Learning Privately with Labeled and Unlabeled Examples

Amos Beimel* Kobbi Nissim† Uri Stemmer‡

Dept. of Computer Science
Ben-Gurion University of the Negev
{beimel|kobbi|stemmer}@cs.bgu.ac.il

August 8, 2014

Abstract

A *private learner* is an algorithm that given a sample of labeled individual examples outputs a generalizing hypothesis while preserving the privacy of each individual. In 2008, Kasiviswanathan et al. (FOCS 2008) gave a generic construction of private learners, in which the sample complexity is (generally) higher than what is needed for non-private learners. This gap in the sample complexity was then further studied in several followup papers, showing that (at least in some cases) this gap is unavoidable. Moreover, those papers considered ways to overcome the gap, by relaxing either the privacy or the learning guarantees of the learner.

We suggest an alternative approach, inspired by the (non-private) models of *semi-supervised learning* and *active-learning*, where the focus is on the sample complexity of *labeled* examples whereas *unlabeled* examples are of a significantly lower cost. We consider private semi-supervised learners that operate on a random sample, where only a (hopefully small) portion of this sample is labeled. The learners have no control over which of the sample elements are labeled. Our main result is that the labeled sample complexity of private learners is characterized by the VC dimension.

We present two generic constructions of private semi-supervised learners. The first construction is of learners where the labeled sample complexity is proportional to the VC dimension of the concept class, however, the unlabeled sample complexity of the algorithm is as big as the representation length of domain elements. Our second construction presents a new technique for decreasing the labeled sample complexity of a given private learner, while roughly maintaining its unlabeled sample complexity. In addition, we show that in some settings the labeled sample complexity does not depend on the privacy parameters of the learner.

*Supported by a grant from the Israeli Science and Technology ministry, by a Israel Science Foundation grant 544/13, and by the Frankel Center for Computer Science.

†Work done while the second author was a visiting scholar at the Harvard Center for Research on Computation and Society (supported by NSF grant CNS-1237235) and at the Boston University Hariri Institute for Computing and Computational Science & Engineering. Supported in part by Israel Science Foundation grant no. 276/12.

‡Supported by the Ministry of Science and Technology (Israel), by the Check Point Institute for Information Security, by the IBM PhD Fellowship Awards Program, and by the Frankel Center for Computer Science.

Contents

1 Introduction

A *private learner* is an algorithm that given a sample of labeled examples, where each example represents an individual, outputs a generalizing hypothesis while preserