i(S) - q_i(\mathcal{D})| \leq O(\varepsilon).$$

Therefore, with probability $(1-\beta-\delta/\varepsilon)$, for every i we have that $|a_i - q_i(\mathcal{D})| \leq O(\alpha+\varepsilon)$ by the triangle inequality. We get the following corollary:

Corollary 4.10. *Let $\mathcal{M}$ be an (ε,δ)-differentially private mechanism that is (α,β)-accurate with respect to a sample of size n for k adaptively chosen statistical queries. Then $\mathcal{M}$ is also $(\alpha+10\varepsilon,\beta+\delta/\varepsilon)$-accurate with respect to the population, provided that $n \geq \frac{1}{\varepsilon^2}\log(\frac{2\varepsilon k}{\delta})$.*

We now instantiate known differentially private mechanisms with the above corollary to obtain mechanisms that provide strong error guarantees with high probability for adaptively chosen statistical queries.

Corollary 4.11 (Theorem 4.3 and [44, 84])**.** *There is a mechanism $\mathcal{M}$ that is (α,β)-accurate with respect to the underlying distribution for k adaptively chosen statistical queries given n samples from X for*

$$n \geq O\left(\frac{\sqrt{k\cdot\log\log k}\cdot\log^{3/2}\left(\frac{1}{\alpha\beta}\right)}{\alpha^2}\right).$$

The mechanism runs in time $\mathrm{poly}(n,\log|X|,\log(1/\beta))$ *per query.*

Corollary 4.12 (Theorem 4.3 and [59])**.** *There is a mechanism $\mathcal{M}$ that is (α,β)-accurate with respect to the underlying distribution for k adaptively chosen statistical queries given n samples from X for*

$$n = O\left(\frac{\sqrt{\log|X|}\cdot\log k\cdot\log^{3/2}\left(\frac{1}{\alpha\beta}\right)}{\alpha^3}\right).$$

The mechanism runs in time $\mathrm{poly}(n,|X|)$ *per query.*

Chapter 5

Characterizing the Sample Complexity of Pure-Private Learners

We give a combinatorial characterization of the sample size sufficient and necessary to learn a class of concepts under pure-differential privacy. This characterization is analogous to the well-known characterization of the sample complexity of non-private learning in terms of the VC dimension of the concept class. We introduce the notion of *probabilistic representation* of a concept class, and our new complexity measure RepDim corresponds to the size of the smallest probabilistic representation of the concept class.

We show that any pure-private learning algorithm for a concept class C with sample complexity n implies $\mathrm{RepDim}(C) = O(n)$, and that there exists a private learning algorithm with sample complexity $n = O(\mathrm{RepDim}(C))$.

5.1 Main Results

In the initial work on private learning, Kasiviswanathan et al. [67] presented a generic construction for learning a concept class C under pure-differential privacy using sample complexity of $O(\log|C|)$.

Beimel et al. [8] observed that the sample complexity can be reduced by considering the smallest hypothesis class H that represents C, in the scene that for every target concept $c \in C$ and every distribution $\mathcal{D}$, there exists a hypothesis $h \in H$ with small $\mathrm{error}_{\mathcal{D}}(c,h)$. Observe that $|H| \leq |C|$ as, in particular, the class C represents itself. Using the generic construction of Kasiviswanathan et al. to choose a hypothesis out of H (instead of C), the sample complexity is improved to $O(\log|H|)$. While for some classes this can dramatically improve the sample complexity, Beimel et al. showed that this technique is not optimal,

and that there are concept classes for which it is possible to further reduce the sample complexity using other techniques.

We make an additional step in improving the sample complexity by considering a *probabilistic* representation of a concept class C. Instead of one collection H representing C, we consider a list of collections $H_1, \ldots, H_r$ such that for every $c \in C$ and every distribution on the examples, if we sample a collection H_i from the list, then with high probability there is a hypothesis $h \in H_i$ that is close to c. To privately learn C, the learning algorithm first samples $i \in \{1, \ldots, r\}$ and then uses the generic construction of Kasiviswanathan et al. to select a hypothesis from H_i. This reduces the sample complexity to $O(\max_i \log|H_i|)$; the *size* of the probabilistic representation is hence defined to be $\max_i \log|H_i|$.

One can ask if there are pure-private learning algorithms with smaller sample complexity than the size of the smallest probabilistic representation. We show that the answer is no — the size of the smallest probabilistic representation is a lower bound on the sample complexity. Thus, the size of the smallest probabilistic representation of a class C, which we call the *representation dimension* and denote by $\mathrm{RepDim}(C)$, characterizes (up to constants) the sample size necessary and sufficient for learning the class C under pure-differential privacy.

As a by-product of our characterization we obtain a pure-private improper-learning algorithm for point functions with constant sample complexity, matching a different private algorithm presented in [8]. Our new algorithm offers some improvement in the sample complexity compared to the algorithm of [8] when considering the learning and privacy parameters. Furthermore, our algorithm can be made computationally efficient without making any computational hardness assumptions, while the efficient version in [8] assumes the existence of one-way functions.

Additional Results. The notion of probabilistic representation applies not only to private learning, but also to optimization problems. We consider a scenario where there is a domain X, a database S of n records, each taken from the domain X, a set of solutions F, and a quality function $q : X^* \times F \to [0,1]$ that we wish to maximize. If the exponential mechanism is used for (approximately) solving the problem, then the size of the database should be $\Omega(\ln|F|)$ in order to achieve a reasonable approximation. Using our notions of a representation of F and of a probabilistic representation of F, one can reduce the size of

the minimal database without paying too much in the quality of the solution. Interestingly, a similar notion to representation, called "solution list algorithms", was considered in [9] for constructing secure protocols for search problems while leaking only a few bits on the input. Curiously, their notion of leakage is very different from that of differential privacy. See the full version of this work for more details [10].

5.2 The Sample Complexity of Pure-Private Learners

In this section we present a combinatorial measure of a concept class C that characterizes the sample complexity necessary and sufficient for privately learning C. The measure is a *probabilistic representation* of the class C. We start with the notation of deterministic representation from [8].

Definition 5.1 ([8])**.** *Let C be a concept class over a domain X. A hypothesis class H is an α-representation for C if for every $c \in C$ and every distribution $\mathcal{D}$ on X there exists a hypothesis $h \in H$ such that* $\mathrm{error}_{\mathcal{D}}(c,h) \leq \alpha$.

Example 5.2 ($\texttt{POINT}_X$)**.** *Recall that for every $x \in X$ the class $\texttt{POINT}_X$ contains the concept $c_x : X \to \{0,1\}$ where $c_x(y) = 1$ iff $x = y$. In [8] it was shown that for $\alpha < 1/2$, every α-representation for $\texttt{POINT}_X$ must be of cardinality at least $\log|X|$, and that an α-representation H for $\texttt{POINT}_X$ exists where $|H| = O\left(\frac{1}{\alpha^2}\log|X|\right)$.*

The above representation can be used for non-private learning, by taking a big enough sample and finding a hypothesis $h \in H$ minimizing the empirical error. For *private* learning it was shown in [8] that a sample of size $O_{\alpha,\beta,\varepsilon}(\log|H|)$ suffices, with a learner that employs the exponential mechanism to choose a hypothesis from H.

Definition 5.3. *For a hypothesis class H we denote* $\mathrm{size}(H) = \ln|H|$*. We define the Deterministic Representation Dimension of a concept class C as*

$$\mathrm{DRepDim}(C) = \min\left\{\mathrm{size}(H) : H \text{ is a } \frac{1}{4}\text{-representation for } C\right\}.$$

Remark 5.4. *Choosing $\frac{1}{4}$ is arbitrary; we could have chosen any (smaller than $\frac{1}{2}$) constant.*

Example 5.5. *By the results of [8], stated in the previous example,* $\mathrm{DRepDim}(\texttt{POINT}_X) = \theta(\log\log|X|)$.

We are now ready to present the notion of a probabilistic representation. The idea behind this notion is that we have a list of hypothesis classes, such that for every concept c and distribution $\mathcal{D}$, if we sample a hypothesis class from the list, then with high probability it contains a hypothesis that is close to c.

Definition 5.6. *Let C be a concept class over a domain X. Let $\mathcal{P}$ be a distribution over $\{1,2,\ldots,r\}$, and let $\mathscr{H} = \{H_1, H_2, \ldots, H_r\}$ be a family of hypothesis classes (every $H_i \in \mathscr{H}$ is a set of boolean functions). We say that $(\mathscr{H},\mathcal{P})$ is an (α,β)-probabilistic representation for C if for every $c \in C$ and every distribution $\mathcal{D}$ on X:*

$$\Pr_{i\sim\mathcal{P}}\left[\exists h \in H_i \ \ s.t.\ \ \mathrm{error}_{\mathcal{D}}(c,h) \leq \alpha\right] \geq 1-\beta.$$

The probability is over randomly choosing a set $H_i \in \mathscr{H}$ (according to $\mathcal{P}$).

Remark 5.7. *As we will see in Section 5.2.1, the existence of such a probabilistic representation $(\mathscr{H},\mathcal{P})$ for a concept class C implies the existence of a private learning algorithm for C with sample complexity that depends on the cardinality of the hypothesis classes $H_i \in \mathscr{H}$. The sample complexity will not depend on $r = |\mathscr{H}|$.*

Example 5.8 ($\texttt{POINT}_X$)**.** *In Section 5.3 we construct for every α and every β a pair $(\mathscr{H},\mathcal{P})$ that (α,β)-probabilistically represents the class $\texttt{POINT}_X$, where $\mathscr{H}$ contains all the sets of at most $\frac{4}{\alpha}\ln(1/\beta)$ boolean functions.*

Definition 5.9. *Let $\mathscr{H} = \{H_1, H_2, \ldots, H_r\}$ be a family of hypothesis classes. We denote $|\mathscr{H}| = r$, and* $\mathrm{size}(\mathscr{H}) = \max\{\ln|H_i| : H_i \in \mathscr{H}\}$. *We define the Representation Dimension of a concept class C as*

$$\mathrm{RepDim}(C) = \min\left\{ \mathrm{size}(\mathscr{H}) \ : \ \begin{array}{l} \exists \mathcal{P} \text{ s.t. } (\mathscr{H},\mathcal{P}) \text{ is a} \\ (\frac{1}{4},\frac{1}{4})\text{-probabilistic} \\ \text{representation for } C \end{array} \right\}.$$

Remark 5.10. *Choosing $\alpha = \beta = \frac{1}{4}$ is arbitrary; we could have chosen any two (smaller than $\frac{1}{2}$) constants.*

Example 5.11 ($\texttt{POINT}_X$)**.** *The* size *of the probabilistic representation mentioned in Example 5.8 is* $\ln(\frac{4}{\alpha}\ln(1/\beta))$*. Placing* $\alpha = \beta = \frac{1}{4}$*, we see that the Representation Dimension of* $\texttt{POINT}_X$ *is constant.*

5.2.1 Equivalence of Probabilistic Representation and Private Learning

We now show that $\mathrm{RepDim}(C)$ characterizes the sample complexity of private learners. We start by showing in Lemma 5.12 that an (α,β)-probabilistic representation of C implies a private learning algorithm whose sample complexity is the size of the representation. We then show in Lemma 5.15 that if there is a private learning algorithm with sample complexity n, then there is probabilistic representation of C of size $O(n)$; this lemma implies that $\mathrm{RepDim}(C)$ is a lower bound on the sample complexity. Recall that $\mathrm{RepDim}(C)$ is the size of the smallest probabilistic representation for $\alpha = \beta = 1/4$. Thus, to complete the proof we show in Lemma 5.17 that a probabilistic representation with $\alpha = \beta = 1/4$ implies a probabilistic representation for arbitrary α and β.

Lemma 5.12. *If there exists a pair* $(\mathscr{H},\mathcal{P})$ *that* (α,β)*-probabilistically represents a class* C*, then for every* ε *there exists an algorithm* A *that* $(6\alpha,4\beta,\varepsilon)$*-PPAC learns* C *with a sample size* $n = O\Big(\frac{1}{\alpha\varepsilon}(\mathrm{size}(\mathscr{H}) + \ln(\frac{1}{\beta}))\Big)$.

Proof. Let $(\mathscr{H},\mathcal{P})$ be an (α,β)-probabilistic representation for the class C, and consider the following algorithm $\mathcal{A}$:

Algorithm $\mathcal{A}$

Inputs: Database $S = (x_i, y_i)_{i=1}^n$, and a privacy parameter ε.

1. Randomly choose $H_i \in \mathscr{H}$ according to $\mathcal{P}$.
2. Choose $h \in H_i$ using the exponential mechanism with privacy parameter ε and quality function $q(S,h) = |\{i : h(x_i) = y_i\}|$.

By the properties of the exponential mechanism, $\mathcal{A}$ is ε-differentially private. We will show that with sample size $n = O\Big(\frac{1}{\alpha\varepsilon}(\mathrm{size}(\mathscr{H}) + \ln(\frac{1}{\beta}))\Big)$, algorithm $\mathcal{A}$ is a $(6\alpha,4\beta)$-PAC learner for C. Fix a target concept $c \in C$ and a distribution $\mathcal{D}$, and define the following 3 good events:

E_1 H_i chosen in step 1 contains at least one hypothesis h s.t. $\text{error}_S(h) \leq 2\alpha$.

E_2 For every $h \in H_i$ s.t. $\text{error}_S(h) \leq 3\alpha$, it holds that $\text{error}_{\mathcal{D}}(c,h) \leq 6\alpha$.

E_3 The exponential mechanism chooses an h such that $\text{error}_S(h) \leq \alpha + \min_{f \in H_i}\{\text{error}_S(f)\}$.

We first show that if those 3 good events happen, algorithm $\mathcal{A}$ returns a 6α-good hypothesis. Event E_1 ensures the existence of a hypothesis $f \in H_i$ s.t. $\text{error}_S(f) \leq 2\alpha$. Thus, event $E_1 \cap E_3$ ensures algorithm $\mathcal{A}$ chooses (using the exponential mechanism) a hypothesis $h \in H_i$ s.t. $\text{error}_S(h) \leq 3\alpha$. Event E_2 ensures, therefore, that this h satisfies $\text{error}_{\mathcal{D}}(c,h) \leq 6\alpha$.

We will now show that these 3 events happen with high probability. As $(\mathscr{H},\mathcal{P})$ is an (α,β)-probabilistic representation for the class C, the chosen H_i contains a hypothesis h s.t. $\text{error}_{\mathcal{D}}(c,h) \leq \alpha$ with probability at least $1-\beta$; by the Chernoff bound with probability at least $1-\exp(-n\alpha/3)$ this hypothesis has empirical error at most 2α. Event E_1 happens with probability at least $(1-\beta)(1-\exp(-n\alpha/3)) > 1-(\beta+\exp(-n\alpha/3))$, which is at least $(1-2\beta)$ for $n \geq \frac{3}{\alpha}\ln(1/\beta)$.

Using the Chernoff bound, the probability that a hypothesis h s.t. $\text{error}_{\mathcal{D}}(c,h) > 6\alpha$ has empirical error $\leq 3\alpha$ is less than $\exp(-n\alpha 3/4)$. Using the union bound, the probability that there is such a hypothesis in H_i is at most $|H_i| \cdot \exp(-n\alpha 3/4)$. Therefore, $\Pr[E_2] \geq 1-|H_i| \cdot \exp(-n\alpha 3/4)$. For $n \geq \frac{4}{3\alpha}(\ln(\frac{|H_i|}{\beta}))$, this probability is at least $(1-\beta)$.

The exponential mechanism ensures that the probability of event E_3 is at least $1-|H_i| \cdot \exp(-\varepsilon\alpha n/2)$ (see Section 3.5.2), which is at least $(1-\beta)$ for $n \geq \frac{2}{\alpha\varepsilon}\ln(\frac{|H_i|}{\beta})$.

All in all, by setting $n = \frac{3}{\alpha\varepsilon}(\text{size}(\mathscr{H}) + \ln(\frac{1}{\beta}))$ we ensure that the probability of $\mathcal{A}$ failing to output a 6α-good hypothesis is at most 4β. □

We will demonstrate the above lemma with two examples:

Example 5.13 (Efficient learner for $\texttt{POINT}_X$)**.** *As described in Example 5.8, there exists an* $(\mathscr{H},\mathcal{P})$ *that* $(\alpha/6,\beta/4)$*-probabilistically represents the class* $\texttt{POINT}_X$*, where* $\text{size}(\mathscr{H}) = O_{\alpha,\beta,\varepsilon}(1)$*. By Lemma 5.12, there exists an algorithm that* $(\alpha,\beta,\varepsilon)$*-PPAC learns* C *with sample size* $m = O_{\alpha,\beta,\varepsilon}(1)$*.*

The existence of an algorithm with sample complexity $O(1)$ *was already proven in [8]. Moreover, assuming the existence of oneway functions, their learner is efficient. Our construction yields an efficient learner, without assumptions. To see this, consider again algorithm* $\mathcal{A}$ *presented*

in the above proof, and note that as $\text{size}(\mathscr{H})$ *is constant, step 2 could be done in constant time. Step 1 can be done efficiently as we can efficiently sample a set* $H_i \in_{\mathcal{P}} \mathscr{H}$*. In Claim 5.19 we initially construct a probabilistic representation in which the description of every hypothesis is exponential in* d*. The representation is then revised using pairwise independence to yield a representation in which every hypothesis* h *has a short description, and given* x *the value* $h(x)$ *can be computed efficiently.*

The next lemma shows that a private learning algorithm implies a probabilistic representation. This lemma can be used to lower bound the sample complexity of private learners.

Lemma 5.14. *If there exists an algorithm* $\mathcal{A}$ *that* $(\alpha, \frac{1}{2}, \varepsilon)$*-PPAC learns a concept class* C *with a sample size* n*, then there exists a pair* $(\mathscr{H}, \mathcal{P})$ *that* $(\alpha, 1/4)$*-probabilistically represents the class* C *such that* $\text{size}(\mathscr{H}) \leq n\varepsilon + 2$*.*

Proof. Let $\mathcal{A}$ be an $(\alpha, \frac{1}{2}, \varepsilon)$-PPAC learner for a class C using hypothesis class F whose sample size is n. For a target concept $c \in C$ and a distribution $\mathcal{D}$, we define G as the set of all hypotheses $h \in F$ such that $\text{error}_{\mathcal{D}}(c,h) \leq \alpha$. Fix some $c \in C$ and a distribution $\mathcal{D}$. As $\mathcal{A}$ is an $(\alpha, \frac{1}{2})$-PAC learner, $\Pr_{\mathcal{D},\mathcal{A}}[\mathcal{A}(S) \in G] \geq \frac{1}{2}$, where the probability is over $\mathcal{A}$'s randomness and over sampling the examples in S (according to $\mathcal{D}$). Therefore, there exists a database S of n samples such that $\Pr_{\mathcal{A}}[\mathcal{A}(S) \in G] \geq \frac{1}{2}$, where the probability is only over the randomness of $\mathcal{A}$. As $\mathcal{A}$ is ε-differentially private, $\Pr_{\mathcal{A}}\left[\mathcal{A}(\vec{0}) \in G\right] \geq e^{-n\varepsilon} \cdot \Pr_{\mathcal{A}}[\mathcal{A}(S) \in G] \geq \frac{1}{2}e^{-n\varepsilon}$, where $\vec{0}$ is a database with n zeros.[1] That is, $\Pr_{\mathcal{A}}\left[\mathcal{A}(\vec{0}) \notin G\right] \leq 1 - \frac{1}{2}e^{-n\varepsilon}$. Now, consider a set H containing the outcomes of $2\ln(4)e^{n\varepsilon}$ independent executions of $\mathcal{A}(\vec{0})$. The probability that H does not contain an α-good hypothesis is at most $(1 - \frac{1}{2}e^{-n\varepsilon})^{2\ln(4)e^{n\varepsilon}} \leq \frac{1}{4}$. Thus, $\mathscr{H} = \{H \subseteq F : |H| \leq 2\ln(4)e^{n\varepsilon}\}$, and $\mathcal{P}$, the distribution induced by $\mathcal{A}(\vec{0})$, are an $(\alpha, 1/4)$-probabilistic representation for class C. It follows that $\text{size}(\mathscr{H}) = \max\{\ln|H| : H \in \mathscr{H}\} = \ln(2\ln(4)) + n\varepsilon < 2 + n\varepsilon$. □

The above lemma yields a lower bound of $\Omega\left(\frac{1}{\varepsilon}\text{RepDim}(C)\right)$ on the sample complexity of private learners for a concept class C. To see this, fix $\alpha \leq \frac{1}{4}$ and let $\mathcal{A}$ be an $(\alpha, \frac{1}{2}, \varepsilon)$-PPAC learner for C with sample size n. By the above lemma, there exists a pair $(\mathscr{H}, \mathcal{P})$

[1] Choosing $\vec{0}$ is arbitrary; we could have chosen any database.

that $(\alpha, 1/4)$-probabilistically represents C s.t. $\text{size}(\mathscr{H}) \leq 2 + n\varepsilon$. Therefore, by definition, $\text{RepDim}(C) \leq 2 + n\varepsilon$. Thus, $n \geq \frac{1}{\varepsilon}(\text{RepDim}(C) - 2) = \Omega\left(\frac{1}{\varepsilon}\text{RepDim}(C)\right)$.

In order to refine this lower bound (and incorporate α in it), we will need a somewhat stronger version of this lemma:

Lemma 5.15. *Let $\alpha \leq 1/4$. If there exists an algorithm $\mathcal{A}$ that $(\alpha, \frac{1}{2}, \varepsilon)$-PPAC learns a concept class C with a sample size n, then there exists a pair $(\mathscr{H}, \mathcal{P})$ that $(1/4, 1/4)$-probabilistically represents the class C such that $\text{size}(\mathscr{H}) \leq n\varepsilon\alpha + 3$.*

Proof. Let $\mathcal{A}$ be an $(\alpha, \frac{1}{2}, \varepsilon)$-PPAC learner for the class C using hypothesis class F whose sample size is n. Without loss of generality, we can assume that $n \geq \frac{3\ln(4)}{4\alpha}$ (since the learner can ignore part of the sample). For a target concept $c \in C$ and a distribution $\mathcal{D}$, we define

$$G^{\alpha}_{\mathcal{D}} = \{h \in F : \text{error}_{\mathcal{D}}(c, h) \leq \alpha\}.$$

Fix some $c \in C$ and a distribution $\mathcal{D}$ over X, and define the following distribution $\widetilde{\mathcal{D}}$ on X:[2]

$$\Pr_{\widetilde{\mathcal{D}}}[x] = \begin{cases} 1 - 4\alpha + 4\alpha \cdot \Pr_{\mathcal{D}}[0], & x = 0. \\ 4\alpha \cdot \Pr_{\mathcal{D}}[x], & x \neq 0. \end{cases}$$

Note that for every $x \in X$,

$$\Pr_{\widetilde{\mathcal{D}}}[x] \geq 4\alpha \cdot \Pr_{\mathcal{D}}[x]. \tag{5.1}$$

As $\mathcal{A}$ is an $(\alpha, \frac{1}{2})$-PAC learner, it holds that

$$\Pr_{\widetilde{\mathcal{D}}, \mathcal{A}}\left[\mathcal{A}(S) \in G^{\alpha}_{\widetilde{\mathcal{D}}}\right] \geq \frac{1}{2},$$

where the probability is over $\mathcal{A}$'s randomness and over sampling the examples in S (according to $\widetilde{\mathcal{D}}$). In addition, by inequality (5.1), every hypothesis h with $\text{error}_{\mathcal{D}}(c, h) > 1/4$ has error strictly greater than α under $\widetilde{\mathcal{D}}$:

$$\text{error}_{\widetilde{\mathcal{D}}}(c, h) \geq 4\alpha \cdot \text{error}_{\mathcal{D}}(c, h) > \alpha.$$

[2] We assume that $0 \in X$. This choice is arbitrary, and 0 could be replaced with any element of X.

So, every α-good hypothesis for c and $\widetilde{\mathcal{D}}$ is a $\frac{1}{4}$-good hypothesis for c and $\mathcal{D}$. That is, $G^{\alpha}_{\widetilde{\mathcal{D}}} \subseteq G^{1/4}_{\mathcal{D}}$. Therefore, $\Pr_{\widetilde{\mathcal{D}},\mathcal{A}}\left[\mathcal{A}(S) \in G^{1/4}_{\mathcal{D}}\right] \geq \frac{1}{2}$.

We say that a database S of n labeled examples is *good* if the unlabeled example 0 appears in S at least $(1-8\alpha)n$ times. Let S be a database constructed by taking n i.i.d. samples from $\widetilde{\mathcal{D}}$, labeled by c. By the Chernoff bound, S is good with probability at least $1-\exp(-4\alpha n/3)$. Hence,

$$\Pr_{\widetilde{\mathcal{D}},\mathcal{A}}\left[(\mathcal{A}(S) \in G^{1/4}_{\mathcal{D}}) \wedge (S \text{ is good})\right] \geq \frac{1}{2} - \exp(-4\alpha n/3) \geq \frac{1}{4}.$$

Therefore, there exists a database S_{good} of n samples that contains the unlabeled example 0 at least $(1-8\alpha)n$ times, and $\Pr_{\mathcal{A}}\left[\mathcal{A}(S_{\text{good}}) \in G^{1/4}_{\mathcal{D}}\right] \geq \frac{1}{4}$, where the probability is only over the randomness of $\mathcal{A}$. All of the examples in S_{good} (including the example 0) are labeled by c.

For $\sigma \in \{0,1\}$, denote by $\vec{0}_\sigma$ a database containing n copies of the example 0 labeled as σ. As $\mathcal{A}$ is ε-differentially private, and as the target concept c labels the example 0 by either 0 or 1, for at least one $\sigma \in \{0,1\}$ it holds that

$$\begin{aligned}\Pr_{\mathcal{A}}[\mathcal{A}(\vec{0}_\sigma) \in G^{1/4}_{\mathcal{D}}] &\geq \exp(-8\alpha\varepsilon n) \cdot \Pr_{\mathcal{A}}\left[\mathcal{A}(S_{\text{good}}) \in G^{1/4}_{\mathcal{D}}\right] \\ &\geq \exp(-8\alpha\varepsilon n) \cdot 1/4. \end{aligned} \tag{5.2}$$

That is, $\Pr_{\mathcal{A}}[\mathcal{A}(\vec{0}_\sigma) \notin G^{1/4}_{\mathcal{D}}] \leq 1 - \frac{1}{4}e^{-8\alpha\varepsilon n}$. Now, consider a set H containing the outcomes of $4\ln(4)e^{8\alpha\varepsilon n}$ independent executions of $\mathcal{A}(\vec{0}_0)$, and the outcomes of $4\ln(4)e^{8\alpha\varepsilon n}$ independent executions of $\mathcal{A}(\vec{0}_1)$. The probability that H does not contain a $\frac{1}{4}$-good hypothesis for c and $\mathcal{D}$ is at most $(1-\frac{1}{4}e^{-8\alpha\varepsilon n})^{4\ln(4)e^{8\alpha\varepsilon n}} \leq \frac{1}{4}$. Thus, $\mathscr{H} = \left\{H \subseteq \mathcal{F} \,:\, |H| \leq 2 \cdot 4\ln(4)e^{8\alpha\varepsilon n}\right\}$, and $\mathcal{P}$, the distribution induced by $\mathcal{A}(\vec{0}_0)$ and $\mathcal{A}(\vec{0}_1)$, are a $(1/4, 1/4)$-probabilistic representation for the class C. Note that the value $c(0)$ is unknown, and can be either 0 or 1. Therefore, the construction uses the two possible values (one of them correct).

It holds that $\text{size}(\mathscr{H}) = \max\{\ln|H| : H \in \mathscr{H}\} = \ln(8\ln(4)) + 8\alpha\varepsilon n \leq n\varepsilon\alpha + 3$. □

Lemma 5.17 below shows how to construct a probabilistic representation for an arbitrary α and β from a probabilistic representation with $\alpha = \beta = 1/4$; in other words we boost α and β. The proof of this lemma is combinatorial. It allows us to start with

a private learning algorithm with constant α and β, move to a representation, use the combinatorial boosting, and move back to a private algorithm with small α and β. This should be contrasted with the private boosting of [48], which is algorithmic and more complicated (however, the algorithm of Dwork et al. [48] is computationally efficient).

We first show how to construct a probabilistic representation for arbitrary β from a probabilistic representation with $\beta = 1/4$.

Claim 5.16. *For every concept class C and for every β, there exists a pair $(\mathscr{H},\mathcal{P})$ that $(1/4,\beta)$-probabilistically represents C where* $\text{size}(\mathscr{H}) \leq \text{RepDim}(C) + \ln\ln(1/\beta)$.

Proof. Let $\beta < 1/4$, and let $(\mathscr{H}^0,\mathcal{P}^0)$ be a $(\frac{1}{4},\frac{1}{4})$-probabilistic representation for C with $\text{size}(\mathscr{H}^0) = \text{RepDim}(C) \triangleq k_0$ (that is, for every $H_i^0 \in \mathscr{H}^0$ it holds that $|H_i^0| \leq e^{k_0}$). Denote $\mathscr{H}^0 = \{H_1^0, H_2^0, \ldots, H_r^0\}$, and consider the following family of hypothesis classes:

$$\mathscr{H}^1 = \left\{ H_{i_1}^0 \cup \cdots \cup H_{i_{\ln(1/\beta)}}^0 \;:\; 1 \leq i_1 \leq \cdots \leq i_{\ln(1/\beta)} \leq r \right\}.$$

Note that for every $H_i^1 \in \mathscr{H}^1$ it holds that $|H_i^1| \leq \ln(1/\beta)e^{k_0}$; so, $\text{size}(\mathscr{H}^1) \triangleq k_1 \leq k_0 + \ln\ln(1/\beta)$. We will now show an appropriate distribution $\mathcal{P}^1$ on $\mathscr{H}^1$ s.t. $(\mathscr{H}^1,\mathcal{P}^1)$ is a $(\frac{1}{4},\beta)$-probabilistic representation for C. To this end, consider the following process for randomly choosing an $H^1 \in \mathscr{H}^1$:

1. Denote $M = \ln(1/\beta)$
2. For $i = 1,\ldots,M$:
   Randomly choose $H_i^0 \in \mathscr{H}^0$ according to $\mathcal{P}^0$.
3. Return $H^1 = \bigcup_{i=1}^M H_i^0$.

The above process induces a distribution on $\mathscr{H}^1$, denoted as $\mathcal{P}^1$. As $\mathscr{H}^0$ is a $(\frac{1}{4},\frac{1}{4})$-probabilistic representation for C, we have that

$$\Pr_{\mathcal{P}^1}\left[\nexists h \in H^1 \; s.t. \; \text{error}_{\mathcal{D}}(c,h) \leq 1/4\right] = \prod_{i=1}^{M} \Pr_{\mathcal{P}^0}\left[\nexists h \in H_i^0 \; s.t. \; \text{error}_{\mathcal{D}}(c,h) \leq 1/4\right] \leq \left(\frac{1}{4}\right)^M \leq \beta. \quad \square$$

Lemma 5.17. *For every concept class C, every α, and every β, there exists $(\mathscr{H},\mathcal{P})$ that (α,β)-*

probabilistically represents C where

$$\text{size}(\mathscr{H}) = O\Big(\ln(\frac{1}{\alpha}) \cdot \Big(\text{RepDim}(C) + \ln\ln\ln(\frac{1}{\alpha}) + \ln\ln(\frac{1}{\beta})\Big)\Big).$$

Proof. Let C be a concept class, and let $(\mathscr{H}^1, \mathcal{P}^1)$ be a $(\frac{1}{4}, \beta/T)$-probabilistic representation for C (where T will be set later). By Claim 5.16, such a representation exists with $\text{size}(\mathscr{H}^1) \triangleq k_1 \leq \text{RepDim}(C) + \ln\ln(T/\beta)$. We use $\mathscr{H}^1$ and $\mathcal{P}^1$ to create an (α, β)-probabilistic representation for C. We begin with two notations:

1. For T hypotheses $h_1, \ldots, h_T$ we denote by $\text{maj}_{h_1,\ldots,h_T}$ the majority hypothesis. That is, $\text{maj}_{h_1,\ldots,h_T}(x) = 1$ if and only if $|\{h_i \,:\, h_i(x) = 1\}| \geq T/2$.
2. For T hypothesis classes $H_1, \ldots, H_T$ we denote $\text{MAJ}(H_1, \ldots, H_T) = \Big\{\text{maj}_{h_1,\ldots,h_T} \,:\, \forall_{1 \leq i \leq T}\, h_i \in H_i\Big\}$.

Consider the following family of hypothesis classes:

$$\mathscr{H} = \Big\{\text{MAJ}(H_{i_1}, \ldots, H_{i_T}) \,:\, H_{i_1}, \ldots, H_{i_T} \in \mathscr{H}^1\Big\}.$$

Moreover, denote the distribution on $\mathscr{H}$ induced by the following random process as $\mathcal{P}$:

> For $j = 1, \ldots, T$:
> Randomly choose $H_{i_j} \in_{\mathcal{P}^1} \mathscr{H}^1$
> Return $\text{MAJ}(H_{i_1}, \ldots, H_{i_T})$.

Next we show that $(\mathscr{H}, \mathcal{P})$ is an (α, β)-probabilistic representation for C: For a fixed pair of a target concept c and a distribution $\mathcal{D}$, randomly choose $H_{i_1}, \ldots, H_{i_T} \in \mathscr{H}^1$ according to $\mathcal{P}^1$. We now show that with probability at least $(1 - \beta)$ the set $\text{MAJ}(H_{i_1}, \ldots, H_{i_T})$ contains at least one α-good hypothesis for $c, \mathcal{D}$.

To this end, denote $\mathcal{D}_1 = \mathcal{D}$ and consider the following thought experiment, inspired by the AdaBoost Algorithm [53]:

For $t = 1\ldots,T$:

1. Fail if H_{i_t} does not contain a $\frac{1}{4}$-good hypothesis for $c, \mathcal{D}_t$.
2. Denote by $h_t \in H_{i_t}$ a $\frac{1}{4}$-good hypothesis for $c, \mathcal{D}_t$.
3. $$\mathcal{D}_{t+1}(x) = \begin{cases} 2\mathcal{D}_t(x), & \text{if } h_t(x) \neq c(x). \\ \left(1 - \frac{\text{error}_{\mathcal{D}_t}(c,h_t)}{1-\text{error}_{\mathcal{D}_t}(c,h_t)}\right)\mathcal{D}_t(x), & \text{otherwise.} \end{cases}$$

Note that as $\mathcal{D}_1$ is a probability distribution on X; the same is true for $\mathcal{D}_2, \mathcal{D}_3, \ldots, \mathcal{D}_T$. As $(\mathscr{H}^1, \mathcal{P}^1)$ is a $(\frac{1}{4}, \beta/T)$-probabilistic representation for C, the failure probability of every iteration is at most β/T. Thus (using the union bound), with probability at least $(1-\beta)$ the whole thought experiment will succeed, and in this case we show that the error of $h_{\text{fin}} = \text{maj}_{h_1,\ldots,h_T}$ is at most α.

Consider the set $R = \{x : h_{\text{fin}}(x) \neq c(x)\} \subseteq X$. This is the set of points on which at least $T/2$ of $h_1, \ldots, h_T$ err. Next consider the partition of R to the following sets:

$$R_t = \left\{x \in R : \left(h_t(x) \neq c(x)\right) \wedge \left(\forall_{i>t}\, h_i(x) = c(x)\right)\right\}.$$

That is, R_t contains the points $x \in R$ on which h_t is last to err. Clearly $\mathcal{D}_t(R_t) \leq 1/4$, as R_t is a subset of the set of points on which h_t errs. Moreover, by the (recursive) definition of $\mathcal{D}_t$, for every $x \in R_t$ we have that $\mathcal{D}_t(x)$ is obtained from $\mathcal{D}_1(x)$ after multiplying it by 2 for every $1 \leq i \leq t$ s.t. $h_i(x) \neq c(x)$, and multiplying it by $\left(1 - \frac{\text{error}_{\mathcal{D}_i}(c,h_i)}{1-\text{error}_{\mathcal{D}_i}(c,h_i)}\right)$ for every $1 \leq i < t$ s.t. $h_i(x) = c(x)$. As there are at least $T/2$ rounds $1 \leq i \leq t$ s.t. $h_i(x) \neq c(x)$, we have that

$$\begin{aligned}
\mathcal{D}_t(R_t) &\geq \mathcal{D}_1(R_t) \cdot 2^{T/2} \cdot \left(\min_{1\leq i<t}\left\{1 - \frac{\text{error}_{\mathcal{D}_i}(c,h_i)}{1-\text{error}_{\mathcal{D}_i}(c,h_i)}\right\}\right)^{t-T/2} \\
&\geq \mathcal{D}_1(R_t) \cdot 2^{T/2} \cdot \left(1 - \frac{1/4}{1-1/4}\right)^{t-T/2} \qquad \left(\text{since } (1-\tfrac{y}{1-y}) \geq (1-\tfrac{1/4}{1-1/4}) \text{ for every } y \leq \tfrac{1}{4}\right) \\
&\geq \mathcal{D}_1(R_t) \cdot 2^{T/2} \cdot \left(1 - \frac{1/4}{1-1/4}\right)^{T/2} \\
&= \mathcal{D}(R_t) \cdot \left(\frac{4}{3}\right)^{T/2},
\end{aligned}$$

so,

$$\mathcal{D}(R_t) \leq \mathcal{D}_t(R_t) \cdot \left(\frac{4}{3}\right)^{-T/2} \leq \frac{1}{4} \cdot \left(\frac{4}{3}\right)^{-T/2}.$$

Finally,

$$\text{error}_{\mathcal{D}}(c, h_{\text{fin}}) = \mathcal{D}(R) = \sum_{t=T/2}^{T} \mathcal{D}(R_t) \leq \frac{T}{2} \cdot \frac{1}{4} \cdot \left(\frac{4}{3}\right)^{-T/2} = \frac{T}{8} \cdot \left(\frac{4}{3}\right)^{-T/2}.$$

Choosing $T = 14\ln(\frac{2}{\alpha})$, we get that $\text{error}_{\mathcal{D}}(c, h_{\text{fin}}) \leq \alpha$. Hence, $(\mathscr{H}, \mathcal{P})$ is an (α, β)-probabilistic representation for C. Moreover, for every $H_i \in \mathscr{H}$ we have that $|H_i| \leq \left(e^{k_1}\right)^T$, and so

$$\begin{aligned}
\text{size}(\mathscr{H}) &\leq k_1 \cdot T \\
&\leq \Big(\text{RepDim}(C) + \ln\ln(T/\beta)\Big) \cdot T \\
&= O\Big(\ln(1/\alpha) \cdot \Big(\text{RepDim}(C) + \ln\ln\ln(1/\alpha) + \ln\ln(1/\beta)\Big)\Big).
\end{aligned}$$

□

The next theorem states the main result of this section – RepDim characterizes the sample complexity of private learning.

Theorem 5.18. *Let C be a concept class. Then, $\widetilde{\Theta}_\beta\left(\frac{\text{RepDim}(C)}{\alpha\varepsilon}\right)$ samples are necessary and sufficient for the private learning of the class C.*

Proof. Fix some $\alpha \leq 1/4, \beta \leq 1/2$, and ε. By Lemma 5.17, there exists a pair $(\mathscr{H}, \mathcal{P})$ that $(\frac{\alpha}{6}, \frac{\beta}{4})$-represent class C, where $\text{size}(\mathscr{H}) = O\Big(\ln(1/\alpha) \cdot \Big(\text{RepDim}(C) + \ln\ln\ln(1/\alpha) + \ln\ln(1/\beta)\Big)\Big)$. Therefore, by Lemma 5.12, there exists an algorithm $\mathcal{A}$ that $(\alpha, \beta, \varepsilon)$-PPAC learns the class C with a sample size

$$n = O_\beta\left(\frac{1}{\alpha\varepsilon}\ln(\frac{1}{\alpha}) \cdot \Big(\text{RepDim}(C) + \ln\ln\ln(\frac{1}{\alpha})\Big)\right).$$

For the lower bound, let $\mathcal{A}$ be an $(\alpha, \beta, \varepsilon)$-PPAC learner for the class C with a sample size n, where $\alpha \leq 1/4$ and $\beta \leq 1/2$. By Lemma 5.15, there exists an $(\mathscr{H}, \mathcal{P})$ that $(\frac{1}{4}, \frac{1}{4})$-probabilistically represents the class C and $\text{size}(\mathscr{H}) = \ln(8) + \ln\ln(4) + 8\alpha\varepsilon n$. Therefore,

by definition, $\text{RepDim}(C) \leq \ln(8\ln(4)) + 8\alpha\varepsilon n$. Thus,

$$n \geq \frac{1}{8\alpha\varepsilon} \cdot \Big(\text{RepDim}(C) - \ln(8\ln(4))\Big) = \Omega\left(\frac{\text{RepDim}(C)}{\alpha\varepsilon}\right).$$

□

5.3 A Probabilistic Representation for Points

Example 5.8 states the existence of a constant size probabilistic representation for the class $\texttt{POINT}_X$. We now give the construction.

Claim 5.19. *There exists an (α,β)-probabilistic representation for $\texttt{POINT}_X$ of size $\ln(4/\alpha) + \ln\ln(1/\beta)$. Furthermore, each hypothesis h in each H_i has a short description and given $x \in X$, the value $h(x)$ can be computed efficiently.*

Proof. Consider the following set of hypothesis classes

$$\mathscr{H} = \left\{H \subseteq 2^X \;:\; |H| \leq \frac{4}{\alpha}\ln(\frac{1}{\beta})\right\}.$$

That is, $H \in \mathscr{H}$ if H contains at most $\frac{4}{\alpha}\ln(\frac{1}{\beta})$ boolean functions. We will show an appropriate distribution $\mathcal{P}$ s.t. $(\mathscr{H},\mathcal{P})$ is an (α,β)-probabilistic representation of the class $\texttt{POINT}_X$. To this end, fix a target concept $c_j \in \texttt{POINT}_X$ and a distribution $\mathcal{D}$ on X (remember that $j \in X$ is the unique point on which $c_j(j) = 1$). We need to show how to randomly choose an $H \in \mathscr{H}$ such that with probability at least $(1-\beta)$ over the choice of H, there will be at least one $h \in H$ such that $\text{error}_{\mathcal{D}}(c_j,h) \leq \alpha$. Consider the following process for randomly choosing an $H \in \mathscr{H}$:

1. Denote $M = \frac{4}{\alpha}\ln(\frac{1}{\beta})$
2. For $i = 1,\ldots,M$ construct hypothesis h_i as follows:
   For each $x \in X$ (independently):
   Let $h_i(x) = 1$ with probability $\alpha/2$,
   and $h_i(x) = 0$ otherwise.
3. Return $H = \{h_1, h_2, \ldots, h_M\}$.

The above process induces a distribution on $\mathscr{H}$, denoted as $\mathcal{P}$. We will next analyze the probability that the returned H does not contain an α-good hypothesis. We start by fixing some i and analyzing the expected error of h_i, conditioned on the event that $h_i(j) = 1$. The probability is taken over the random coins used to construct h_i.

$$\begin{aligned}
\mathop{\mathbb{E}}_{h_i}\left[\mathrm{error}_{\mathcal{D}}(c_j, h_i) \;\middle|\; h_i(j) = 1\right] &= \mathop{\mathbb{E}}_{h_i}\left[\mathop{\mathbb{E}}_{x\sim\mathcal{D}}\left[\left|c_j(x) - h_i(x)\right|\right] \;\middle|\; h_i(j) = 1\right] \\
&= \mathop{\mathbb{E}}_{x\sim\mathcal{D}}\left[\mathop{\mathbb{E}}_{h_i}\left[\left|c_j(x) - h_i(x)\right| \;\middle|\; h_i(j) = 1\right]\right] \\
&\leq \frac{\alpha}{2}.
\end{aligned}$$

Using Markov's Inequality,

$$\Pr_{h_i}\left[\mathrm{error}_{\mathcal{D}}(c_j, h_i) \geq \alpha \;\middle|\; h_i(j) = 1\right] \leq \frac{1}{2}.$$

So, the probability that h_i is α-good for c_j and $\mathcal{D}$ is:

$$\begin{aligned}
\Pr_{h_i}\left[\mathrm{error}_{\mathcal{D}}(c_j, h_i) \leq \alpha\right] &\geq \Pr_{h_i}\left[h_i(j) = 1\right] \cdot \Pr_{h_i}\left[\mathrm{error}_{\mathcal{D}}(c_j, h_i) \leq \alpha \;\middle|\; h_i(j) = 1\right] \\
&\geq \frac{\alpha}{2} \cdot \frac{1}{2} = \frac{\alpha}{4}.
\end{aligned}$$

Thus, the probability that H fails to contain an α-good hypothesis is at most $\left(1 - \frac{\alpha}{4}\right)^M$, which is less than β for our choice of M. This concludes the proof that $(\mathscr{H}, \mathcal{P})$ is an (α, β)-probabilistic representation for $\texttt{POINT}_X$.

When a hypothesis $h_i()$ was constructed in the above random process, the value of $h_i(x)$ was independently drawn for every $x \in X$. This results in a hypothesis whose description size is $O(|X|)$, which in turn, will result in a non-efficient learning algorithm. We next explain how to construct hypotheses whose description is short. To achieve this goal, note that in the above analysis we only care about the probability that $h_i(x) = 0$ given that $h_i(j) = 1$. Thus, we can choose the values of h_i in a pairwise independent way, e.g., using a random polynomial of degree 2. The size of the description in this case is $O(\log|X|)$. □

Chapter 6

Private Learning: Pure vs. Approximate Differential Privacy

In the previous chapter we characterized the sample complexity of pure-private learners. In this chapter we show striking differences between the required sample complexity for private learning under pure-differential privacy and its variant approximate-differential privacy.

6.1 Main Results

Recall that the sample complexity of properly learning point functions over a domain X with pure differential privacy is $\Omega(\log|X|)$ [8]. An instantiation of the Propose-Test-Release (PTR) framework [43] by Smith and Thakurta [86] results, almost immediately, with a proper learner for point functions, exhibiting $O(1)$ sample complexity while preserving approximate differential privacy. This simple technique does not suffice for our other constructions, and we introduce a new tool for coping with proper private learning of thresholds and axis-aligned rectangles:

Recursive algorithm for quasi-concave promise problems. We define a family of optimization problems, which we call *Quasi-Concave Promise Problems*. The possible solutions to these problems are ordered, and *quasi-concavity* means that if two solutions $f \leq h$ have quality of at least $\mathcal{X}$, then any solution $f \leq g \leq h$ also has quality of at least $\mathcal{X}$. The optimization goal is, when there exists a solution with a promised quality of (at least) r, to find a solution with quality $\approx r$. We observe that a quasi-concave promise problem can be privately approximated using a solution to a smaller instance of a quasi-concave

promise problem. This allows us to construct an efficient recursive algorithm solving such problems privately. We show that the task of learning threshold functions over a domain X is, in fact, a quasi-concave promise problem, and it can be privately solved using our algorithm with sample size roughly $2^{O(\log^* |X|)}$.

Implications for private learning. We give new approximate-private *proper*-learning algorithms for point and threshold functions. We also construct a new private proper-learner for (a discrete version of) the class of all axis-aligned rectangles over ℓ dimensions. Our algorithms exhibit sample complexity that is significantly lower than bounds given in prior work, either with no dependency on the domain size $|X|$, or with a very mild dependency of $2^{O(\log^* |X|)}$. This separates the sample complexity of proper-learning under pure and approximate privacy, as $\Omega(\log|X|)$ samples are necessary for each of those learning tasks under pure-differential privacy. Our algorithms are time-efficient.

6.1.1 Additional Results

In the full version of this work [11] we extend our results in the following directions.

Private Query Release. Given a set Q of queries $q : X^n \to \mathbb{R}$, the *query release* problem for Q is to output accurate answers to all queries in Q. That is, we want a differentially private algorithm $\mathcal{M} : X^n \to \mathbb{R}^{|Q|}$ such that for every database $S \in X^n$, with high probability over $y \leftarrow \mathcal{M}(S)$, we have $y_q \approx q(S)$ for all $q \in Q$. As with private learning, we show significant differences between the sample complexity required for private query release of simple predicate classes under pure and approximate differential privacy.

Label privacy. We examine private learning under a relaxation of differential privacy called *label privacy* (see [26] and references therein), where the learner is required to only protect the privacy of the labels in the sample. Chaudhuri et al. [26] have proved lower bounds for label-private learners in terms of the doubling dimension of the target concept class. We show that the VC dimension completely characterizes the sample complexity of such learners; that is, the sample complexity of learning with label privacy is equal (up to constants) to learning without privacy.

6.2 Approximate-Private Proper-Learner for Points

Recall that $\Omega(\log|X|)$ examples are necessary for every pure-private proper-learner for the class $\texttt{POINT}_X$, defined as

Definition 6.1. *For $j \in X$ let $c_j : X \to \{0,1\}$ be defined as $c_j(x) = 1$ if $x = j$ and $c_j(x) = 0$ otherwise. Define the concept class $\texttt{POINT}_X = \{c_j\}_{j \in X}$.*

As we will now see, algorithm $\mathcal{A}_{\text{dist}}$ (defined in Section 3.5) can be used as a *proper* (ε, δ)-private learner for $\texttt{POINT}_X$ with sample complexity $O_{\alpha,\beta,\varepsilon,\delta}(1)$. This is our first (and simplest) example separating the sample complexity of pure and approximate private proper-learners. Consider the following algorithm.

Algorithm LearnPoints

Input: parameters $\alpha, \beta, \varepsilon, \delta$, and a labeled database $S \in (X \times \{0,1\})^n$.

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

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

Lemma 6.2. *Let $\alpha, \beta, \varepsilon, \delta$ be s.t. $2/(\alpha\beta) \leq |X|$. Algorithm* LearnPoints *is an efficient $(\alpha, \beta, \varepsilon, \delta)$-PPAC proper learner for $\texttt{POINT}_X$ using a sample of*

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

labeled examples.

Proof. As algorithm $\mathcal{A}_{\text{dist}}$ is (ε, δ)-differentially private, all that remains is the utility analysis. Fix a target concept $c_\ell \in \texttt{POINT}_X$ and a distribution $\mathcal{D}$ on X. In a sample S labeled

by c_ℓ, the only point that can appear with the label 1 is ℓ, and algorithm $\mathcal{A}_{\text{dist}}$ has two possible outputs: $\ell, \perp$.

If $\mathcal{D}(\ell) > \frac{\alpha}{2}$ then (using the Chernoff bound), with probability at least $(1 - \exp(-\alpha n/16))$, the labeled example $(\ell, 1)$ appears in S at least $r = \alpha n/4$ times. Note that $q(S, \ell) \geq r$, and every $i \neq \ell$ has quality $q(S, i) = 0$. For

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

by Proposition 3.28, this gap is big enough s.t. algorithm $\mathcal{A}_{\text{dist}}$ outputs ℓ with probability at least $(1 - \beta/2)$. Therefore, when $\mathcal{D}(\ell) > \frac{\alpha}{2}$, the probability of LearnPoints outputting an α-good solution is at least $(1 - \exp(-\alpha n/16))(1 - \beta/2)$, which is at least $(1 - \beta)$ for $n \geq (16/\alpha)\ln(2/\beta)$.

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

Remark 6.3. *Recall that Algorithm* LearnPoints *outputs a random $c_i \in \texttt{POINT}_X$ whenever $\mathcal{A}_{\text{dist}}$ outputs $\perp$. In order for this random c_i to be good (w.h.p.) we needed $|X|$ (i.e., the number of possible concepts) to be at least $2/(\alpha\beta)$. This requirement could be avoided by outputting the all-zero hypothesis $c_0 \equiv 0$ whenever $\mathcal{A}_{\text{dist}}$ outputs $\perp$. However, this approach results in a* proper *learner only if we add the all-zero concept to $\texttt{POINT}_X$.*

6.3 Towards a Private Learner for Thresholds

Recall the class $\texttt{THRESH}_X$ of all threshold functions over a (totally ordered) domain X:

$$\texttt{THRESH}_X = \{c_x : x \in X\} \quad \text{where} \quad c_x(y) = 1 \text{ iff } y \leq x.$$

Note that $\text{VC}(\texttt{THRESH}_X) = 1$, and, therefore, there exists a *proper* non-private learner for $\texttt{THRESH}_X$ with sample complexity $O_{\alpha,\beta}(1)$. As $|\texttt{THRESH}_X| = |X|$, one can use the generic construction of Kasiviswanathan et al. [67] and get a proper ε-private learner for this class

with sample complexity $O_{\alpha,\beta,\varepsilon}(\log|X|)$. Feldman and Xiao [51] showed that this is in fact optimal, and every pure ε-private learner for this class (proper or improper) must have sample complexity $\Omega(\log|X|)$.

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

In Section 6.4, we present a proper (ε,δ)-private learner for $\texttt{THRESH}_X$ with sample complexity (roughly) $2^{O(\log^*|X|)}$. We use this section for motivating the construction. We start with two simplifying assumptions. First, when given a labeled sample S, we aim at choosing a hypothesis $h \in \texttt{THRESH}_X$ approximately minimizing the empirical error (rather than the generalization error). Second, we assume that we are given a "diverse" sample S that contains many points labeled as 1 and many points labeled as 0. These two assumptions (and any other informalities made hereafter) will be removed in Section 6.4.

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

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

1. The interval G contains "a lot" of ones, *and* "a lot" of zeros in S.

2. Every interval $I \subseteq X$ of length $|I| \leq |G|/k$ does not contain, simultaneously, "too many" ones and "too many" zeros in S, where k is some constant.[1]

Such an interval will be referred to as a *k-good* interval. Note that a k-good interval is, in particular, an ℓ-good interval for every $\ell \geq k$. Figure 6.1 illustrates such an interval

[1] We identify the *length* of an interval $I \subseteq X$ with its size $|I|$.

G, where the dotted line represents the (unknown) target concept, and the bold dots correspond to sample points.

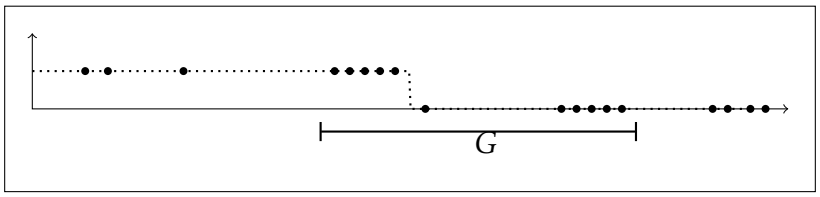

Figure 6.1: An illustration of a 4-good interval G.

Given such a 4-good interval G, we can (without using the sample S) define a set H of five hypotheses s.t. at least one of them has small empirical error. To see this, consider Figure 6.2, where G is divided into four equal intervals g_1, g_2, g_3, g_4, and five hypotheses $h_1, \ldots, h_5$ are defined s.t. the points where they switch from one to zero are located at the edges of g_1, g_2, g_3, g_4.

Now, as the interval G contains both ones and zeros, it must be that the target concept c_ℓ switches from 1 to 0 inside G. Assume without loss of generality that this switch occurs inside g_2. Note that g_2 is of length $|G|/4$ and, therefore, either does not contain too many ones, and h_2 is "close" to the target concept, or does not contain too many zeros, and h_3 is "close" to the target concept. For this argument to go through we need "not too many" to be smaller than αn (say $(3/4)\alpha n$), where α is our approximation parameter and n is the sample size.

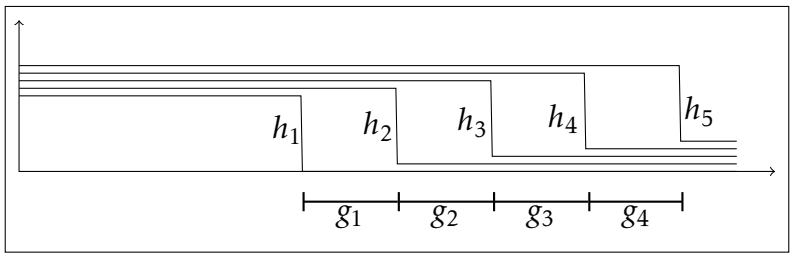

Figure 6.2: Extracting a small set of hypotheses from a good interval.

After defining such a set H, we could use the exponential mechanism to choose a hypothesis $h \in H$ with small empirical error on S. As the size of H is constant, this requires only a constant number of samples. To conclude, finding a 4-good interval G (while preserving privacy) is sufficient for choosing a good hypothesis. We next explain how to find such an interval.

Assume, for now, that we have a differentially private algorithm that given a sample S, returns an *interval length* J s.t. there exists a *2-good* interval $G \subseteq X$ of length $|G| = J$. This length J is used to find an explicit *4-good* interval as follows. Divide X into intervals $\{A_i\}$ of length $2J$, and into intervals $\{B_i\}$ of length $2J$ right shifted by J as in Figure 6.3.

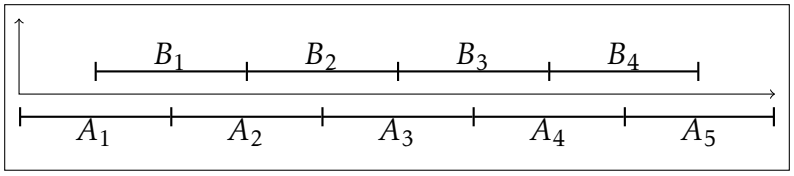

Figure 6.3: Dividing the axis X_d into intervals of length $2J$.

As the promised 2-good interval G is of length J, at least one of the these intervals contains G. We next explain how to privately choose such interval. If, e.g., $G \subseteq A_2$ then A_2 contains both a lot of zeros and a lot of ones. The target concept must switch inside A_2, and, therefore, every other $A_i \neq A_2$ cannot contain both zeros and ones. For every interval A_i, define its quality $q(A_i)$ to be the minimum between the number of zeros in A_i and the number of ones in A_i. Therefore, $q(A_2)$ is large, while $q(A_i) = 0$ for every $A_i \neq A_2$. That is, A_2 scores much better than any other A_i under this quality function q. The sensitivity of $q()$ is 1 and we can use algorithm $\mathcal{A}_{\text{dist}}$ to privately identify A_2. It suffices, e.g., that $q(A_2) \geq (1/4)\alpha n$, and we can, therefore, set our "a lot" bound to be $(1/4)\alpha n$. Recall that $G \subseteq A_2$ is a 2-good interval, and that $|A_2| = 2|G|$. The identified A_2 is, therefore, a 4-good interval.

To conclude, if we could indeed find (while preserving privacy) a length J s.t. there exists a 2-good interval G of that length, then our task would be completed.

Computing the interval length J. At first attempt, one might consider performing a binary search for such a length $1 \leq J \leq |X|$, in which every comparison will be made using the Laplace mechanism. More specifically, for every length $1 \leq J \leq |X|$, define

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

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

Note that $Q(\cdot)$ is a monotonically non-decreasing function, and that $Q(1) = 0$ (as in a correctly labeled sample a point cannot appear both with the label 1 and with the label 0). Recall our assumption that the sample S is "diverse" (contains many points labeled as 1 and many points labeled as 0), and, therefore, $Q(|X|)$ is large. Hence, there exists a J s.t. $Q(J)$ is "big enough" (say at least $(1/4)\alpha n$) while $Q(J-1)$ is "small enough" (say at most $(3/4)\alpha n$). That is, a J s.t. (1) there exists an interval of length J containing lots of ones and lots of zeros; and (2), every interval of length $< J$ cannot contain too many ones and too many zeros simultaneously. Such a J can easily be (privately) obtained using a (noisy) binary search. However, such a binary search on X requires $\log|X|$ noisy comparisons, which, in turn, requires a sample of size $\text{poly}(\log|X|)$ in order to achieve reasonable utility guarantees.

As a second attempt, one might consider performing a binary search, not on $1 \leq J \leq |X|$, but rather on the power j of an interval of length 2^j. That is, performing a search for a power $0 \leq j \leq \log|X|$ for which there exists a 2-good interval of length 2^j. Here there are only $\log\log|X|$ noisy comparisons, and the sample size is reduced to $\text{poly}(\log\log|X|)$. Again, a (noisy) binary search on $0 \leq j \leq \log|X|$ can (privately) yield an appropriate length $J = 2^j$ s.t. $Q(2^j)$ is "big enough", while $Q(2^{j-1})$ is "small enough". Such a $J = 2^j$ is, indeed, a length of a 2-good interval. Too see this, note that as $Q(2^j)$ is "big enough", there exists an interval of length 2^j containing lots of ones and lots of zeros. Moreover, as $Q(2^{j-1})$ is "small enough", every interval of length $2^{j-1} = (1/2)2^j$ cannot contain too many ones and too many zeros simultaneously.

Remark 6.4. *A binary search as above would have to operate on noisy values of $Q(\cdot)$ (as otherwise differential privacy cannot be obtained). For this reason, we set the bounds for "big enough" and "small enough" to overlap. Namely, we search for a value j such that $Q(2^j) \geq (\alpha/4)n$ and $Q(2^{j-1}) \leq (3\alpha/4)n$, where α is our approximation parameter, and n is the sample size.*

To summarize, using a binary search we find a length $J = 2^j$ such that there exists a 2-good interval of length J. Then, using $\mathcal{A}_{\text{dist}}$, we find a 4-good interval. Finally, we

partition this interval to 4 intervals, and using the exponential mechanism we choose a starting point or end point of one of these intervals as our the threshold.

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

6.4 Privately Approximating Quasi-Concave Promise Problems

We next define the notions that enable our recursive algorithm.

Definition 6.5. *A function $Q(\cdot)$ over a totally ordered domain is* quasi-concave *if $Q(\ell) \geq \min\{Q(i), Q(j)\}$ for every $i \leq \ell \leq j$.*

Definition 6.6 (Quasi-concave promise problems)**.** *A* quasi-concave promise problem *consists of an ordered set of possible solutions $[0, T] = \{0, 1, \ldots, T\}$, a database $S \in X^n$, a 1-sensitive quality function $Q : X^* \times [0, T] \to \mathbb{R}$, an approximation parameter α, and another parameter r (called a* quality promise*).*

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

Example 6.7. *Consider a sample $S = (x_i, y_i)_{i=1}^n$, labeled by some target function $c_j \in \mathsf{THRESH}_X$. The goal of choosing a hypothesis with small empirical error can be viewed as a quasi-concave promise problem as follows. Denote our totally ordered domain X as $X = \{0, 1, 2, \ldots, |X| - 1\}$, set the range of possible solutions to $[0, T] = X$, the approximation parameter to α and the quality promise to n. Define $Q(S, k) = |\{i : c_k(x_i) = y_i\}|$; i.e., $Q(S, k)$ is the number of points in S correctly classified by $c_k \in \mathsf{THRESH}_X$. Note that the target concept c_j satisfies $Q(S, j) = n$. Our task is to find a hypothesis $h_k \in \mathsf{THRESH}_X$ s.t. $\mathrm{error}_S(h_k) \leq \alpha$, which is equivalent to finding $k \in [0, |X| - 1]$ s.t. $Q(S, k) \geq (1 - \alpha)n$.*

To see that $Q(S, \cdot)$ is quasi-concave, let $u \leq v \leq w$ be s.t. $Q(S, u), Q(S, w) \geq \lambda$. See Figure 6.4 for an illustration. Consider j, the index of the target concept, and assume w.l.o.g. that $j \leq v$ (the

other case is symmetric). That is, $j \leq v \leq w$. Note that c_v errs only on points in between j and v, and c_w errs on all these points. That is, $\mathrm{error}_S(c_v) \leq \mathrm{error}_S(c_w)$*, and, therefore,* $Q(S, v) \geq \lambda$.

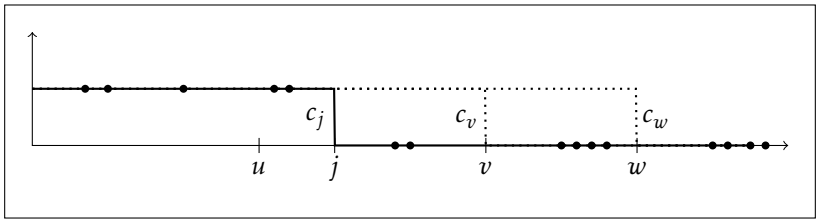

Figure 6.4: An illustration for Example 6.7. Here c_j is the target concept and the bold dots correspond to sample points. Note that c_w errs on every point on which c_v errs.

Remark 6.8. *Note that if the sample S in Example 6.7 is* not *consistent with any $c \in \mathsf{THRESH}_X$, then there is no j s.t. $Q(S, j) = n$, and the quality promise is void. Moreover, in such a case $Q(S, \cdot)$ might not be quasi-concave.*

We are interested in solving quasi-concave promise problems while preserving differential privacy. As motivated by Remark 6.8, privacy must be preserved even when $Q(S, \cdot)$ is not quasi-concave or $Q(S, p) < r$ for all $p \in [0, T]$. We are now ready to present and analyze our algorithm RecConcave (see inline comments for some of the underlying intuition).

Algorithm RecConcave

Inputs: Range $[0,T]$, quality function Q, quality promise r, parameters $\alpha,\varepsilon,\delta$, and a database S.

Optional Input: a bound $N \geq 1$ on the number of recursive calls (set $N = \infty$ otherwise).

1. If $[(T \leq 32)$ or $(N = 1)]$, then use the exponential mechanism with the quality function Q and the parameter ε to choose and return an index $j \in [0,\ldots,T]$. Otherwise set $N = N - 1$.

2. Let T' be the smallest power of 2 s.t. $T' \geq T$, and define $Q(S,i) = \min\{0, Q(S,T)\}$ for $T < i \leq T'$.

3. For $0 \leq j \leq \log(T')$ let
$$L(S,j) = \max_{\substack{[a,b]\subseteq[0,T'] \\ b-a+1=2^j}} \left(\min_{i\in[a,b]} \Big(Q(S,i) \Big) \right).$$
For $j = \log(T') + 1$ let $L(S,j) = \min\{0, L(S,\log(T')\}$.
   % If $L(S,j) = x$ then (1) there exists an interval $I \subseteq [0,T']$ of length 2^j s.t. $Q(S,i) \geq x$ for all $i \in I$; and (2) in every interval $I \subseteq [0,T']$ of length 2^j there exists a point $i \in I$ s.t. $Q(S,i) \leq x$. Note that $L(S,j+1) \leq L(S,j)$. See Figure 6.5 for an illustration.

4. Define the function $q(S,j) = \min\Big(L(S,j) - (1-\alpha)r, r - L(S,j+1)\Big)$ where $0 \leq j \leq \log(T')$.
   % If $q(S,j)$ is high for some j, then there exists an interval $I = [a, a+2^j-1]$ s.t. every $i \in I$ has a quality $Q(S,i) >> (1-\alpha)r$, and for every interval $I' = [a', a'+2^{j+1}-1]$ there exists $i' \in I'$ with quality $Q(S,i) \ll r$. See Figure 6.5.

5. Let $R = \dfrac{\alpha}{2} r$.
   % R is the promise parameter for the recursive call. Note that for the maximal j with $L(S,j) \geq (1-\frac{\alpha}{2})r$ we get $q(S,j) \geq \frac{\alpha}{2} r = \mathrm{R}$.

6. Execute RecConcave recursively on the range $\{0,\ldots,\log(T')\}$, the quality function $q(\cdot,\cdot)$, the promise R, an approximation parameter 1/4, and ε,δ,N. Denote the returned value by k, and let $K = 2^k$.
   % If the call to RecConcave was successful, then k is s.t. $q(S,k) \geq (1-\frac{1}{4})R = \frac{3\alpha}{8} r$. That is, $L(S,k) \geq (1-\frac{5\alpha}{8})r$ and $L(S,k+1) \leq (1-\frac{3\alpha}{8})r$. Note in the top level call the approximation parameter α is arbitrary (given as input), and that in all of the lower level calls the approximation parameter is fixed at $\frac{1}{4}$.

Algorithm RecConcave (continued)

7. Divide $[0, T']$ into the following intervals of length $8K$ (the last ones might be trimmed):

$$A_1 = [0, 8K-1],\ A_2 = [8K, 16K-1],\ A_3 = [16K, 24K-1], \ldots$$
$$B_1 = [4K, 12K-1],\ B_2 = [12K, 20K-1],\ B_3 = [20K, 28K-1], \ldots$$

   % We show that in at least one of those two partitions (say the $\{A_i\}$), there exists a good interval A_g s.t. $Q(S,i) = r$ for some $i \in A_g$, and $Q(S,i) \leq (1 - \frac{3\alpha}{8})r$ for all $i \in \{0, \ldots, T\} \setminus A_g$.

8. For every such interval $I \in \{A_i\} \cup \{B_i\}$ let $u(S, I) = \max_{i \in I} \Big(Q(S, i) \Big)$.

9. Use algorithm $\mathcal{A}_{\text{dist}}$ twice with parameters ε, δ and the quality function $u(\cdot, \cdot)$, once to choose an interval $A \in \{A_i\}$, and once more to choose an interval $B \in \{B_i\}$.

   % By the properties of $\mathcal{A}_{\text{dist}}$, w.h.p. at least one of the returned A and B is good.

10. Use the exponential mechanism with the quality function $Q(\cdot, \cdot)$ and parameter ε to choose and return an index $j \in (A \cup B)$.

    % We show that a constant fraction of the solutions in $(A \cup B)$ have high qualities, and, hence, the exponential mechanism needs only a constant sample complexity in order to achieve good utility guarantees.

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

Notation. Given an integer n, let $\log^{\lceil * \rceil}(n)$ denote the number of times that the function $\lceil \log(x) \rceil$ must be iteratively applied before the result is less than or equal to 1, i.e., $\log^{\lceil * \rceil}(n) = 1 + \log^{\lceil * \rceil}\lceil \log(n) \rceil$ if $n > 1$ and zero otherwise. Observe that $\log^{\lceil * \rceil}(n) = \log^*(n)$.[2]

Observation 6.9. *On a range* $[0, T]$ *there could be at most* $\log^{\lceil * \rceil}(T) = \log^*(T)$ *recursive calls throughout the execution of* RecConcave.

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

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

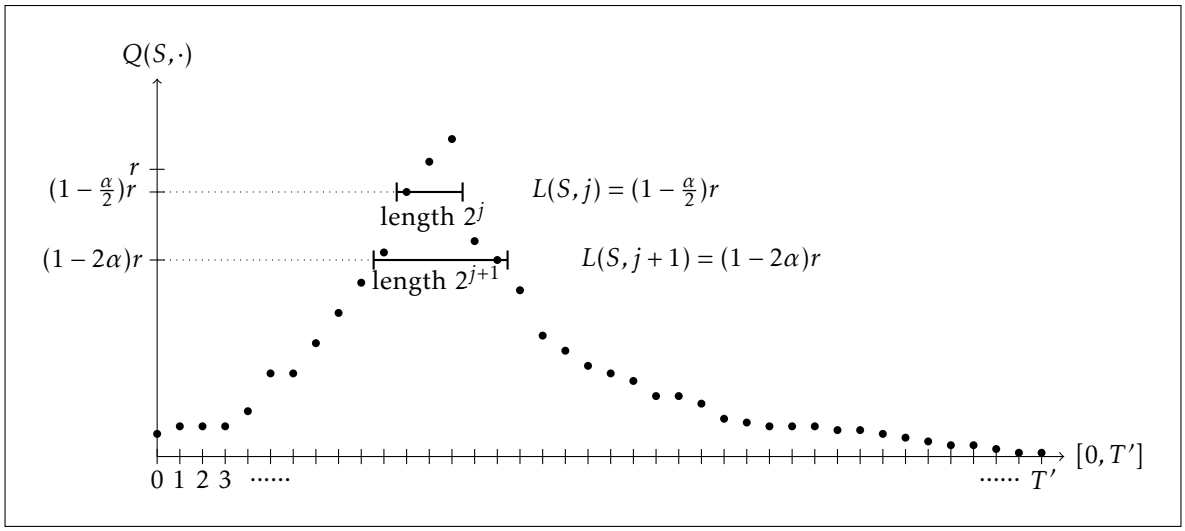

Figure 6.5: A demonstration for the functions L and q from steps 3,4 of RecConcave. In the illustration, every interval of length 2^j contains at least one point with quality at most $(1-\alpha/2)r$, and there exists an interval of length 2^j containing only points with quality at least $(1-\alpha/2)r$. Hence, $L(S,j)=(1-\alpha/2)r$. Similarly, $L(S,j+1)=(1-2\alpha)r$. Therefore, for this j we have that $q(S,j)=\min\{L(S,j)-(1-\alpha)r, r-L(S,j+1)\}=(\alpha/2)r$. The reason for defining $q(\cdot,\cdot)$ is the following. We were interested in identifying a j with an appropriate lower bound on $L(S,j)$ and with an appropriate upper bound on $L(S,j+1)$. That is, in order to decide whether a given j is a good, we need to check both $L(S,j)$ and $L(S,j+1)$. After defining $q(S,\cdot)$, we can simply look for a j with a high $q(S,j)$. A high $q(S,j)$ implies upper and lower bounds (respectively) on $L(S,j), L(S,j+1)$.

Observation 6.10. *Let $\{f_1,f_2,\ldots,f_N\}$ be a set of 1-sensitive functions mapping X^* to $\mathbb{R}$. Then $f_{max}(S)=\max_i\{f_i(S)\}$ and $f_{min}(S)=\min_i\{f_i(S)\}$ are 1-sensitive functions.*

We now proceed with the privacy analysis of algorithm RecConcave.

Lemma 6.11. *When executed on a 1-sensitive quality function Q, parameters ε,δ, and a bound on the recursion depth N, algorithm RecConcave preserves $(3N\varepsilon,3N\delta)$-differential privacy.*

Proof. Note that since Q is a 1-sensitive function, all of the quality functions defined throughout the execution of RecConcave are of sensitivity 1 (see Observation 6.10). In each recursive call algorithm RecConcave invokes at most three differentially private mechanisms—once with the exponential mechanism (in step 1 or in step 11), and at most twice with algorithm $\mathcal{A}_{\text{dist}}$ (in step 9). As there are at most N recursive calls, we conclude that throughout the entire execution algorithm RecConcave invokes most $3N$

mechanisms, each (ε, δ)-differentially private. Hence, using composition (Theorem 3.5), algorithm RecConcave is $(3N\varepsilon, 3N\delta)$-differentially private. □

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

Claim 6.12. *Let $Q : X^* \times [0,T] \to \mathbb{R}$ be a quality function, and let the functions $L(\cdot,\cdot)$ and $q(\cdot,\cdot)$ be as in steps 3, 4 of algorithm RecConcave. Then $q(S,\cdot)$ is quasi-concave for every $S \in X^*$.*

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

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

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

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

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

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

Observe that for every $N \leq \log^*(n)$ we have that[3]

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

Lemma 6.13. *Let $Q : X^* \times [0,T] \to \mathbb{R}$ be a 1-sensitive quality function, and let $S \in X^*$ be a database s.t. $Q(S,\cdot)$ is quasi-concave. Let $\alpha \leq 1/2$ and let $\beta, \varepsilon, \delta, r, N$ be s.t.*

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

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

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

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

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

Assume that the stated lemma holds whenever algorithm RecConcave performs at most $t-1$ recursive calls, and let $S, [0,T], r, \alpha, \varepsilon, \delta, N$ be inputs (satisfying the conditions of Lemma 6.13) on which algorithm RecConcave performs t recursive calls. Consider the first call in the execution of RecConcave on those inputs, and denote by T' the smallest power of 2 s.t. $T' \geq T$. In order to apply the inductive assumption, we need to show that for the recursive call in step 6, all the conditions of Lemma 6.13 hold.

We first note that by Claim 6.12, the quality function $q(S,\cdot)$ defined in step 4 is quasi-concave. We next show that the recursive call is performed with an appropriate quality promise $R = (\alpha/2)r$. The conditions of the lemma ensure that $L(S,0) \geq r$, and, by definition,

[3] For example

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

we have that $L(S, \log(T')+1) \le 0$. There exists therefore a $j \in [0, \log(T')]$ for which $L(S,j) \ge (1-\alpha/2)r$, and $L(S, j+1) < (1-\alpha/2)r$. Plugging these inequalities in the definition of $q(S,j)$ we get that $q(S,j) \ge (\alpha/2)r$. Therefore, there exists an index $j \in [0, \log(T')]$ with quality $q(S,j) \ge R$. Moreover, the recursive call of step 6 executes RecConcave on the range $[0, \log(T')] = [0, \lceil \log(T) \rceil]$ with $(N-1)$ as the bound on the recursion depth, with $\widetilde{\alpha} \triangleq 1/4$ as the approximation parameter, and with a quality promise R satisfying

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

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

As promised by the conditions of the lemma, there exists a point $p \in [0,T]$ with quality $Q(S,p) \ge r$. Consider the following two intervals: $P_1 = [p - 2\cdot 2^k + 1, p]$ and $P_2 = [p, p + 2\cdot 2^k - 1]$, and denote $P = P_1 \cup P_2$ (these two intervals might be trimmed if p is close to the edges of $[0,T]$). Assuming P_1, P_2 are not trimmed, they both are intervals of length $2\cdot 2^k$, and, therefore, each of them contains a point i_1, i_2, respectively, with quality $Q(S,i_1), Q(S,i_2) \le (1-3\alpha/8)r$. Therefore, by the quasi-concavity of $Q(S,\cdot)$, every point $\ell \ge i_2$ and every point $\ell \le i_1$ must have quality at most $Q(S,\ell) \le (1-3\alpha/8)r$ (otherwise, by the quasi-concavity of $Q(S,\cdot)$, every point between p and ℓ must have quality strictly greater than $(1-3\alpha/8)r$, contradicting the quality bound on i_1, i_2). See Figure 6.6.

Note that if P_1 (or P_2) is trimmed, then there are no points on the left of (or on the

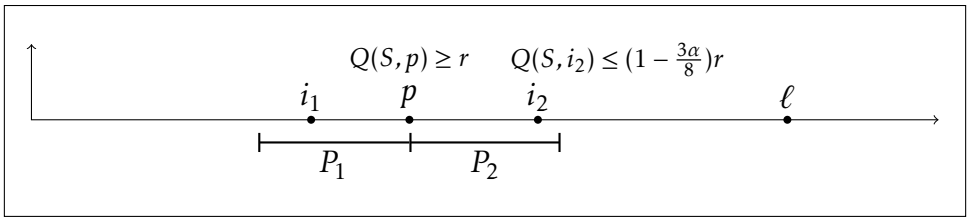

Figure 6.6: A point $\ell \notin P$ cannot have quality greater than $(1-\frac{3\alpha}{8})r$.

right of) P. So, the interval P contains the point p with quality $Q(S,p) \geq r$ and every point $i \in [0,T] \setminus P$ has quality of at most $(1-3\alpha/8)r$. Moreover, P is of length $4 \cdot 2^k - 1$. As the intervals of the partitions $\{A_i\}$ and $\{B_i\}$ are of length $8 \cdot 2^k$, and the $\{B_i\}$ are shifted by $4 \cdot 2^k$, there must exist an interval $C \in \{A_i\} \cup \{B_i\}$ s.t. $P \subseteq C$. Assume without loss of generality that $C \in \{A_i\}$.

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

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

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

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

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

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

the exponential mechanism ensures that the probability of step 10 failing to return a point $h \in (A \cup B)$ with $Q(S,h) \geq (1-\alpha)r$ is at most β: As there are at least $(1/16)|A \cup B|$ solutions with quality at least $(1-5\alpha/8)r$, the probability that the exponential mechanism outputs a specific solution $h \in (A \cup B)$ with $Q(S,h) \leq (1-\alpha)r$ is at most

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

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

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

which is at most β for our choice of r.

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

Combining Lemma 6.11 and Lemma 6.13 we get the following theorem.

Theorem 6.14. *Let algorithm RecConcave be executed on a range $[0,T]$, a 1-sensitive quality function Q, a database S, a bound on the recursion depth N, privacy parameters $\varepsilon/(3N)$, $\delta/(3N)$, approximation parameter α, and a quality promise r. The following two statements hold:*

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

$$\max_{i\in[0,T]}\{Q(S,i)\} \geq r \geq 8^N \cdot \frac{36N}{\alpha\varepsilon}\left\{\log\left(\frac{6N}{\beta\delta}\right) + \underbrace{\log\log\cdots\log}_{N \text{ times}}(T)\right\}, \tag{6.1}$$

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

Remark 6.15. *Recall that the number of recursive calls on a range $[0,T]$ is always bounded by $\log^*(T)$, and note that for $N = \log^*(T)$ we have that $\log^{\lceil N\rceil}(T) \leq 1$. Therefore, the promise*

requirement in Inequality (6.1) can be replaced with

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

Remark 6.16. *The computational efficiency of algorithm* RecConcave *depends on the quality function $Q(\cdot,\cdot)$. Note, however, that it suffices to efficiently implement the top level call (i.e., without the recursion). This is true because an iteration of algorithm* RecConcave*, operating on a range $[0,T]$, can easily be implemented in time* $\mathrm{poly}(T)$*, and the range given as input to recursive calls is logarithmic in the size of the initial range.*

6.5 Approximate-Private Proper-Learner for Thresholds

As we will now see, algorithm RecConcave can be used as a proper $(\alpha,\beta,\varepsilon,\delta,m)$-private learner for $\mathtt{THRESH}_X$. Recall Example 6.7 (showing that the goal of choosing a hypothesis with small empirical error can be viewed as a quasi-concave promise problem), and consider the following algorithm.

Algorithm LearnThresholds

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

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

Theorem 6.17. *For every $1 \le N \le \log^*|X|$, algorithm* LearnThresholds *is an efficient proper $(\alpha,\beta,\varepsilon,\delta)$-PPAC learner for $\mathtt{THRESH}_X$ with sample size n, where the sample size is*

$$n = O\left(8^N \cdot \frac{N}{\alpha\varepsilon}\left\{\log\left(\frac{N}{\beta\delta}\right) + \underbrace{\log\log\cdots\log}_{N \text{ times}}|X|\right\}\right).$$

Proof. By Theorem 6.14, algorithm LearnThresholds is (ε, δ)-differentially private. We now proceed with the utility analysis. Recall that by Lemma 3.19 (generalization bound for thresholds) it suffices to show that, for every input sample, algorithm LearnThresholds outputs w.h.p. a hypothesis with small empirical error. To that end, fix a target concept $c_j \in \mathsf{THRESH}_X$, and let S be an arbitrary database containing n examples from X, labeled by c_j.

Observe that for the target concept c_j we have $Q(S, j) = n$, and algorithm RecConcave is executed in step 3 with a valid quality promise. Moreover, as shown in Example 6.7, algorithm RecConcave is executed with a quasi-concave quality function.

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

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

algorithm RecConcave ensures that with probability at least $(1-\beta/2)$, the index k at step 2 is s.t. $Q(k) \geq (1-\alpha/2)n$. The empirical error of c_k is at most $\alpha/2$ in such a case. Therefore, with probability at least $(1-\beta/2)$, algorithm LearnThresholds outputs a hypothesis with error at most $\alpha/2$. Thus, by Lemma 3.19 (generalization bound for threshold functions), we conclude that algorithm LearnThresholds is a proper $(\alpha, \beta, \varepsilon, \delta)$-PPAC learner for THRESH_X with sample size n, where

$$n \geq 8^N \cdot \frac{72N}{\alpha\varepsilon}\left\{\log\left(\frac{12N}{\beta\delta}\right) + \underbrace{\log\log\cdots\log}_{N \text{ times}}|X|\right\}.$$

□

Remark 6.18. *By using $N = \log^*|X|$ in Theorem 6.17, we can bound the sample complexity of LearnThresholds by*

$$n = O\left(\frac{8^{\log^*|X|} \cdot \log^*|X|}{\alpha\varepsilon}\log\left(\frac{\log^*|X|}{\beta\delta}\right)\right).$$

6.6 Axis-Aligned Rectangles in High Dimension

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

Definition 6.19. *For a totally ordered domain X and $\vec{a} = (a_1, \ldots, a_\ell), \vec{b} = (b_1, \ldots, b_\ell) \in X^\ell$ define the concept $c_{[\vec{a},\vec{b}]} : X^\ell \to \{0,1\}$ where $c_{[\vec{a},\vec{b}]}(\vec{x}) = 1$ if and only if for every $1 \leq i \leq \ell$ we have $a_i \leq x_i \leq b_i$. Define the concept class of all axis-aligned rectangles over X^ℓ as $\texttt{RECTANGLE}_X^\ell = \{c_{[\vec{a},\vec{b}]}\}_{\vec{a},\vec{b} \in X^\ell}$.*

6.6.1 Preliminaries from Private Query Release

Before formally presenting our learner for $\texttt{RECTANGLE}_X^\ell$, we introduce additional preliminaries from private query release.

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

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

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

Definition 6.20. *Let C be a concept class and let S be a database. A function $\mathrm{Est} : C \to [0,1]$ is called α-*close *to S if $|Q_c(S) - \mathrm{Est}(c)| \leq \alpha$ for every $c \in C$. If, furthermore, Est is defined in terms of a database $\hat{S}$, i.e., $\mathrm{Est}(c) = Q_c(\hat{S})$, we say that $\hat{S}$ is α-close to S.*

Definition 6.21. *Let C be a class of concepts mapping X to $\{0,1\}$. Let $\mathcal{A}$ be an algorithm that on an input database $S \in X^*$ outputs a description of a function $\mathrm{Est} : C \to [0,1]$. Algorithm $\mathcal{A}$ is an $(\alpha, \beta, \varepsilon, \delta, n)$-query release mechanism for predicates in the class C, if*

1. *$\mathcal{A}$ is (ε, δ)-differentially private;*

2. *For every input* $S \in X^n$, *we have* $\Pr_{\mathcal{A}}[\text{Est } \textit{is } \alpha\textit{-close to } S] \geq 1 - \beta$.

The probability is over the coin tosses of algorithm $\mathcal{A}$. *If* $\mathcal{A}$*'s output is defined by a database* $\hat{S} \in X^*$, *and* $\text{Est}(\cdot)$ *is defined as* $\text{Est}(c) = Q_c(\hat{S})$, *then algorithm* $\mathcal{A}$ *is said to produce* synthetic data.

Remark 6.22. *Note that without the privacy requirements producing a synthetic database is a trivial task as it is possible to simply output the input database* S. *Furthermore, ignoring computational complexity, an* $(\alpha, \beta, \varepsilon, \delta, n)$*-query release mechanism can always be transformed to produce synthetic data, by finding a database* $\hat{S}$ *that is* α*-close to* Est. *Such a database must exist (as in particular* S *is* α*-close to* Est*), and is* 2α*-close to* S *(by the triangle inequality).*

6.6.2 Privately Learning $\texttt{RECTANGLE}_X^\ell$

Observe that the VC dimension of $\texttt{RECTANGLE}_X^\ell$ is 2ℓ, and, thus, it can be learned non-privately with sample complexity $O_{\alpha,\beta}(\ell)$. Note that $|\texttt{RECTANGLE}_X^\ell| = |X|^{O(\ell)}$, and, therefore, the generic construction of Kasiviswanathan et al. [67] yields an inefficient proper ε-private learner for this class with sample complexity $O_{\alpha,\beta,\varepsilon}(\ell \cdot \log|X|)$.

In [68], Kearns gave an efficient (noise resistant) non-private learner for this class. The learning model there was a variant of the statistical queries model [68], in which the learner is also being given access to the underling distribution $\mathcal{D}$. Every learning algorithm in the statistical queries model can be transformed to satisfy differential privacy while preserving efficiency [15, 67]. However, as Kearns' algorithm assumes direct access to $\mathcal{D}$, this transformation cannot be applied directly.

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

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

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

Note that once the interval sets $\mathcal{I}_i$ are defined for each axis $i \in [\ell]$, estimating every single p_I can be done via statistical queries, and can, therefore, be made private using the transformation of [15, 67]. Alternatively, estimating (simultaneously) all the p_I (on the i^{th} axis) could be done privately using the Laplace mechanism. This use of the Laplace mechanism is known as a histogram (see Theorem 3.25).

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

We now overcome this issue using a query release mechanism for $\texttt{THRESH}_X$. Such a mechanism is constructed in the full version of this work [11]; here we use it for privately finding $\mathcal{I}_i$.

Theorem 6.23. *Fix $\alpha, \beta, \varepsilon, \delta$. There exists an efficient $(\alpha, \beta, \varepsilon, \delta, n)$-query release mechanism for $\texttt{THRESH}_X$, where*

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

Moreover, the algorithm produces synthetic data.

As we next explain, such a query release mechanism can be used to (privately) divide the axes. Given an interval $[a,b] \subseteq X$ and a sample S, we denote the probability mass of $[a,b]$ under $\mathcal{D}$ as $\mathcal{D}[a,b]$, and the number of sample points in this interval as $\#_S[a,b]$. Standard arguments in learning theory (specifically, Theorem 3.17) state that for a large enough sample (whose size is bigger than the VC dimensions of the intervals class) w.h.p. $(1/|S|)\,\#_S[a,b] \approx \mathcal{D}[a,b]$ for *every* interval $[a,b] \subseteq X$.

On an input database $S \in X^*$, our query release mechanism for $\texttt{THRESH}_X$ outputs an alternative database $\hat{S} \in X^*$ s.t.

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

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

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

So, in order to divide the i^{th} axis we apply our query release mechanism for $\texttt{THRESH}_X$, and divide the axis using the returned synthetic database. In order to accumulate error of up to α/ℓ on each axis (as required by Kearns' algorithm), we need to execute the query release mechanism with an approximation parameter of (roughly) α/ℓ. Every such execution requires, therefore, a sample of $\widetilde{O}_{\alpha,\beta,\varepsilon,\delta}\left(\ell^{2.5} \cdot 8^{\log^* |X|}\right)$ elements. As there are ℓ such executions (one for each axis), using composition (Theorem 3.6), the described learner is of sample complexity $\widetilde{O}_{\alpha,\beta,\varepsilon,\delta}\left(\ell^{3} \cdot 8^{\log^* |X|}\right)$.

Theorem 6.24. *There exists an efficient $(\alpha,\beta,\varepsilon,\delta)$-PPAC proper-learner for $\texttt{RECTANGLE}_X^\ell$ with sample size n, where*

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

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

[4]The general construction of Kasiviswanathan et al. [67] yields an (inefficient) pure-private proper-learner for this class with sample complexity $O_{\alpha,\beta,\varepsilon}(\ell \cdot \log|X|)$. Feldman and Xiao [51] showed that this is in fact optimal, and every ε-private (proper or improper) learner for this class must have sample complexity $\Omega(\ell \cdot \log|X|)$.

Chapter 7

Private Learning of Threshold Functions

In Chapter 6 we showed that relaxing the privacy requirement from *pure* to *approximate* differential privacy can drastically reduce the sample complexity of private learners. Namely, we showed positive results for approximate-private proper-learners with sample complexity that either matches or comes very close to that of non-private learning. In light of these positive results, one might hope that the sample complexity of approximate-private proper-learning is actually characterized by the VC dimension and is of the same order as that of non-private learning. In this chapter we show that this is not the case.

7.1 Main Results

We revisit the task of properly learning threshold functions, and show that accomplishing this task with differential privacy is *impossible* when the data universe is infinite (e.g., $\mathbb{N}$ or $[0,1]$). In fact, we show that the sample complexity must grow with the size $|X|$ of the data universe: $n = \Omega(\log^* |X|)$, which is tantalizingly close to the upper bound of $n = 2^{O(\log^* |X|)}$ from Chapter 6.

Theorem 7.1. *The sample complexity of* properly *learning threshold functions over a data universe X with approximate differential privacy is at least* $\Omega(\log^* |X|)$.

This is the first non-trivial lower bound on the sample complexity of approximate-private learners. As the VC dimension of the threshold functions is 1, this result separates the sample complexity of approximate-private proper-learners from that of non-private ones. In particular, properly-learning threshold functions over an infinite domain is impossible under approximate differential privacy.

Inspired by the ideas in our lower bound, we present a simplification of the algorithm from Chapter 6 and improve the sample complexity to $2^{(1+o(1))\log^*|X|}$ (from roughly $8^{\log^*|X|}$). Our algorithm is also computationally efficient, running in time nearly linear in the bit-representation of its input database. Closing the gap between the lower bound of $\approx \log^*|X|$ and the upper bound of $\approx 2^{\log^*|X|}$ remains an intriguing open problem.

Our lower bound extends to the concept class of ℓ-dimensional thresholds. An ℓ-dimensional threshold function, defined over the domain X^ℓ, is a conjunction of ℓ threshold functions, each defined on one component of the domain. This shows that our separation between the sample complexity of private and non-private learning applies to concept classes of every VC dimension.

Theorem 7.2. *For every finite, totally ordered X and $\ell \in \mathbb{N}$, the sample complexity of properly learning the class C of ℓ-dimensional threshold functions on X^ℓ with differential privacy is at least $\Omega(\ell \cdot \log^*|X|) = \Omega(\mathrm{VC}(C) \cdot \log^*|X|)$.*

Based on these results, it would be interesting to fully characterize the difference between the sample complexity of proper non-private learners and of proper learners with (approximate) differential privacy. Furthermore, our results still leave open the possibility that *improper* PAC learning with (approximate) differential privacy has sample complexity $O(\mathrm{VC}(C))$. We consider this to be an important question for future work.

7.1.1 Techniques

Our results for proper learning of threshold functions are obtained by analyzing the sample complexity of a related but simpler problem, which we call the *interior-point problem*. Here we want a mechanism $\mathcal{M} : X^n \to X$ (for a totally ordered data universe X) such that for every database $S \in X^n$, with high probability we have $\min_i S_i \leq \mathcal{M}(S) \leq \max_i S_i$. We give reductions showing that the sample complexity of this problem is equivalent to that of proper learning of threshold functions:

Theorem 7.3. *Over every totally ordered data universe X, the interior-point problem and proper PAC learning of threshold functions have the same sample complexity (up to constant factors) under differential privacy.*

Thus we obtain our lower bound and our simplified and improved upper bound for proper learning by proving such bounds for the interior-point problem, such as:

Theorem 7.4. *The sample complexity for solving the interior-point problem over a data universe X with differential privacy is* $\Omega(\log^*|X|)$.

Note that for every fixed distribution $\mathcal{D}$ over X there exists a simple differentially private algorithm for solving the interior-point problem (w.h.p.) over databases sampled i.i.d. from $\mathcal{D}$ – simply output a point z s.t. $\Pr_{x\sim\mathcal{D}}[x \geq z] = 1/2$. Hence, in order to prove Theorem 7.4, we show a distribution $\mathcal{D}$ over *databases* of size $n \approx \log^*|X|$ on which privately solving the interior-point problem is impossible. The construction is recursive: we use a hard distribution over databases of size $(n-1)$ over a data universe of size logarithmic in $|X|$ to construct the hard distribution over databases of size n over X.

7.1.2 Additional Results

In the full version of this work [22] we extend our results in the following directions.

Private Query Release. Given a set Q of queries $q : X^n \to \mathbb{R}$, the *query release* problem for Q is to output accurate answers to all queries in Q. That is, we want a differentially private algorithm $\mathcal{M} : X^n \to \mathbb{R}^{|Q|}$ such that for every database $S \in X^n$, with high probability over $y \leftarrow \mathcal{M}(S)$, we have $y_q \approx q(S)$ for all $q \in Q$. Similarly for private learning of threshold functions, we show new upper and lower bounds on the necessary database size for private query release of threshold functions.

Private Distribution Learning. A fundamental problem in statistics is *learning distributions*, which is the task of learning an unknown distribution $\mathcal{D}$ given i.i.d. samples from it. The query release problem for threshold functions is closely related to the problem of learning an arbitrary distribution $\mathcal{D}$ on X up to small error in Kolmogorov (or CDF) distance. While closeness in Kolmogorov distance is a relatively weak measure of closeness for distributions, under various structural assumptions (e.g., the two distributions have probability mass functions that cross in a constant number of locations), it implies closeness in the much stronger notion of total variation distance. Other works have developed

additional techniques that use weak hypotheses learned under Kolmogorov distance to test and learn distributions under total variation distance (e.g., [31, 30, 32]).

The Dvoretzky-Kiefer-Wolfowitz inequality [37] gives a probabilistic bound on the Kolmogorov distance between a distribution and the *empirical distribution* of i.i.d. samples. It implies that without privacy, any distribution over X can be learned to within arbitrarily small constant error via the empirical CDF of $O(1)$ samples. On the other hand, we show that with approximate differential privacy, distribution learning instead requires sample complexity that grows with the size of the domain.

7.2 The Interior Point Problem – Lower Bound

In this work we exhibit a close connection between the problem of privately learning threshold functions and solving the *interior point problem* as defined below.

Definition 7.5. *An algorithm $\mathcal{A} : X^n \to X$ solves the* interior point problem *on X with error probability β if for every $S \in X^n$,*

$$\Pr[\min S \leq \mathcal{A}(S) \leq \max S] \geq 1 - \beta,$$

where the probability is taken over the coins of $\mathcal{A}$. The sample complexity of the algorithm $\mathcal{A}$ is the database size n.

We call a solution x with $\min S \leq x \leq \max S$ an *interior point* of S. Note that x need not be a member of the database S.

We now prove our lower bound for the interior point problem, and show that solving the interior point problem on a domain X under approximate-differential privacy requires sample complexity

$$n = \Omega(\log^* |X|).$$

In the full version of this work [22], we also show how this lower bound follows from the construction of a new combinatorial object we call an "interior point fingerprinting code". This is a variant on traditional fingerprinting codes, which have been used recently to show nearly optimal lower bounds for other problems in approximate differential privacy [24, 49, 7].

To prove our lower bound, we inductively construct a sequence of database distributions $\{\mathcal{D}_n\}$ supported on data universes $[T(n)] = \{0, 1, 2, \ldots, T(n)-1\}$ (for $T(n+1) = 2^{\tilde{O}(T(n))}$) over which any differentially private mechanism using n samples must fail to solve the interior point problem. Given a hard distribution $\mathcal{D}_n$ over n elements $(x_1, x_2, \ldots, x_n)$ from $[T(n)]$, we construct a hard distribution $\mathcal{D}_{n+1}$ over elements $(y_0, y_1, \ldots, y_n)$ from $[T(n+1)]$ by setting y_0 to be a random number, and letting each other y_i agree with y_0 on the x_i most significant digits. We then show that if y is the output of any differentially private interior point mechanism on $(y_0, \ldots, y_n)$, then with high probability, y agrees with y_0 on at least $\min x_i$ entries and at most $\max x_i$ entries. Thus, a private mechanism for solving the interior point problem on $\mathcal{D}_{n+1}$ can be used to construct a private mechanism for $\mathcal{D}_n$, and so $\mathcal{D}_{n+1}$ must also be a hard distribution.

Our lower bound for the interior point problem depends on a number of parameters we now define. Fix $0 < \varepsilon < \frac{1}{4}$. Let $\delta(n)$ be any positive non-increasing sequence s.t. for every n it holds that

$$P_n \triangleq \frac{e^\varepsilon}{e^\varepsilon + 1} + (e^\varepsilon + 1) \sum_{j=1}^{n} \delta(j) < \frac{3}{4}.$$

In particular, for $0 < \varepsilon < \frac{1}{4}$, it suffices that $\sum_{n=1}^{\infty} \delta(n) \leq \frac{1}{20}$, which it true for $\delta(n) \leq 1/(50n^2)$. Let $b(n) = 1/\delta(n)$ and define the function T recursively by

$$T(1) = 2 \quad \text{and} \quad T(n+1) = b(n+1)^{T(n)}.$$

Lemma 7.6. *Fix any constant $0 < \varepsilon < 1/4$. Let $\delta(n) \leq 1/(50n^2)$. Then for every positive integer n, there exists a distribution $\mathcal{D}_n$ over databases $S \in [T(n)]^n = \{0, 1, \ldots, T(n)-1\}^n$ such that for every $(\varepsilon, \delta(n))$-differentially private mechanism $\mathcal{M}$,*

$$\Pr[\min S \leq \mathcal{M}(S) \leq \max S] \leq P_n,$$

where the probability is taken over $S \leftarrow_R \mathcal{D}_n$ and the coins of $\mathcal{M}$.

In words, every $(\varepsilon, \delta(n))$-differentially private mechanism with sample complexity n for solving the interior point problem on a domain of size $T(n)$ has success probability strictly less than 3/4. We remark that our choice of $\delta = O(1/n^2)$ is unimportant; any monotonically non-increasing convergent series will do.

Proof of Lemma 7.6. The proof is by induction on n. We first argue that the claim holds for $n = 1$ by letting $\mathcal{D}_1$ be uniform over the singleton databases (0) and (1). To that end let $x \leftarrow_R \mathcal{D}_1$ and note that for any $(\varepsilon, \delta(1))$-differentially private mechanism $\mathcal{M}_0 : \{0,1\} \to \{0,1\}$ it holds that

$$\Pr[\mathcal{M}_0(x) = x] \leq e^{\varepsilon} \Pr[\mathcal{M}_0(\overline{x}) = x] + \delta(1) = e^{\varepsilon}(1 - \Pr[\mathcal{M}_0(x) = x]) + \delta(1),$$

giving the desired bound on $\Pr[\mathcal{M}_0(x) = x]$.

Now inductively suppose we have a distribution $\mathcal{D}_n$ that satisfies the claim. We construct a distribution $\mathcal{D}_{n+1}$ on databases $(y_0, y_1, \ldots, y_n) \in [T(n+1)]^{n+1}$ that is sampled as follows:

- Sample $(x_1, \ldots, x_n) \leftarrow_R \mathcal{D}_n$.
- Sample a uniformly random y_0 from $[T(n+1)]$. We write the base $b(n+1)$ representation of y_0 as $y_0^{(1)} y_0^{(2)} \ldots y_0^{(T(n))}$.
- For each $i = 1, \ldots, n$ let y_i be a base $b(n+1)$ number (written $y_i^{(1)} y_i^{(2)} \ldots y_i^{(T(n))}$) that agrees with the base $b(n+1)$ representation of y_0 on the first x_i digits and contains a random sample from $[b(n+1)]$ in every index thereafter.

Suppose for the sake of contradiction that there was an $(\varepsilon, \delta(n+1))$-differentially private mechanism $\hat{\mathcal{M}}$ that could solve the interior point problem on $\mathcal{D}_{n+1}$ with probability greater than P_{n+1}. We use $\hat{\mathcal{M}}$ to construct the following private mechanism $\mathcal{M}$ for solving the interior point problem on $\mathcal{D}_n$, giving the desired contradiction:

Algorithm $\mathcal{M}(S)$

Input: Database $S = (x_1, \ldots, x_n) \in [T(n)]^n$

1. Construct $\hat{S} = (y_0, \ldots, y_n)$ by sampling from $\mathcal{D}_{n+1}$, but starting with the database S. That is, sample y_0 uniformly at random and set every other y_i to be a random base $b(n+1)$ string that agrees with y_0 on the first x_i digits.
2. Compute $y \leftarrow \hat{\mathcal{M}}(\hat{S})$.
3. Return the length of the longest prefix of y (in base $b(n+1)$ notation) that agrees with y_0.

The mechanism $\mathcal{M}$ is also $(\varepsilon, \delta(n+1))$-differentially private, since for all pairs of adjacent databases S, S' and every $F \subseteq [T(n)]$,

$$\begin{aligned}
\Pr[\mathcal{M}(S) \in F] &= \mathop{\mathbb{E}}_{y_0 \leftarrow_R [T(n+1)]} \Pr[\hat{\mathcal{M}}(\hat{S}) \in \hat{F} \mid y_0] \\
&\leq \mathop{\mathbb{E}}_{y_0 \leftarrow_R [T(n+1)]} (e^{\varepsilon} \Pr[\hat{\mathcal{M}}(\hat{S}') \in \hat{F} \mid y_0] + \delta(n+1)) \\
&\qquad \text{(since } \hat{S} \text{ and } \hat{S}' \text{ are neighboring databases for fixed } y_0) \\
&= e^{\varepsilon} \Pr[\mathcal{M}(S') \in F] + \delta(n+1),
\end{aligned}$$

where $\hat{F}$ is the set of y that agree with y_0 in exactly the first x entries for some $x \in F$.

Now we argue that $\mathcal{M}$ solves the interior point problem on $\mathcal{D}_n$ with probability greater than P_n. First we show that $x \geq \min S$ with probability greater than P_{n+1}. Observe that by construction, all the elements of $\hat{S}$ agree in at least the first $\min S$ digits, and hence so does any interior point of $\hat{S}$. Therefore, if $\mathcal{M}'$ succeeds in outputting an interior point y of $\hat{S}$, then y must in particular agree with y_0 in at least $\min S$ digits, so $x \geq \min S$.

Now we use the privacy that $\hat{\mathcal{M}}$ provides to y_0 to show that $x \leq \max S$ except with probability at most $e^{\varepsilon}/b(n+1) + \delta(n+1)$. Fix a database S. Let $w = \max S$, and fix all the randomness of $\mathcal{M}$ but the $(w+1)$st entry of y_0 (note that since $w = \max S$, this fixes $y_1, \ldots, y_n$). Since the $(w+1)$st entry of y_0 is still a uniformly random element of $[b(n+1)]$, the privately produced entry y^{w+1} should not be able to do much better than randomly guessing $y_0^{(w+1)}$. Formally, for each $z \in [b(n+1)]$, let $\hat{S}_z$ denote the database $\hat{S}$ with $y_0^{(w+1)}$ set to z and everything else fixed as above. Then by the differential privacy of $\hat{\mathcal{M}}$,

$$\begin{aligned}
\Pr_{z \in [b(n+1)]}[\hat{\mathcal{M}}(\hat{S}_z)^{w+1} = z] &= \frac{1}{b(n+1)} \sum_{z \in [b(n+1)]} \Pr[\hat{\mathcal{M}}(\hat{S}_z)^{w+1} = z] \\
&\leq \frac{1}{b(n+1)} \sum_{z \in [b(n+1)]} \mathop{\mathbb{E}}_{z' \leftarrow_R [b(n+1)]} \left[e^{\varepsilon} \Pr[\hat{\mathcal{M}}(\hat{S}_{z'})^{w+1} = z] + \delta(n+1) \right] \\
&\leq \frac{e^{\varepsilon}}{b(n+1)} + \delta(n+1),
\end{aligned}$$

where all probabilities are also taken over the coins of $\hat{\mathcal{M}}$. Observe that $x \leq \max S$ whenever

$y^{w+1} \neq y_0^{(w+1)}$, and thus, we have that $x \leq \max S$ except with probability at most $e^{\varepsilon}/b(n+1) + \delta(n+1)$. By a union bound, $\min S \leq x \leq \max S$ with probability greater than

$$P_{n+1} - \left(\frac{e^{\varepsilon}}{b(n+1)} + \delta(n+1) \right) = P_n.$$

This gives the desired contradiction. □

We now finalize our lower bound on the sample complexity of private algorithms for solving the interior point problem by estimating the $T(n)$ guaranteed by Lemma 7.6.

Theorem 7.7. *Fix any constant $0 < \varepsilon < 1/4$. Let $\delta(n) \leq 1/(50n^2)$. Then for every positive integer n, solving the interior point problem on X with probability at least 3/4 and with $(\varepsilon, \delta(n))$-differential privacy requires sample complexity $n \geq \Omega(\log^* |X|)$.*

Lemma 7.6 states that every $(\varepsilon, \delta(n))$-differentially private mechanism with sample complexity n for solving the interior point problem on a domain of size $T(n)$ has success probability strictly less than 3/4. Alternatively, every $(\varepsilon, \delta(n))$-differentially private mechanism that solves the interior point problem on a domain of size $T(n)$ with success probability at least 3/4 must have sample complexity greater than n. Thus, to prove Theorem 7.7, we need to show that $n = \Omega\left(\log^*(T(n))\right)$.

Proof of Theorem 7.7. Let $T(n)$ be as in Lemma 7.6. We introduce the following notation for iterated exponentials:

$$\text{tower}^{(0)}(x) = x \quad \text{and} \quad \text{tower}^{(k)}(x) = 2^{\text{tower}^{(k-1)}(x)}.$$

Observe that for $k \geq 1$, $x \geq 1$, and $M > 16$,

$$\begin{aligned} M^{\text{tower}^{(k)}(x)} &= 2^{\text{tower}^{(k)}(x) \cdot \log M} \\ &= \text{tower}^{(2)}(\text{tower}^{(k-1)}(x) + \log\log M) \\ &\leq \text{tower}^{(2)}(\text{tower}^{(k-1)}(x + \log\log M)) \\ &= \text{tower}^{(k+1)}(x + \log\log M). \end{aligned}$$

By induction on n we get an upper bound of

$$T(n) \leq \text{tower}^{(n-1)}(2 + n\log\log(50n^2)) \leq \text{tower}^{(n+\log^*(2+n\log\log(50n^2)))}(1).$$

This immediately shows that solving the interior point problem on $X = [T(n)]$ requires sample complexity

$$\begin{aligned} n &\geq \log^* T(n) - \log^*(2 + n\log\log(50n^2)) \\ &\geq \log^* T(n) - O(\log^*\log^* T(n)) \\ &= \log^*|X| - O(\log^*\log^*|X|). \end{aligned}$$

To get a lower bound for solving the interior point problem on X when $|X|$ is not of the form $T(n)$, note that a mechanism for X is also a mechanism for every X' s.t. $|X'| \leq |X|$. The lower bound follows by setting $|X'| = T(n)$ for the largest n such that $T(n) \leq |X|$. □

7.3 The Interior Point Problem – Upper Bound

We now present a recursive algorithm, RecPrefix, for privately solving the interior point problem. Recall that in Chapter 6 we presented algorithm RecConcave for privately solving quasi-concave promise problems. While algorithm RecPrefix is more efficient, algorithm RecConcave is more general. Indeed, the interior point problem can easily be stated as a quasi-concave promise problems, but the reverse direction is not immediately clear.

Theorem 7.8. *Let $\beta, \varepsilon, \delta > 0$, let X be a finite, totally ordered domain, and let $n \in \mathbb{N}$ with $n \geq \frac{18500}{\varepsilon} \cdot 2^{\log^*|X|} \cdot \log^*(|X|) \cdot \ln(\frac{4\log^*|X|}{\beta\varepsilon\delta})$. If RecPrefix (defined below) is executed on a database $S \in X^n$ with parameters $\frac{\beta}{3\log^*|X|}, \frac{\varepsilon}{2\log^*|X|}, \frac{\delta}{2\log^*|X|}$, then*

1. *RecPrefix is (ε, δ)-differentially private;*
2. *With probability at least $(1-\beta)$, the output x satisfies $\min\{x_i : x_i \in S\} \leq x \leq \max\{x_i : x_i \in S\}$.*

The idea of the algorithm is that on each level of recursion, RecPrefix takes an input database S over X and constructs a database S' over a smaller universe X', where $|X'| =$

$\log|X|$, in which every element is the length of the longest prefix of a pair of elements in S (represented in binary). In a sense, this reverses the construction presented in Section 7.2. We remark that RecPrefix is computationally efficient, running in time $O(n \cdot \log|X|)$, which is linear in the bit-representation of the input database.

7.3.1 The Choosing Mechanism

Before formally presenting the algorithm RecPrefix, we describe an (ε, δ)-differentially private variant of the exponential mechanism called the *choosing mechanism*.[1]

A quality function $q : X^* \times F \to \mathbb{N}$ with sensitivity at most 1 is of *k-bounded-growth* if adding an element to a database can increase (by 1) the score of at most k solutions, without changing the scores of other solutions. Specifically, it holds that

1. $q(\emptyset, f) = 0$ for all $f \in F$,
2. If $S_2 = S_1 \cup \{x\}$, then $q(S_1, f) + 1 \geq q(S_2, f) \geq q(S_1, f)$ for all $f \in F$, and
3. If $S_2 = S_1 \cup \{x\}$, then there are at most k values of f for which $q(S_2, f) = q(S_1, f) + 1$.

The choosing mechanism is a differentially private algorithm for approximately solving bounded-growth choice problems: Given a database S and a k-bounded-growth quality function $q : X^* \times F \to \mathbb{N}$, the goal is to privately identify a solution $f \in F$ with high $q(S, f)$. We denote $\mathrm{OPT}(S) = \max_{f \in F}\{q(S, f)\}$. Step 1 of the algorithm checks whether a good solution exists (otherwise any solution is approximately optimal) and step 2 invokes the exponential mechanism, but with the *small* set $G(S)$ of good solutions instead of F.

Algorithm Choosing Mechanism

Input: database S, quality function q, solution set F, and parameters $\beta, \varepsilon, \delta$ and k.

1. Set $\widetilde{\mathrm{OPT}}(S) = \mathrm{OPT}(S) + \mathrm{Lap}(\frac{4}{\varepsilon})$. If $\widetilde{\mathrm{OPT}}(S) < \frac{8}{\varepsilon}\ln(\frac{4k}{\beta\varepsilon\delta})$ then halt and return $\perp$.
2. Let $G(S) = \{f \in F : q(S, f) \geq 1\}$. Choose and return $f \in G(S)$ using the exponential mechanism with parameter $\varepsilon/2$.

[1] The choosing mechanism was first introduced in the work of Beimel et al. [11], which is presented in Chapter 6. Due to space limitations, we omitted its description and applications from Chapter 6. See [11] for more details.

The following lemmas give the privacy and utility guarantees of the choosing mechanism.

Lemma 7.9. *Fix $\delta > 0$, and $0 < \varepsilon \leq 2$. If q is a k-bounded-growth quality function, then the choosing mechanism is (ε, δ)-differentially private.*

Lemma 7.10. *Assume that the choosing mechanism is executed on a k-bounded-growth quality function q, and on a database S containing n elements. With probability at least $(1-\beta)$, the choosing mechanism outputs a solution f with quality $q(S,f) \geq \mathrm{OPT}(S) - \frac{16}{\varepsilon}\ln(\frac{4kn}{\beta\varepsilon\delta})$.*

Sometimes, a solution $f \in F$ s.t. $q(S,f) > 0$ suffices. In such cases it is convenient to use to following variant of Lemma 7.10:

Lemma 7.11. *Assume that the choosing mechanism is executed on a k-bounded-growth quality function q, and on a database S s.t. there exists a solution $\hat{f}$ with quality $q(S,\hat{f}) \geq \frac{16}{\varepsilon}\ln(\frac{4k}{\beta\varepsilon\delta})$. With probability at least $(1-\beta)$, the choosing mechanism outputs a solution f with quality $q(S,f) \geq 1$.*

We supply the proofs of privacy and utility for the choosing mechanism.

Proof of Lemma 7.9. Let $\mathcal{A}$ denote the choosing mechanism (Algorithm 8). Let S, S' be neighboring databases of n elements. We need to show that $\Pr[\mathcal{A}(S) \in R] \leq \exp(\varepsilon) \cdot \Pr[\mathcal{A}(S') \in R] + \delta$ for every set of outputs $R \subseteq F \cup \{\perp\}$. Note first that $\mathrm{OPT}(S) = \max_{f \in F}\{q(S,f)\}$ has sensitivity at most 1, so by the properties of the Laplace mechanism,

$$\begin{aligned}
\Pr[\mathcal{A}(S) = \perp] &= \Pr\left[\widetilde{\mathrm{OPT}}(S) < \frac{8}{\varepsilon}\ln(\frac{4k}{\beta\varepsilon\delta})\right] \\
&\leq \exp(\frac{\varepsilon}{4}) \cdot \Pr\left[\widetilde{\mathrm{OPT}}(S') < \frac{8}{\varepsilon}\ln(\frac{4k}{\beta\varepsilon\delta})\right] \\
&= \exp(\frac{\varepsilon}{4}) \cdot \Pr[\mathcal{A}(S') = \perp]. \qquad (7.1)
\end{aligned}$$

Similarly, we have $\Pr[\mathcal{A}(S) \neq \perp] \leq \exp(\varepsilon/4)\Pr[\mathcal{A}(S') \neq \perp]$. Thus, we my assume below that $\perp \notin R$. (If $\perp \in R$, then we can write $\Pr[\mathcal{A}(S) \in R] = \Pr[\mathcal{A}(S) = \perp] + \Pr[\mathcal{A}(S) \in R \setminus \{\perp\}]$, and similarly for S'.)

Case (a): $\mathrm{OPT}(S) < \frac{4}{\varepsilon}\ln(\frac{4k}{\beta\varepsilon\delta})$**.** It holds that

$$\begin{aligned}\Pr[\mathcal{A}(S) \in R] &\leq \Pr[\mathcal{A}(S) \neq \perp] \\ &\leq \Pr\left[\mathrm{Lap}\left(\frac{4}{\varepsilon}\right) > \frac{4}{\varepsilon}\ln\left(\frac{4k}{\beta\varepsilon\delta}\right)\right] \\ &\leq \delta \leq \Pr[\mathcal{A}(S') \in R] + \delta.\end{aligned}$$

Case (b): $\mathrm{OPT}(S) \geq \frac{4}{\varepsilon}\ln(\frac{4k}{\beta\varepsilon\delta})$**.** Let $G(S)$ and $G(S')$ be the sets used in step 2 in the executions on S and on S', respectively. We will show that the following two facts hold:

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

Fact 2 : For every possible output $f \in G(S) \cap G(S')$, it holds that $\Pr[\mathcal{A}(S) = f] \leq e^{\varepsilon} \cdot \Pr[\mathcal{A}(S') = f]$.

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

$$\begin{aligned}\Pr[\mathcal{A}(S) \in R] &= \Pr[\mathcal{A}(S) \in R \setminus B] + \sum_{f \in R \cap B} \Pr[\mathcal{A}(S) = f] \\ &\leq e^{\varepsilon} \cdot \Pr[\mathcal{A}(S') \in R \setminus B] + |R \cap B|\frac{\delta}{k} \\ &\leq e^{\varepsilon} \cdot \Pr[\mathcal{A}(S') \in R] + \delta.\end{aligned}$$

To prove Fact 1, let $f \in G(S) \setminus G(S')$. That is, $q(S, f) \geq 1$ and $q(S', f) = 0$. As q has sensitivity at most 1, it must be that $q(S, f) = 1$. As there exists $\hat{f} \in S$ with $q(S, \hat{f}) \geq \frac{4}{\varepsilon}\ln(\frac{4k}{\beta\varepsilon\delta})$, we have that

$$\Pr[\mathcal{A}(S) = f] \leq \Pr\left[\begin{array}{c}\text{The exponential} \\ \text{mechanism chooses } f\end{array}\right] \leq \frac{\exp(\frac{\varepsilon}{4} \cdot 1)}{\exp(\frac{\varepsilon}{4} \cdot \frac{4}{\varepsilon}\ln(\frac{4k}{\beta\varepsilon\delta}))} = \exp\left(\frac{\varepsilon}{4}\right)\frac{\beta\varepsilon\delta}{4k},$$

which is at most δ/k for $\varepsilon \leq 2$.

To prove Fact 2, let $f \in G(S) \cap G(S')$ be a possible output of $\mathcal{A}(S)$. We use the following Fact 3, proved below.

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

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

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

We now prove Fact 3. Let $\mathcal{X} \triangleq \sum_{h \in G(S)} \exp(\frac{\varepsilon}{4} q(S, h))$. Since there exists a solution $\hat{f}$ s.t. $q(S, \hat{f}) \geq \frac{4}{\varepsilon} \ln(\frac{4k}{\beta \varepsilon \delta})$, we have $\mathcal{X} \geq \exp(\frac{\varepsilon}{4} \cdot \frac{4}{\varepsilon} \ln(\frac{4k}{\beta \varepsilon \delta})) \geq \frac{4k}{\varepsilon}$.

Now, recall that q is of k-bounded growth, so $|G(S') \setminus G(S)| \leq k$, and every $h \in (G(S') \setminus G(S))$ satisfies $q(S', h) = 1$. Hence,

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

where the last inequality follows from the fact that $\mathcal{X} \geq 4k/\varepsilon$. This concludes the proof of Fact 3, and completes the proof of the lemma. □

The utility analysis for the choosing mechanism is rather straightforward:

Proof of Lemma 7.11. Recall that the mechanism defines $\widetilde{\mathrm{OPT}}(S)$ as $\mathrm{OPT}(S) + \mathrm{Lap}(\frac{4}{\varepsilon})$. Note that the mechanism succeeds whenever $\widetilde{\mathrm{OPT}}(S) \geq \frac{8}{\varepsilon} \ln(\frac{4k}{\beta \varepsilon \delta})$. This happens provided the

$\mathrm{Lap}\left(\frac{4}{\varepsilon}\right)$ random variable is at most $\frac{8}{\varepsilon}\ln(\frac{4k}{\beta\varepsilon\delta})$, which happens with probability at least $(1-\beta)$. □

Proof of Lemma 7.10. Note that if $\mathrm{OPT}(S) < \frac{16}{\varepsilon}\ln(\frac{4kn}{\beta\varepsilon\delta})$, then every solution is a good output, and the mechanism cannot fail. Assume, therefore, that there exists a solution f s.t. $q(f,S) \geq \frac{16}{\varepsilon}\ln(\frac{4kn}{\beta\varepsilon\delta})$, and recall that the mechanism defines $\widetilde{\mathrm{OPT}}(S)$ as $\mathrm{OPT}(S) + \mathrm{Lap}(\frac{4}{\varepsilon})$. As in the proof of Lemma 7.11, with probability at least $1-\beta/2$, we have $\widetilde{\mathrm{OPT}}(S) \geq \frac{8}{\varepsilon}\ln\left(\frac{4k}{\beta\varepsilon\delta}\right)$. Assuming this event occurs, we will show that with probability at least $1-\beta/2$, the exponential mechanism chooses a solution f s.t. $q(S,f) \geq \mathrm{OPT}(S) - \frac{16}{\varepsilon}\ln(\frac{4kn}{\beta\varepsilon\delta})$.

By the growth-boundedness of q, and as S is of size n, there are at most kn possible solutions f with $q(S,f) > 0$. That is, $|G(S)| \leq kn$. By the properties of the exponential mechanism, we obtain a solution as desired with probability at least

$$\left(1 - kn \cdot \exp\left(-\frac{\varepsilon}{4} \cdot \frac{16}{\varepsilon}\ln\left(\frac{4kn}{\beta\varepsilon\delta}\right)\right)\right) \geq \left(1 - \frac{\beta}{2}\right).$$

By a union bound, we get that the choosing mechanism outputs a good solution with probability at least $(1-\beta)$. □

7.3.2 The RecPrefix Algorithm

We are now ready to present and analyze the algorithm RecPrefix.

We start the analysis of RecPrefix with the following two simple observations.

Observation 7.12. *There are at most* $\log^*|X|$ *recursive calls throughout the execution of* RecPrefix *on a database* $S \in X^*$.

Observation 7.13. *Let* RecPrefix *be executed on a database* $S \in X^n$, *where* $n \geq 2^{\log^*|X|} \cdot \frac{2312}{\varepsilon} \cdot \ln(\frac{4}{\beta\varepsilon\delta})$. *Every recursive call throughout the execution operates on a database containing at least* $\frac{1540}{\varepsilon} \cdot \ln(\frac{4}{\beta\varepsilon\delta})$ *elements.*

Proof. This follows from Observation 7.12 and from the fact that the i^{th} recursive call is executed on a database of size $n_i = \frac{n}{2^{i-1}} - k\sum_{\ell=0}^{i-2}(\frac{1}{2})^\ell \geq \frac{n}{2^i} - 2k$. □

We now analyze the utility guarantees of RecPrefix by proving the following lemma.

Algorithm RecPrefix

Input: Database $S = (x_j)_{j=1}^n \in X^n$, parameters $\beta, \varepsilon, \delta$.

1. If $|X| \leq 32$, then use the exponential mechanism with privacy parameter ε and quality function $q(S,x) = \min\left\{\#\{j : x_j \geq x\}, \#\{j : x_j \leq x\}\right\}$ to choose and return a point $x \in X$.
2. Let $k = \lfloor \frac{386}{\varepsilon} \ln(\frac{4}{\beta\varepsilon\delta}) \rfloor$, and let $Y = (y_1, y_2, \ldots, y_{n-2k})$ be a random permutation of the smallest $(n-2k)$ elements in S.
3. For $j = 1$ to $\frac{n-2k}{2}$, define z_j as the length of the longest prefix for which y_{2j-1} agrees with y_{2j} (in base 2 notation).
4. Execute RecPrefix recursively on $S' = (z_j)_{j=1}^{(n-2k)/2} \in (X')^{(n-2k)/2}$ with parameters $\beta, \varepsilon, \delta$. Recall that $|X'| = \log|X|$. Denote the returned value by z.
5. Use the choosing mechanism to choose a prefix L of length $(z+1)$ with a large number of agreements among elements in S. Use parameters $\beta, \varepsilon, \delta$, and the quality function $q : X^* \times \{0,1\}^{z+1} \to \mathbb{N}$, where $q(S, I)$ is the number of agreements on I among $x_1, \ldots, x_n$.
6. For $\sigma \in \{0,1\}$, define $L_\sigma \in X$ to be the prefix L followed by $(\log|X| - z - 1)$ appearances of σ.
7. Compute $\widehat{big} = \mathrm{Lap}(\frac{1}{\varepsilon}) + \#\{j : x_j \geq L_1\}$.
8. If $\widehat{big} \geq \frac{3k}{2}$ then return L_1. Otherwise return L_0.

Lemma 7.14. *Let $\beta, \varepsilon, \delta$, and $S \in X^n$ be inputs on which RecPrefix performs at most N recursive calls, all of which are on databases of at least $\frac{1540}{\varepsilon} \cdot \ln(\frac{4}{\beta\varepsilon\delta})$ elements. With probability at least $(1 - 3\beta N)$, the output x is s.t.*

1. *$\exists x_i \in S$ s.t. $x_i \leq x$;*
2. *$|\{i : x_i \geq x\}| \geq k \triangleq \lfloor \frac{386}{\varepsilon} \cdot \ln(\frac{4}{\beta\varepsilon\delta}) \rfloor$.*

Before proving the lemma, we make a combinatorial observation that motivates the random shuffling in step 2 of RecPrefix. A pair of elements $y, y' \in S$ is useful in Algorithm RecPrefix if many of the values in S lie between y and y' – a prefix on which y, y' agree is also a prefix of every element between y and y'. A prefix common to a useful pair can hence be identified privately via stability-based techniques. Towards creating useful pairs, the set S is shuffled randomly. We will use the following lemma:

Claim 7.15. *Let* $(\Pi_1, \Pi_2, \ldots, \Pi_n)$ *be a random permutation of* $(1, 2, \ldots, n)$*. Then for all* $r \geq 1$*,*

$$\Pr\left[\left|\left\{i : |\Pi_{2i-1} - \Pi_{2i}| \leq \frac{r}{12}\right\}\right| \geq r\right] \leq 2^{-r}$$

Proof. We need to show that w.h.p. there are at most r "bad" pairs (Π_{2i-1}, Π_{2i}) within distance $\frac{r}{12}$. For each i, we call Π_{2i-1} the left side of the pair, and Π_{2i} the right side of the pair. Let us first choose r elements to be placed on the left side of r bad pairs (there are $\binom{n}{r}$ such choices). Once those are fixed, there are at most $(\frac{r}{6})^r$ choices for placing elements on the right side of those pairs. Now we have r pairs and $n-2r$ unpaired elements that can be shuffled in $(n-r)!$ ways. Overall, the probability of having at least r bad pairs is at most

$$\frac{\binom{n}{r}(\frac{r}{6})^r(n-r)!}{n!} = \frac{(\frac{r}{6})^r}{r!} \leq \frac{(\frac{r}{6})^r}{\sqrt{r}r^r e^{-r}} = \frac{e^r}{\sqrt{r}6^r} \leq 2^{-r},$$

where we have used Stirling's approximation for the first inequality. □

Suppose we have paired random elements in our input database S, and constructed a database S' containing lengths of the prefixes for those pairs. Moreover, assume that by recursion we have identified a length z which is the length of at least r random pairs. Although those prefixes may be different for each pair, Claim 7.15 guarantees that (w.h.p.) at least one of these prefixes is the prefix of at least $\frac{r}{12}$ input elements. This will help us in (privately) identifying such a prefix.

Proof of Lemma 7.14. The proof is by induction on the number of recursive calls, denoted as t. For $t = 1$ (i.e., $|X| \leq 32$), the claim holds as long as the exponential mechanism outputs an x with $q(S, x) \geq k$ except with probability at most β. By Proposition 3.27, it suffices to have $n \geq \frac{1540}{\varepsilon} \cdot \ln(\frac{4}{\beta\varepsilon\delta})$, since $32\exp(-\varepsilon(n/2-k)/2) \leq \beta$.

Assume that the stated lemma holds whenever RecPrefix performs at most $t-1$ recursive calls. Let $\beta, \varepsilon, \delta$ and $S = (x_i)_{i=1}^n \in X^n$ be inputs on which algorithm RecPrefix performs t recursive calls, all of which are on databases containing at least $\frac{1540}{\varepsilon} \cdot \ln(\frac{4}{\beta\varepsilon\delta})$ elements. Consider the first call in the execution on those inputs, and let $y_1, \ldots, y_{n-2k}$ be the random

permutation chosen in step 2. We say that a pair y_{2j-1}, y_{2j} is *close* if

$$\left|\left\{i : \begin{array}{c} y_{2j-1} \le y_i \le y_{2j} \\ \text{or} \\ y_{2j} \le y_i \le y_{2j-1} \end{array}\right\}\right| \le \frac{k-1}{12}.$$

By Claim 7.15, except with probability at most $2^{-(k-1)} < \beta$, there are at most $(k-1)$ close pairs. We continue the proof assuming that this is the case.

Let $S' = (z_i)_{i=1}^{(n-2k)/2}$ be the database constructed in step 3. By the inductive assumption, with probability at least $(1 - 3\beta(t-1))$, the value z obtained in step 4 is s.t. (1) $\exists z_i \in S'$ s.t. $z_i \le z$; and (2) $|\{z_i \in S' : z_i \ge z\}| \ge k$. We proceed with the analysis assuming that this event happened.

By (2), there are at least k pairs y_{2j-1}, y_{2j} that agree on a prefix of length at least z. At least one of those pairs, say y_{2j^*-1}, y_{2j^*}, is not *close*. Note that every y between y_{2j^*-1} and y_{2j^*} agrees on the same prefix of length z, and that there are at least $\frac{k-1}{12}$ such elements in S. Moreover, as the next bit is either 0 or 1, at least half of those elements agree on a prefix of length $(z+1)$. Thus, when using the choosing mechanism in step 5 (to choose a prefix of length $(z+1)$), there exists at least one prefix with quality at least $\frac{k-1}{24} \ge \frac{16}{\varepsilon} \cdot \ln(\frac{4}{\beta\varepsilon\delta})$. By Lemma 7.10, the choosing mechanism ensures, therefore, that with probability at least $(1-\beta)$, the chosen prefix L is the prefix of at least one $y_{i'} \in S$, and, hence, this $y_{i'}$ satisfies $L_0 \le y_{i'} \le L_1$ (defined in step 6). We proceed with the analysis assuming that this is the case.

Let $z_{\hat{j}} \in S'$ be s.t. $z_{\hat{j}} \le z$. By the definition of $z_{\hat{j}}$, this means that $y_{2\hat{j}-1}$ and $y_{2\hat{j}}$ agree on a prefix of length at most z. Hence, as L is of length $z+1$, we have that either $\min\{y_{2\hat{j}-1}, y_{2\hat{j}}\} < L_0$ or $\max\{y_{2\hat{j}-1}, y_{2\hat{j}}\} > L_1$. If $\min\{y_{2\hat{j}-1}, y_{2\hat{j}}\} < L_0$, then L_0 satisfies Condition 1 of being a good output. It also satisfies Condition 2 because $y_{i'} \ge L_0$ and $y_{i'} \in Y$, which we took to be the smallest $n-2k$ elements of S. Similarly, L_1 is a good output if $\max\{y_{2\hat{j}-1}, y_{2\hat{j}}\} > L_1$. In any case, at least one out of L_0, L_1 is a good output.

If both L_0 and L_1 are good outputs, then step 8 cannot fail. We have already established the existence of $L_0 \le y_{i'} \le L_1$. Hence, if L_1 is not a good output, then there are at most $(k-1)$ elements $x_i \in S$ s.t. $x_i \ge L_1$. Hence, the probability of $\widehat{big} \ge 3k/2$ and step 8 failing is at most $\exp(-\frac{\varepsilon k}{2}) \le \beta$. It remains to analyze the case where L_0 is not a good output (and L_1

is).

If L_0 is not a good output, then every $x_j \in S$ satisfies $x_j > L_0$. In particular, $\min\{y_{2\hat{j}-1}, y_{2\hat{j}}\} > L_0$, and, hence, $\max\{y_{2\hat{j}-1}, y_{2\hat{j}}\} > L_1$. Recall that there are at least $2k$ elements in S that are bigger than $\max\{y_{2\hat{j}-1}, y_{2\hat{j}}\}$. As $k \geq \frac{2}{\varepsilon}\ln(\frac{1}{\beta})$, the probability that $\widehat{big} < 3k/2$ and RecPrefix fails to return L_1 in this case is at most β.

All in all, RecPrefix fails to return an appropriate x with probability at most $3\beta t$. □

We now proceed with the privacy analysis.

Lemma 7.16. *When executed for N recursive calls, RecPrefix is $(2\varepsilon N, 2\delta N)$-differentially private.*

Proof. The proof is by induction on the number of recursive calls, denoted by t. If $t = 1$ (i.e., $|X| \leq 32$), then by Proposition 3.27 the exponential mechanism ensures that RecPrefix is $(\varepsilon, 0)$-differentially private. Assume that the stated lemma holds whenever RecPrefix performs at most $t-1$ recursive calls, and let $S_1, S_2 \in X^*$ be two neighboring databases on which RecPrefix performs t recursive calls.[2] Let $\mathcal{B}$ denote an algorithm consisting of steps 1-4 of RecPrefix (the output of $\mathcal{B}$ is the value z from step 4). Consider the executions of $\mathcal{B}$ on S_1 and on S_2, and denote by Y_1, S_1' and by Y_2, S_2' the elements Y, S' as they are in the executions on S_1 and on S_2.

We show that the distributions on the databases S_1' and S_2' are similar in the sense that for each database in one of the distributions there exists a neighboring database in the other that has the same probability. Thus, applying the recursion (which is differentially private by the inductive assumption) preserves privacy. We now make this argument formal.

First note that as S_1, S_2 differ in only one element, there is a bijection between orderings Π and $\widehat{\Pi}$ of the smallest $(n-2k)$ elements of S_1 and of S_2, respectively, s.t. Y_1 and Y_2 are neighboring databases. This is because there exists a permutation of the smallest $(n-2k)$ elements of S_1 that is a neighbor of the smallest $(n-2k)$ elements of S_2; composition with this fixed permutation yields the desired bijection. Moreover, note that whenever Y_1, Y_2 are neighboring databases, the same is true for S_1' and S_2'. Hence, for every set of outputs F it holds that

[2] The recursion depth is determined by $|X|$, which is identical in S_1 and in S_2.

$$\begin{aligned}
\Pr[\mathcal{B}(S) \in F] &= \sum_{\Pi} \Pr[\Pi] \cdot \Pr[\mathsf{RecPrefix}(S_1') \in F | \Pi] \\
&\leq e^{2\varepsilon(t-1)} \cdot \sum_{\Pi} \Pr[\Pi] \cdot \Pr[\mathsf{RecPrefix}(S_2') \in F | \widehat{\Pi}] + 2\delta(t-1) \\
&= e^{2\varepsilon(t-1)} \cdot \sum_{\widehat{\Pi}} \Pr[\widehat{\Pi}] \cdot \Pr[\mathsf{RecPrefix}(S_2') \in F | \widehat{\Pi}] + 2\delta(t-1) \\
&= e^{2\varepsilon(t-1)} \cdot \Pr[\mathcal{B}(S') \in F] + 2\delta(t-1)
\end{aligned}$$

So when executed for t recursive calls, the sequence of steps 1-4 of RecPrefix is $(2\varepsilon(t-1), 2\delta(t-1))$-differentially private. In steps 5 and 7, algorithm RecPrefix interacts with its database through the choosing mechanism and using the Laplace mechanism, each of which is (ε, δ)-differentially private. By composition (Theorem 3.5), we get that RecPrefix is $(2t\varepsilon, 2t\delta)$-differentially private. □

Combining Lemma 7.14 and Lemma 7.16 we obtain Theorem 7.8.

Informal Discussion and Open Questions

An natural open problem is to close the gap between our (roughly) $2^{\log^* |X|}$ upper bound on the sample complexity of privately solving the interior point problem (Theorem 7.8), and our $\log^* |X|$ lower bound (Theorem 7.7). Below we describe an idea for reducing the upper bound to $\text{poly}(\log^* |X|)$.

In our recursive construction for the lower bound, we took n elements $(x_1, \ldots, x_n)$ and generated $n+1$ elements where y_0 is a random element (independent of the x_i's), and every x_i is the length of the longest common prefix of y_0 and y_i. Therefore, a change limited to one x_i affects only one y_i and privacy is preserved (assuming that our future manipulations on $(y_0, \ldots, y_n)$ preserve privacy). While the representation length of domain elements grows exponentially on every step, the database size grows by 1. This resulted in the $\Omega(\log^* |X|)$ lower bound.

In RecPrefix on the other hand, every level of recursion shrank the database size by a factor of $\frac{1}{2}$, and hence, we required a sample of (roughly) $2^{\log^* |X|}$ elements. Specifically,

in each level of recursion, two input elements y_{2j-1}, y_{2j} were paired and a new element z_j was defined as the length of their longest common prefix. As with the lower bound, we wanted to ensure that a change limited to one of the inputs affects only one new element, and hence, every input element is paired only once, and the database size shrinks.

If we could pair input elements *twice* then the database size would only be reduced additively (which will hopefully result in a poly($\log^* |X|$) upper bound). However, this must be done carefully, as we are at risk of deteriorating the privacy parameter ε by a factor of 2 and thus remaining with an exponential dependency in $\log^* |X|$. Consider the following thought experiment for pairing elements.

Input: $(x_1, \ldots, x_n) \in X^n$.

1. Let $(y_1^0, \ldots, y_n^0)$ denote a random permutation of $(x_1, \ldots, x_n)$.
2. For $t = 1$ to $\log^* |X|$:

   For $i = 1$ to $(n-t)$, let y_i^t be the length of the longest common prefix of y_i^{t-1} and y_{i+1}^{t-1}.

As (most of the) elements are paired twice on every step, the database size reduces additively. In addition, every input element x_i affects at most $t+1$ elements at depth t, and the privacy loss is acceptable. However, this still does not solve the problem. Recall that every iteration of RecPrefix begins by randomly shuffling the inputs. Specifically, we needed to ensure that (w.h.p.) the number of "close" pairs is limited. The reason was that if a "not close" pair agrees on a prefix L, then L is the prefix of "a lot" of other elements as well, and we could privately identify L. In the above process we randomly shuffled only the elements at depth 0. Thus we do not know if the number of "close" pairs is small at depth $t > 0$. On the other hand, if we changed the pairing procedure to shuffle at every step, then each input element x_i might affect 2^t elements at depth t, causing the privacy loss to deteriorate rapidly.

7.4 Private PAC Learning vs. Private Empirical Learning

Recall that a PAC learner for a concept class C is required, given a labeled sample, to output a hypothesis with small error w.r.t. the target function and, importantly, w.r.t. the

underlying distribution. We now define the notion of an *empirical learner* which is similar to a PAC learner where accuracy is measured w.r.t. the fixed input database.

Definition 7.17 (Empirical Learner)**.** *Algorithm A is an (α, β)*-accurate empirical learner *for a concept class C over X using hypothesis class H with sample complexity m if for every $c \in C$ and for every database $S = ((x_i, c(x_i)), \ldots, (x_m, c(x_m))) \in (X \times \{0,1\})^m$ algorithm A outputs a hypothesis $h \in H$ satisfying $\Pr[\text{error}_S(c,h) \leq \alpha] \geq 1 - \beta$. The probability is taken over the coin tosses of A.*

Note that without privacy (and ignoring computational efficiency) identifying a hypothesis with small empirical error is trivial for every concept class C and for every database of size at least 1. This is not the case with (ε, δ)-differential privacy, and it can be shown that the sample complexity of every empirical learner for a concept class C is at least $\Omega(\frac{1}{\alpha\varepsilon}\text{VC}(C))$ (see [17] for a similar analysis).

In the next section we translate our bounds for the interior point problem to bounds on the sample complexity of empirically learning threshold functions (properly). In order to obtain bounds on the sample complexity of properly PAC learning thresholds, we now show that empirical learning and PAC learning are equivalent under differential privacy. While this equivalence is not limited to threshold functions, we state it specifically for thresholds in order to obtain better bounds.

One direction follows immediately from a standard generalization bound for learning thresholds (Lemma 3.19):

Lemma 7.18. *Any algorithm $\mathcal{A}$ for empirically learning $\texttt{THRESH}_X$ with (α, β)-accuracy is also a $(2\alpha, \beta + \beta')$-accurate PAC learner for $\texttt{THRESH}_X$ when given at least $\max\{n, 4\ln(2/\beta')/\alpha\}$ samples.*

For the other direction, we note that a distribution-free learner must perform well on the uniform distribution on the rows of any fixed database, and thus must be useful for outputting a consistent hypothesis on such a database. Thus if we have a PAC learner $\mathcal{A}$, the mechanism $\tilde{\mathcal{A}}$ that samples m rows from its input database $D \in (X \times \{0,1\})^n$ and runs $\mathcal{A}$ on the result should output a consistent hypothesis for D. The random sampling has two competing effects on privacy. On one hand, the possibility that an individual is sampled multiple times incurs additional privacy loss. On the other hand, if $n > m$, then a "secrecy-of-the-sample" argument shows that random sampling actually improves privacy,

since any individual is unlikely to have their data affect the computation at all. Refining an argument of Kasiviswanathan et al. [67], we show that if n is only a constant factor larger than m, these two effects offset, and the resulting mechanism is still differentially private.

Lemma 7.19. *Fix $\varepsilon \le 1$ and let $\mathcal{A}$ be an (ε, δ)-differentially private algorithm operating on databases of size m. For $n \ge 2m$, construct an algorithm $\tilde{\mathcal{A}}$ that on input a database D of size n, subsamples (with replacement) m rows from D, and runs $\mathcal{A}$ on the result. Then $\tilde{\mathcal{A}}$ is $(\tilde{\varepsilon}, \tilde{\delta})$-differentially private for*

$$\tilde{\varepsilon} = 6\varepsilon m/n \quad \text{and} \quad \tilde{\delta} = \exp(6\varepsilon m/n)\frac{4m}{n} \cdot \delta.$$

Proof. Let D, D' be adjacent databases of size n, and suppose without loss of generality that they differ on the last row. Let T be a random variable denoting the sequence of indices sampled by $\tilde{\mathcal{A}}$, and let $\ell(T)$ be the multiplicity of index n in T. Fix a subset S of the range of $\tilde{\mathcal{A}}$. For each $k = 0, 1, \ldots, m$ let

$$\begin{aligned}
p_k &= \Pr[\ell(T) = k] = \binom{m}{k} n^{-k}(1 - 1/n)^{m-k} = \binom{m}{k}(n-1)^{-k}(1-1/n)^m, \\
q_k &= \Pr[\mathcal{A}(D|_T) \in S | \ell(T) = k], \\
q'_k &= \Pr[\mathcal{A}(D'|_T) \in S | \ell(T) = k].
\end{aligned}$$

Here, $D|_T$ denotes the subsample of D consisting of the indices in T, and similarly for $D'|_T$. Note that $q_0 = q'_0$, since $D|_T = D'|_T$ if index n is not sampled. Our goal is to show that

$$\Pr[\tilde{\mathcal{A}}(D) \in S] = \sum_{k=0}^{m} p_k q_k \le e^{\tilde{\varepsilon}} \sum_{k=0}^{m} p_k q'_k + \tilde{\delta} = e^{\tilde{\varepsilon}} \Pr[\tilde{\mathcal{A}}(D') \in S] + \tilde{\delta}.$$

To do this, we first show that

$$q_k \le e^{\varepsilon} q_{k-1} + \delta.$$

To establish this inequality, we define a coupling of the conditional random variables $U = (T | \ell(T) = k)$ and $U' = (T | \ell(T) = k-1)$ with the property that U and U' are always at Hamming distance 1 (and hence $D|_U$ and $D|_{U'}$ are neighbors). Specifically, consider the

joint distribution (U, U') over $[n]^m \times [n]^m$ sampled as follows. Let $I' \subseteq [m]$ be a random set of indices with $|I'| = k-1$. Let i be a random index in $[m] - I'$, and let $I = I' \cup \{i\}$. Define U and U' by setting U_j for $j \in I$, setting $U'_j = n$ for $j \in I'$, setting $U_j = U'_j$ to be a random element of $[m-1]$ for each $j \notin I'$, and setting U_i to be random element of $[m-1]$. One can verify that the marginal distributions of U and U' are indeed uniform over the sequences in $[n]^m$ with $\ell(U) = k$ and $\ell(U') = k-1$, and moreover that U and U' are always at Hamming distance 1 apart, differing only at index i. Thus,

$$\begin{aligned}
q_k &= \Pr[A(D|_T) \in S | \ell(T) = k] \\
&= \mathop{\mathbb{E}}_{(U,U')} [\Pr[A(D|_U) \in S]] \\
&\leq \mathop{\mathbb{E}}_{(U,U')} [e^\varepsilon \Pr[A(D|_{U'}) \in S] + \delta] \\
&= e^\varepsilon \Pr[A(D|_T) \in S | \ell(T) = k-1] + \delta \\
&= e^\varepsilon q_{k-1} + \delta.
\end{aligned}$$

Hence,

$$q_k \leq e^{k\varepsilon} q_0 + \frac{e^{k\varepsilon} - 1}{e^\varepsilon - 1} \delta.$$

Hence,

$$\begin{aligned}
\Pr[\tilde{\mathcal{A}}(D) \in S] &= \sum_{k=0}^{m} p_k q_k \\
&\leq \sum_{k=0}^{m} \binom{m}{k} (n-1)^{-k} (1-1/n)^m \left(e^{k\varepsilon} q_0 + \frac{e^{k\varepsilon} - 1}{e^\varepsilon - 1} \delta \right) \\
&= q_0 (1-1/n)^m \sum_{k=0}^{m} \binom{m}{k} \left(\frac{e^\varepsilon}{n-1} \right)^k + \frac{\delta}{e^\varepsilon - 1} (1-1/n)^m \sum_{k=0}^{m} \binom{m}{k} \left(\frac{e^\varepsilon}{n-1} \right)^k - \frac{\delta}{e^\varepsilon - 1} \\
&= q_0 (1-1/n)^m \left(1 + \frac{e^\varepsilon}{n-1} \right)^m + \frac{\delta}{e^\varepsilon - 1} (1-1/n)^m \left(1 + \frac{e^\varepsilon}{n-1} \right)^m - \frac{\delta}{e^\varepsilon - 1} \\
&= q_0 \left(1 - \frac{1}{n} + \frac{e^\varepsilon}{n} \right)^m + \frac{\left(1 - \frac{1}{n} + \frac{e^\varepsilon}{n} \right)^m - 1}{e^\varepsilon - 1} \delta. \qquad (7.2)
\end{aligned}$$

Similarly, we also have that

$$\Pr[\tilde{\mathcal{A}}(D') \in S] \geq q_0 \left(1 - \frac{1}{n} + \frac{e^{-\varepsilon}}{n}\right)^m - \frac{\left(1 - \frac{1}{n} + \frac{e^{-\varepsilon}}{n}\right)^m - 1}{e^{-\varepsilon} - 1}\delta. \tag{7.3}$$

Combining inequalities 7.2 and 7.3 we get that

$$\Pr[\tilde{\mathcal{A}}(D) \in S] \leq \left(\frac{1 - \frac{1}{n} + \frac{e^{\varepsilon}}{n}}{1 - \frac{1}{n} + \frac{e^{-\varepsilon}}{n}}\right)^m \cdot \left\{\Pr[\tilde{\mathcal{A}}(D') \in S] + \frac{1 - \left(1 - \frac{1}{n} + \frac{e^{-\varepsilon}}{n}\right)^m}{1 - e^{-\varepsilon}}\delta\right\} + \frac{\left(1 - \frac{1}{n} + \frac{e^{\varepsilon}}{n}\right)^m - 1}{e^{\varepsilon} - 1}\delta,$$

proving that $\mathcal{A}'$ is $(\tilde{\varepsilon}, \tilde{\delta})$-differentially private for

$$\tilde{\varepsilon} \leq m \cdot \ln\left(\frac{1 + \frac{e^{\varepsilon}-1}{n}}{1 + \frac{e^{-\varepsilon}-1}{n}}\right) \leq \frac{6\varepsilon m}{n}$$

and

$$\begin{aligned}
\tilde{\delta} &\leq \exp(6\varepsilon m/n)\frac{1 - \left(1 + \frac{e^{-\varepsilon}-1}{n}\right)^m}{1 - e^{-\varepsilon}} \cdot \delta + \frac{\left(1 + \frac{e^{\varepsilon}-1}{n}\right)^m - 1}{e^{\varepsilon} - 1} \cdot \delta \\
&\leq \exp(6\varepsilon m/n)\frac{1 - \exp\left(2\frac{e^{-\varepsilon}-1}{n/m}\right)}{1 - e^{-\varepsilon}} \cdot \delta + \frac{\exp\left(\frac{e^{\varepsilon}-1}{n/m}\right) - 1}{e^{\varepsilon} - 1} \cdot \delta \\
&\leq \exp(6\varepsilon m/n)\frac{2m}{n} \cdot \delta + \frac{2m}{n} \cdot \delta \\
&\leq \exp(6\varepsilon m/n)\frac{4m}{n} \cdot \delta.
\end{aligned}$$

□

We now use Lemma 7.19 to complete the equivalence:

Lemma 7.20. *Suppose $\mathcal{A}$ is an (ε, δ)-differentially private (α, β)-accurate PAC learner for a concept class $\mathcal{C}$ with sample complexity m. Then there is an (ε, δ)-differentially private (α, β)-accurate empirical learner for $\mathcal{C}$ with sample complexity $n = 9m$. Moreover, if $\mathcal{A}$ is proper, then so is the resulting empirical learner.*

Proof. Consider a database $D = \{(x_i, y_i)\} \in (X \times \{0,1\})^n$. Let $\mathcal{D}$ denote the uniform distribution over the rows of D. Then drawing m i.i.d. samples from $\mathcal{D}$ is equivalent to subsampling

m rows of D (with replacement). Consider the algorithm $\tilde{A}$ that subsamples (with replacement) m rows from D and runs $\mathcal{A}$ on it. Then with probability at least $1-\beta$, algorithm $\mathcal{A}$ outputs an α-good hypothesis on $\mathcal{D}$, which is in turn an α-consistent hypothesis for D. Moreover, by Lemma 7.19 (secrecy-of-the-sample), algorithm $\mathcal{A}$ is (ε,δ)-differentially private. □

7.5 Private Learning of Thresholds vs. the Interior Point Problem

We show that with differential privacy, there is a $\Theta(1/\alpha)$ multiplicative relationship between the sample complexities of properly PAC learning thresholds with (α,β)-accuracy and of solving the interior point problem with error probability $\Theta(\beta)$. Specifically, we show

Theorem 7.21. *Let X be a totally ordered domain. Then,*

1. *If there exists an (ε,δ)-differentially private algorithm solving the interior point problem on X with error probability β and sample complexity n, then there is a $(2\varepsilon,(1+e^{\varepsilon})\delta)$-differentially private $(2\alpha,2\beta)$-accurate proper PAC learner for THRESH_X with sample complexity $\max\left\{\frac{n}{2\alpha},\frac{4\log(2/\beta)}{\alpha}\right\}$.*
2. *If there exists an (ε,δ)-differentially private (α,β)-accurate proper PAC learner for THRESH_X with sample complexity n, then there is a $(2\varepsilon,(1+e^{\varepsilon})\delta)$-differentially private algorithm that solves the interior point problem on X with error β and sample complexity $27\alpha n$.*

By the equivalence between empirically learning thresholds and PAC learning thresholds (see the previous section), it suffices to show a $\Theta(1/\alpha)$ relationship between the sample complexity of solving the interior point problem and the sample complexity of empirically learning thresholds.

Lemma 7.22. *Let X be a totally ordered domain. Then,*

1. *If there exists an (ε,δ)-differentially private algorithm solving the interior point problem on X with error probability β and sample complexity n, then there is a $(2\varepsilon,(1+e^{\varepsilon})\delta)$-*

differentially private algorithm for properly and empirically learning thresholds with (α, β)-accuracy and sample complexity $n/(2\alpha)$.

2. *If there exists an (ε, δ)-differentially private algorithm that is able to properly and empirically learn thresholds on X with (α, β)-accuracy and sample complexity $n/(3\alpha)$, then there is a $(2\varepsilon, (1+e^{\varepsilon})\delta)$-differentially private algorithm that solves the interior point problem on X with error β and sample complexity n.*

Proof. For the first direction, let $\mathcal{A}$ be a private algorithm for the interior point problem on databases of size n. Consider the algorithm $\mathcal{A}'$ that, on input a database D of size $n/(2\alpha)$, runs $\mathcal{A}'$ on a database D' consisting of the largest $n/2$ elements of D that are labeled 1 and the smallest $n/2$ elements of D that are labeled 0. If there are not enough of either such element, pad D' with $\min\{X\}$'s or $\max\{X\}$'s, respectively. Note that if x is an interior point of D' then c_x is a threshold function with error at most $\frac{n/2}{n/(2\alpha)}$ on D, and is hence α-consistent with D. For privacy, note that changing one row of D changes at most two rows of D'. Hence, applying algorithm $\mathcal{A}$ preserves $(2\varepsilon, (e^{\varepsilon}+1)\delta)$-differential privacy.

For the reverse direction, suppose $\mathcal{A}'$ privately finds an α-consistent threshold functions for databases of size $n/(3\alpha)$. Define $\mathcal{A}$ on a database $D' \in X^n$ to label the smaller $n/2$ points 1 and the larger $n/2$ points 0 to obtain a labeled database $D \in (X \times \{0,1\})^n$, pad D with an equal number of $(\min\{X\}, 1)$ and $(\max\{X\}, 0)$ entries to make it of size $n/(3\alpha)$, and run $\mathcal{A}'$ on the result. Note that if c_x is a threshold function with error at most α on D then x is an interior point of D', as otherwise c_x has error at least $\frac{n/2}{n/(3\alpha)} > \alpha$ on D. For privacy, note that changing one row of D' changes at most two rows of D. Hence, applying algorithm $\mathcal{A}'$ preserves $(2\varepsilon, (e^{\varepsilon}+1)\delta)$-differential privacy. □

7.6 Thresholds in High Dimension

We next show that the bound of $\Omega(\log^*|X|)$ on the sample complexity of private proper-learners for $\mathtt{THRESH}_X$ extends to conjunctions of ℓ independent threshold functions in ℓ dimensions. As we will see, every private proper-learner for this class requires a sample of $\Omega(\ell \cdot \log^*|X|)$ elements.

The significance of this lower bound is twofold. First, for reasonable settings of parameters (e.g., δ is negligible and items in X are of polynomial bit length in n), our $\Omega(\log^*|X|)$

lower bound for threshold functions is dominated by the dependence on $\log(1/\delta)$ in the upper bound. However, $\ell \cdot \log^*|X|$ can still be much larger than $\log(1/\delta)$, even when δ is negligible in the bit length of items in X^ℓ. Second, the lower bound for threshold functions only yields a separation between the sample complexities of private and non-private learning for a class of VC dimension 1. Since the concept class of ℓ-dimensional thresholds has VC dimension of ℓ, we obtain an $\omega(\mathrm{VC}(C))$ lower bound for concept classes even with arbitrarily large VC dimension.

Consider the following extension of THRESH_X to ℓ dimensions.

Definition 7.23. *For a totally ordered set X and $\vec{a} = (a_1, \ldots, a_\ell) \in X^\ell$ define the concept $c_{\vec{a}} : X^\ell \to \{0,1\}$ where $c_{\vec{a}}(\vec{x}) = 1$ if and only if for every $1 \leq i \leq \ell$ it holds that $x_i \leq a_i$. Define the concept class of all thresholds over X^ℓ as $\mathsf{THRESH}_X^\ell = \{c_{\vec{a}}\}_{\vec{a} \in X^\ell}$.*

Note that the VC dimension of THRESH_X^ℓ is ℓ. We obtain the following lower bound on the sample complexity of privately learning THRESH_X^ℓ.

Theorem 7.24. *For every $n, \ell \in \mathbb{N}$, $\alpha > 0$, and $\delta \leq \ell^2/(1500n^2)$, any $(\varepsilon = \frac{1}{2}, \delta)$-differentially private and $(\alpha, \beta = \frac{1}{8})$-accurate proper learner for THRESH_X^ℓ requires sample complexity $n = \Omega(\frac{\ell}{\alpha} \log^*|X|)$.*

This is the result of a general hardness amplification theorem for private proper learning. We show that if privately learning a concept class C requires sample complexity n, then learning the class C^ℓ of conjunctions of ℓ different concepts from C requires sample complexity $\Omega(\ell n)$.

Definition 7.25. *For $\ell \in \mathbb{N}$, a data universe X and a concept class C over X, define a concept class C^ℓ over X^ℓ to consist of all $\vec{c} = (c_1, \ldots, c_\ell)$, where $\vec{c} : X^\ell \to \{0,1\}$ is defined by $\vec{c}(\vec{x}) = c_1(x_1) \wedge c_2(x_2) \wedge \cdots \wedge c_\ell(x_\ell)$.*

Theorem 7.26. *Let $\alpha, \beta, \varepsilon, \delta > 0$. Let C be a concept class over a data universe X, and assume there is a domain element $p_1 \in X$ s.t. $c(p_1) = 1$ for every $c \in C$. Let $\mathcal{D}$ be a distribution over databases containing n examples from X labeled by a concept in C, and suppose that every (ε, δ)-differentially private algorithm fails to find an (α/β)-consistent hypothesis $h \in C$ for $D \sim \mathcal{D}$ with probability at least 2β. Then any (ε, δ)-differentially private and (α, β)-accurate proper learner for C^ℓ requires sample complexity $\Omega(\ell n)$.*

Note that in the above theorem we assumed the existence of a domain element $p_1 \in X$ on which every concept in C evaluates to 1. To justify the necessity of such an assumption, consider the class of *point functions* over a domain X defined as $\texttt{POINT}_X = \{c_x : x \in X\}$ where $c_x(y) = 1$ iff $y = x$. As was shown in [11], this class can be privately learned using $O_{\alpha,\beta,\varepsilon,\delta}(1)$ labeled examples (i.e., the sample complexity has no dependency in $|X|$). Observe that since there is no $x \in X$ on which every point concept evaluates to 1, we cannot use Theorem 7.26 to lower bound the sample complexity of privately learning $\texttt{POINT}_X^\ell$. Indeed, the class $\texttt{POINT}_X^\ell$ is identical (up to renaming of domain elements) to the class $\texttt{POINT}_{X\ell}$, and can be privately learned using $O_{\alpha,\beta,\varepsilon,\delta}(1)$ labeled examples.

Remark 7.27. *Similarly to Theorem 7.26 it can be shown that if privately learning a concept class C requires sample complexity n, and if there exists a domain element $p_0 \in X$ s.t. $c(p_0) = 0$ for every $c \in C$, then learning the class of* disjunctions *of ℓ concepts from C requires sample complexity ℓn.*

Proof of Theorem 7.26. Assume toward a contradiction that there exists an (ε, δ)-differentially private and (α, β)-accurate proper learner $\mathcal{A}$ for C^ℓ using $\ell n/9$ samples. Recall that the task of privately outputting a good hypothesis on any fixed database is essentially equivalent to the task of private PAC learning (see Section 7.5). We can assume, therefore, that $\mathcal{A}$ outputs an α-consistent hypothesis for every fixed database of size at least $n' \triangleq \ell n$ with probability at least $1 - \beta$.

We construct an algorithm $\mathsf{Solve}_{\mathcal{D}}$ which uses $\mathcal{A}$ in order to find an (α/β)-consistent threshold function for databases of size n from $\mathcal{D}$. Algorithm $\mathsf{Solve}_{\mathcal{D}}$ takes as input a set of n labeled examples in X and applies $\mathcal{A}$ on a database containing n' labeled examples in X^ℓ. The n input points are embedded along one random axis, and random samples from $\mathcal{D}$ are placed on each of the other axes (with n labeled points along each axis).

Algorithm $\mathsf{Solve}_{\mathcal{D}}$

Input: Database $D = (x_i, y_i)_{i=1}^n \in (X \times \{0,1\})^n$.

1. Initiate S as an empty multiset.
2. Let r be a (uniform) random element from $\{1, 2, \ldots, \ell\}$.
3. For $i = 1$ to n, let $\vec{z_i} \in X^\ell$ be the vector with r^{th} coordinate x_i, and all other coordinates p_1 (recall that every concept in C evaluates to 1 on p_1). Add to S the labeled example $(\vec{z_i}, y_i)$.
4. For every axis $t \neq r$:
   (a) Let $D' = (x'_i, y'_i)_{i=1}^n \in (X \times \{0,1\})^n$ denote a (fresh) sample from $\mathcal{D}$.
   (b) For $i = 1$ to n, let $\vec{z'_i} \in X^\ell$ be the vector whose t^{th} coordinate is x'_i, and its other coordinates are p_1. Add to S the labeled example $(\vec{z'_i}, y'_i)$.
5. Let $(h_1, h_2, \ldots, h_\ell) = \vec{h} \leftarrow \mathcal{A}(S)$.
6. Return h_r.

First observe that $\mathsf{Solve}_{\mathcal{D}}$ is (ε, δ)-differentially private. To see this, note that a change limited to one input entry affects only one entry of the multiset S. Hence, applying the (ε, δ)-differentially private algorithm $\mathcal{A}$ on S preserves privacy.

Consider the execution of $\mathsf{Solve}_{\mathcal{D}}$ on a database D of size n, sampled from $\mathcal{D}$. We first argue that $\mathcal{A}$ is applied on a multiset S correctly labeled by a concept from C^ℓ. For $1 \leq t \leq \ell$ let $(x_i^t, y_i^t)_{i=1}^n$ be the sample from $\mathcal{D}$ generated for the axis t, let $(\vec{z_i^t}, y_i^t)_{i=1}^n$ denote the corresponding elements that were added to S, and let c_t be s.t. $c_t(x_i^t) = y_i^t$ for every $1 \leq i \leq n$. Now observe that

$$(c_1, c_2, \ldots, c_\ell)(\vec{z_i^t}) = c_1(p_1) \wedge c_2(p_1) \wedge \cdots \wedge c_t(x_i^t) \wedge \cdots \wedge c_\ell(p_1) = y_i^t,$$

and hence S is perfectly labeled by $(c_1, c_2, \ldots, c_\ell) \in C^\ell$.

By the properties of $\mathcal{A}$, with probability at least $1 - \beta$ we have that $\vec{h}$ (from step 5) is an α-consistent hypothesis for S. Assuming that this is the case, there could be at most $\beta\ell$ "bad" axes on which $\vec{h}$ errs on more than $\alpha n/\beta$ points. Moreover, as r is a random axis,

and as the points along the r^{th} axis are distributed exactly like the points along the other axes, the probability that r is a "bad" axis is at most $\frac{\beta\ell}{\ell} = \beta$. Overall, $\text{Solve}_{\mathcal{D}}$ outputs an (α/β)-consistent hypothesis with probability at least $(1-\beta)^2 > 1-2\beta$. This contradicts the hardness of the distribution $\mathcal{D}$. □

Now the proof of Theorem 7.24 follows from the lower bound on the sample complexity of privately finding an α-consistent threshold function (see Section 7.2):

Lemma 7.28 (Follows from Lemma 7.6 and 7.22)**.** *There exists a constant $\lambda > 0$ s.t. the following holds. For every totally ordered data universe X there exists a distribution $\mathcal{D}$ over databases containing at most $n = \frac{\lambda}{\alpha}\log^*|X|$ labeled examples from X such that every $(\frac{1}{2}, \frac{1}{50n^2})$-differentially private algorithm fails to find an α-consistent threshold function for $D \sim \mathcal{D}$ with probability at least $\frac{1}{4}$.*

We remark that, in general, an algorithm for query release can be used to construct a private learner with similar sample complexity. Hence, Theorem 7.24 also yields the following lower bound on the sample complexity of releasing approximated answers to queries from THRESH^{ℓ}_X.

Theorem 7.29. *For every $n, \ell \in \mathbb{N}$, $\alpha > 0$, and $\delta \leq \ell^2/(7500n^2)$, any $(\frac{1}{150}, \delta)$-differentially private algorithm for releasing approximated answers for queries from THRESH^{ℓ}_X with $(\alpha, \frac{1}{150})$-accuracy must have sample complexity $n = \Omega(\frac{\ell}{\alpha}\log^*|X|)$.*

In order to prove the above theorem we use our lower bound on privately learning THRESH^{ℓ}_X together with the following reduction from private learning to query release.

Lemma 7.30 ([56, 11])**.** *Let C be a class of predicates. If there exists a $(\frac{1}{150}, \delta)$-differentially private algorithm capable of releasing queries from C with $(\frac{1}{150}, \frac{1}{150})$-accuracy and sample complexity n, then there exists a $(\frac{1}{5}, 5\delta)$-differentially private $(\frac{1}{5}, \frac{1}{5})$-accurate PAC learner for C with sample complexity $O(n)$.*

Proof of Theorem 7.29. Let $\delta \leq \ell^2/(7500n^2)$. Combining our lower bound on the sample complexity of privately learning THRESH^{ℓ}_X (Theorem 7.24) together with the reduction stated in Lemma 7.30, we get a lower bound of $m \triangleq \Omega(\ell \cdot \log^*|X|)$ on the sample complexity of every $(\frac{1}{150}, \delta)$-differentially private algorithm for releasing queries from THRESH^{ℓ}_X with $(\frac{1}{150}, \frac{1}{150})$-accuracy.

In order to refine this argument and get a bound that incorporates the approximation parameter, let $\alpha \leq 1/150$, and assume towards contradiction that there exists a $(\frac{1}{150}, \delta)$-differentially private algorithm $\tilde{\mathcal{A}}$ for releasing queries from THRESH_X^ℓ with $(\alpha, \frac{1}{150})$-accuracy and sample complexity $n < m/(150\alpha)$.

We will derive a contradiction by using $\tilde{\mathcal{A}}$ in order to construct a $(\frac{1}{150}, \frac{1}{150})$-accurate algorithm for releasing queries from THRESH_X^ℓ with sample complexity less than m. Consider the algorithm $\mathcal{A}$ that on input a database D of size $150\alpha n$, applies $\tilde{\mathcal{A}}$ on a database $\tilde{D}$ containing the elements in D together with $(1 - 150\alpha)n$ copies of $(\min X)$. Afterwards, algorithm $\mathcal{A}$ answers every query $c \in \mathsf{THRESH}_X^\ell$ with $a_c \triangleq \frac{1}{150\alpha}(\tilde{a_c} - 1 + 150\alpha)$, where $\{\tilde{a_c}\}$ are the answers received from $\tilde{\mathcal{A}}$.

Note that as $\tilde{\mathcal{A}}$ is $(\frac{1}{150}, \delta)$-differentially private, so is $\mathcal{A}$. We now show that $\mathcal{A}$'s output is $\frac{1}{150}$-accurate for D whenever $\tilde{\mathcal{A}}$'s output is α-accurate for $\tilde{D}$, which happens with all but probability $\frac{1}{150}$. Fix a query $c \in \mathsf{THRESH}_X^\ell$ and assume that $c(D) = t/(150\alpha n)$. Note that $c(\min X) = 1$, and hence, $c(\tilde{D}) = t/n + (1 - 150\alpha)$. By the utility properties of $\tilde{\mathcal{A}}$,

$$\begin{aligned} a_c &= \frac{1}{150\alpha}(\tilde{a_c} - 1 + 150\alpha) \\ &\leq \frac{1}{150\alpha}(c(\tilde{D}) + \alpha - 1 + 150\alpha) \\ &= \frac{1}{150\alpha}(t/n + \alpha) \\ &= t/(150\alpha n) + 1/150 \\ &= c(D) + 1/150. \end{aligned}$$

Similar arguments show that $a_c \geq c(D) - 1/150$, proving that $\mathcal{A}$ is $(1/150, 1/150)$-accurate and contradicting the lower bound on the sample complexity of such algorithms. □

Chapter 8

Conclusions and Open Problems

The unifying theme of this thesis is the exploration of intersection points between differential privacy and learning theory. In chapters 5, 6, and 7, we considered the task of designing private analogues to existing PAC learning algorithms, and studied the price (in terms of the sample complexity) of enforcing privacy guarantees on top of the utility guarantees. In Chapter 4, we studied the possibility of using differential privacy as a tool in order to construct *new* learning algorithms, specifically, for answering adaptively chosen queries on a distribution using i.i.d. samples from it.

8.1 The Sample Complexity of Private Learners

We have made important progress towards understanding the sample complexity of private learners. For the case of pure differential privacy, in Chapter 5, we presented a combinatorial characterization for the sample complexity of (improper) learners. Afterwards, in Chapter 6, we showed that relaxing the privacy guarantees from pure to approximate differential privacy can, in some cases, drastically reduce the necessary sample complexity. For example, while every pure-private learner for the class of threshold functions over a domain X requires $\Omega(\log|X|)$ samples [51], we construct an approximate-private proper-learner for thresholds using $2^{O(\log^*|X|)}$ samples.

> **Open Question 1:** *Construct a* generic *learner preserving approximate differential privacy and exhibiting better sample complexity than the generic constructions of pure-private learners.*

In Chapter 7 we revisit the sample complexity of privately learning thresholds, and prove that a dependency in $\log^*|X|$ is necessary – every approximate-private proper-learner

for thresholds must use $\Omega(\log^*|X|)$ samples. Closing the gap between $\log^*|X|$ and $2^{\log^*|X|}$ remains an intriguing open question.

> **Open Question 2:** *What is the sample complexity of properly learning threshold functions under* approximate *differential privacy?*

While our $\Omega(\log^*|X|)$ lower bound shows that approximate-private proper-learners can require asymptotically more samples than non-private learners, this separation is very mild. Showing a stronger separation would be very interesting.

> **Open Question 3:** *Are there cases where the gap in the sample complexity of non-private learning and approximate-private proper-learning is, say,* $\Omega(\log|X|)$*?*

Furthermore, our results still leave open the possibility that *improperly* learning thresholds with approximate differential privacy can be done using a constant sample complexity.

> **Open Question 4:** *What is the sample complexity of* improperly *learning threshold functions under* approximate *differential privacy?*

Actually, no lower bounds are currently known on the sample complexity of approximate private improper-learners, and it might be the case that the sample complexity of approximate-private improper-learners is characterized by the VC dimension, as is non-private learning.

> **Open Question 5:** *Explore the sample complexity of approximate-private* improper-*learning.*

8.2 Differential Privacy as a Tool – Adaptive Queries

An interesting aspect of the theoretical research in differential privacy is that it had produced results that are not directly focused on privacy: Some algorithmic techniques for online computation can be formulated and analyzed using the terminology of differential privacy [66], and differential privacy has proved useful as a tool for constructing economic mechanisms and games (e.g., [74, 77, 63, 79]).

A recent line of work uses differential privacy as a tool for obtaining statistical utility from data. Dwork et al. [40] model and study the problem of false discovery caused

by adaptivity in statistical analysis of data. They observe that the problem of false discovery can be related to differential privacy, a deep insight that allowed them to use tools developed for differentially private analyses for ensuring statistical validity in adaptive data analysis.

In Chapter 4, we simplified and strengthened the results of Dwork et al., providing tight bounds on the probability of false detection based on the parameters of the private computation. As our connection between differential privacy and generalization is optimal, any significant improvement to our bounds must come from using an alternative approach. We know that differential privacy is not necessary in order to avoid false discovery in adaptive data analysis; however, we do not know if a different approach can achieve better results.

> **Open Question 6:** *Can we achieve better guarantees for ensuring statistical validity in adaptive data analysis without differential privacy?*

Towards resolving this question, it is interesting to understand whether the problem can be solved using a *deterministic* mechanism. We can show that, at least in some settings, this is possible (by constructing a very unnatural deterministic mechanism that uses randomness from its input sample to instantiate a differentially private mechanism).

> **Open Question 7:** *Is there a natural* deterministic *mechanism for answering a non-trivial number of adaptive queries?*

The problem was studied from a computational perspective by Ullman, Steinke, and Hardt [61, 85], who proved computational hardness of preventing false discovery in adaptive settings. Currently, there is a gap between the lower and upper bounds in the dependency on the accuracy parameter.

> **Open Question 8:** *Close the gaps between the current upper and lower bounds for ensuring statistical validity in adaptive data analysis.*

We hope that providing answers to the above questions will lead to further understanding of how concepts like differential privacy can be used as tools for statistical inference and machine learning.

Bibliography

[1] Pankaj K. Agarwal, Sariel Har-Peled, and Kasturi R. Varadarajan. Geometric approximation via coresets. In *COMBINATORIAL AND COMPUTATIONAL GEOMETRY, MSRI*, pages 1–30. University Press, 2005.

[2] Dana Angluin and Philip Laird. Learning from noisy examples. *Machine Learning*, 2(4):343–370, 1988.

[3] Martin Anthony and John Shawe-Taylor. A result of Vapnik with applications. *Discrete Applied Mathematics*, 47(3):207–217, 1993.

[4] Matin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 2009.

[5] Peter Auer and Ronald Ortner. A new PAC bound for intersection-closed concept classes. *Machine Learning*, 66(2-3):151–163, 2007.

[6] Raef Bassily, Kobbi Nissim, Adam Smith, Thomas Steinke, Uri Stemmer, and Jonathan Ullman. Algorithmic stability for adaptive data analysis. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2016, pages 1046–1059, New York, NY, USA, 2016. ACM.

[7] Raef Bassily, Adam Smith, and Abhradeep Thakurta. Private empirical risk minimization: Efficient algorithms and tight error bounds. In *FOCS*, pages 464–473, 2014.

[8] Amos Beimel, Hai Brenner, Shiva Prasad Kasiviswanathan, and Kobbi Nissim. Bounds on the sample complexity for private learning and private data release. *Machine Learning*, 94(3):401–437, 2014.

[9] Amos Beimel, Paz Carmi, Kobbi Nissim, and Enav Weinreb. Private approximation of search problems. *SIAM J. Comput.*, 38(5):1728–1760, 2008.

[10] Amos Beimel, Kobbi Nissim, and Uri Stemmer. Characterizing the sample complexity of private learners. In *ITCS*, pages 97–110. ACM, 2013.

[11] Amos Beimel, Kobbi Nissim, and Uri Stemmer. Private learning and sanitization: Pure vs. approximate differential privacy. In *APPROX-RANDOM*, volume 8096 of *Lecture Notes in Computer Science*, pages 363–378. Springer, 2013.

[12] Amos Beimel, Kobbi Nissim, and Uri Stemmer. Learning privately with labeled and unlabeled examples. In *SODA*, pages 461–477. SIAM, 2015.

[13] Amos Beimel, Kobbi Nissim, and Uri Stemmer. Private learning and sanitization: Pure vs. approximate differential privacy. *Theory of Computing*, 12(1):1–61, 2016.

[14] Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society. Series B (Methodological)*, 57(1):289–300, 1995.

[15] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: The SuLQ framework. In Chen Li, editor, *PODS*, pages 128–138. ACM, 2005.

[16] Avrim Blum and Moritz Hardt. The ladder: A reliable leaderboard for machine learning competitions. *CoRR*, abs/1502.04585, 2015.

[17] Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to noninteractive database privacy. *J. ACM*, 60(2):12, 2013.

[18] Avrim Blum and Aaron Roth. Fast private data release algorithms for sparse queries. In *APPROX-RANDOM*, pages 395–410, 2013.

[19] Anselm Blumer, A. Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *J. ACM*, 36(4):929–965, October 1989.

[20] Carlo Emilio Bonferroni. Teoria statistica delle classi e calcolo delle probabilita. *Pubbl. d. R. Ist. Super. di Sci. Econom. e Commerciali di Firenze.*, 8, 1936.

[21] Mark Bun, Kobbi Nissim, and Uri Stemmer. Simultaneous private learning of multiple concepts. In *ITCS*, pages 369–380. ACM, 2016.

[22] Mark Bun, Kobbi Nissim, Uri Stemmer, and Salil P. Vadhan. Differentially private release and learning of threshold functions. In *FOCS*, pages 634–649, 2015.

[23] Mark Bun, Thomas Steinke, and Jonathan Ullman. Make up your mind: The price of online queries in differential privacy. *CoRR*, abs/1604.04618, 2016.

[24] Mark Bun, Jonathan Ullman, and Salil P. Vadhan. Fingerprinting codes and the price of approximate differential privacy. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 1–10, 2014.

[25] Mark Bun and Mark Zhandry. Order-revealing encryption and the hardness of private learning. In *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part I*, pages 176–206, 2016.

[26] Kamalika Chaudhuri and Daniel Hsu. Sample complexity bounds for differentially private learning. In Sham M. Kakade and Ulrike von Luxburg, editors, *COLT*, volume 19 of *JMLR Proceedings*, pages 155–186. JMLR.org, 2011.

[27] Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. Differentially private empirical risk minimization. *J. Mach. Learn. Res.*, 12:1069–1109, July 2011.

[28] Herman Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Ann. Math. Statist.*, 23:493–507, 1952.

[29] Benny Chor, Amos Fiat, and Moni Naor. Tracing traitors. In *CRYPTO*, pages 257–270. Springer, August 21-25 1994.

[30] Constantinos Daskalakis, Ilias Diakonikolas, and Rocco A. Servedio. Learning k-modal distributions via testing. *Theory of Computing*, 10:535–570, 2014.

[31] Constantinos Daskalakis, Ilias Diakonikolas, Rocco A. Servedio, Gregory Valiant, and Paul Valiant. Testing k-modal distributions: Optimal algorithms via reductions. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1833–1852, 2013.

[32] Constantinos Daskalakis and Gautam Kamath. Faster and sample near-optimal algorithms for proper learning mixtures of gaussians. In *Proceedings of The 27th Conference on Learning Theory, COLT 2014, Barcelona, Spain, June 13-15, 2014*, pages 1183–1213, 2014.

[33] Anindya De. Lower bounds in differential privacy. In Ronald Cramer, editor, *TCC*, volume 7194 of *Lecture Notes in Computer Science*, pages 321–338. Springer, 2012.

[34] Ayhan Demiriz, Kristin P. Bennett, and John Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1):225–254, 2002.

[35] Irit Dinur and Kobbi Nissim. Revealing information while preserving privacy. In *PODS*, pages 202–210. ACM, 2003.

[36] Olive Jean Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56:52–64, 1961.

[37] A. Dvoretzky, J. Kiefer, and J. Wolfowitz. Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator. *Ann. Math. Statist.*, 27(3):642–669, 09 1956.

[38] Cynthia Dwork. Differential privacy. In *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, pages 1–12, 2006.

[39] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. Generalization in adaptive data analysis and holdout reuse. In *Advances in Neural Information Processing Systems (NIPS)*, Montreal, December 2015.

[40] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. Preserving statistical validity in adaptive data analysis. In *ACM Symposium on the Theory of Computing (STOC)*. ACM, June 2015.

[41] Cynthia Dwork, Vitaly Feldman, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Aaron Roth. The reusable holdout: Preserving validity in adaptive data analysis. *Science*, 349(6248):636–638, June 2015.

[42] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In Serge Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 486–503. Springer, 2006.

[43] Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, STOC '09, pages 371–380, New York, NY, USA, 2009. ACM.

[44] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2006.

[45] Cynthia Dwork, Moni Naor, Omer Reingold, Guy N. Rothblum, and Salil P. Vadhan. On the complexity of differentially private data release: efficient algorithms and hardness results. In Michael Mitzenmacher, editor, *STOC*, pages 381–390. ACM, 2009.

[46] Cynthia Dwork and Kobbi Nissim. Privacy-preserving datamining on vertically partitioned databases. In Matthew K. Franklin, editor, *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 528–544. Springer, 2004.

[47] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[48] Cynthia Dwork, Guy N. Rothblum, and Salil P. Vadhan. Boosting and differential privacy. In *FOCS*, pages 51–60. IEEE Computer Society, 2010.

[49] Cynthia Dwork, Kunal Talwar, Abhradeep Thakurta, and Li Zhang. Analyze gauss: Optimal bounds for privacy-preserving principal component analysis. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, STOC '14, pages 11–20, New York, NY, USA, 2014. ACM.

[50] Andrzej Ehrenfeucht, David Haussler, Michael J. Kearns, and Leslie G. Valiant. A general lower bound on the number of examples needed for learning. *Inf. Comput.*, 82(3):247–261, 1989.

[51] Vitaly Feldman and David Xiao. Sample complexity bounds on differentially private learning via communication complexity. *SIAM J. Comput.*, 44(6):1740–1764, 2015.

[52] Yoav Freund. An adaptive version of the boost by majority algorithm. *Machine Learning*, 43(3):293–318, 2001.

[53] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119 – 139, 1997.

[54] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:2000, 1998.

[55] Andrew Gelman and Eric Loken. The statistical crisis in science. *American Scientist*, 102(6):460, 2014.

[56] Anupam Gupta, Moritz Hardt, Aaron Roth, and Jonathan Ullman. Privately releasing conjunctions and the statistical query barrier. In Lance Fortnow and Salil P. Vadhan, editors, *STOC*, pages 803–812. ACM, 2011.

[57] Anupam Gupta, Aaron Roth, and Jonathan Ullman. Iterative constructions and private data release. In Ronald Cramer, editor, *Theory of Cryptography*, volume 7194 of *Lecture Notes in Computer Science*, pages 339–356. Springer Berlin Heidelberg, 2012.

[58] Steve Hanneke. The optimal sample complexity of pac learning. *J. Mach. Learn. Res.*, 17(1):1319–1333, January 2016.

[59] Moritz Hardt and Guy N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *FOCS*, pages 61–70. IEEE Computer Society, 2010.

[60] Moritz Hardt and Kunal Talwar. On the geometry of differential privacy. In *STOC*, pages 705–714, 2010.

[61] Moritz Hardt and Jonathan Ullman. Preventing false discovery in interactive data analysis is hard. In *FOCS*. IEEE, October 19-21 2014.

[62] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[63] Zhiyi Huang and Sampath Kannan. The exponential mechanism for social welfare: Private, truthful, and nearly optimal. In *FOCS*, pages 140–149. IEEE Computer Society, 2012.

[64] John P. A. Ioannidis. Why most published research findings are false. *PLoS Medicine*, 2(8):124, August 2005.

[65] Peter Kairouz, Sewoong Oh, and Pramod Viswanath. The composition theorem for differential privacy. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1376–1385, 2015.

[66] Adam Kalai and Santosh Vempala. Efficient algorithms for online decision problems. *J. Comput. Syst. Sci.*, 71(3):291–307, October 2005.

[67] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. What can we learn privately? *SIAM J. Comput.*, 40(3):793–826, 2011.

[68] Michael J. Kearns. Efficient noise-tolerant learning from statistical queries. *J. ACM*, 45(6):983–1006, 1998.

[69] Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT press, Cambridge, Massachusetts, 1994.

[70] Daniel Kifer, Adam Smith, and Abhradeep Thakurta. Private convex empirical risk minimization and high-dimensional regression. In *COLT*, pages 94–103, 2012.

[71] Lucas Kowalczyk, Tal Malkin, Jonathan Ullman, and Mark Zhandry. Strong hardness of privacy from weak traitor tracing. *CoRR*, abs/1607.06141, 2016.

[72] Andrew McCallum and Kamal Nigam. Employing em and pool-based active learning for text classification. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, pages 350–358, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[73] Frank McSherry. Differential privacy for measure concentration. Blog post on "Windows on Theory". http://windowsontheory.org/2014/02/04/differential-privacy-for-measure-concentration/, 2014.

[74] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *FOCS*, pages 94–103. IEEE Computer Society, 2007.

[75] Jack Murtagh and Salil P. Vadhan. The complexity of computing the optimal composition of differential privacy. In *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part I*, pages 157–175, 2016.

[76] Kobbi Nissim, Sofya Raskhodnikova, and Adam Smith. Smooth sensitivity and sampling in private data analysis. In *STOC*, pages 75–84. ACM, 2007.

[77] Kobbi Nissim, Rann Smorodinsky, and Moshe Tennenholtz. Approximately optimal mechanism design via differential privacy. In *ITCS*, pages 203–213, 2012.

[78] Kobbi Nissim, Uri Stemmer, and Salil Vadhan. Locating a small cluster privately. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, PODS '16, pages 413–427, New York, NY, USA, 2016. ACM.

[79] Aaron Roth. Differential privacy as a tool for mechanism design in large systems. *SIGMETRICS Perform. Eval. Rev.*, 42(3):39–39, December 2014.

[80] Aaron Roth and Tim Roughgarden. Interactive privacy via the median mechanism. In *Proceedings of the 42nd ACM symposium on Theory of computing*, STOC '10, pages 765–774, New York, NY, USA, 2010. ACM.

[81] Robert E. Schapire. The strength of weak learnability. *Mach. Learn.*, 5(2):197–227, 1990.

[82] V.V. Shenmaier. The problem of a minimal ball enclosing k points. *Journal of Applied and Industrial Mathematics*, 7(3):444–448, 2013.

[83] Hans Ulrich Simon. An almost optimal PAC algorithm. In *Proceedings of The 28th Conference on Learning Theory, COLT 2015, Paris, France, July 3-6, 2015*, pages 1552–1563, 2015.

[84] Thomas Steinke and Jonathan Ullman. Between pure and approximate differential privacy. *CoRR*, abs/1501.06095, 2015.

[85] Thomas Steinke and Jonathan Ullman. Interactive fingerprinting codes and the hardness of preventing false discovery. In *COLT*, pages 1588–1628, 2015.

[86] Abhradeep Thakurta and Adam Smith. Differentially private feature selection via stability arguments, and the robustness of the lasso. In Shai Shalev-Shwartz and Ingo Steinwart, editors, *COLT*, volume 30 of *JMLR Proceedings*, pages 819–850. JMLR.org, 2013.

[87] Jonathan Ullman. Answering $n^{2+o(1)}$ counting queries with differential privacy is hard. In *STOC*, pages 361–370. ACM, June 1-4 2013.

[88] Jonathan Ullman and Salil P. Vadhan. PCPs and the hardness of generating private synthetic data. In Yuval Ishai, editor, *TCC*, volume 6597 of *Lecture Notes in Computer Science*, pages 400–416. Springer, 2011.

[89] Salil Vadhan. The complexity of differential privacy, 2016.

[90] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, November 1984.

[91] Vladimir Vapnik and Alexey Chervonenkis. Theory of pattern recognition [in russian]. *Nauka, Moscow*, 1974.

[92] Vladimir N. Vapnik and Alexey Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.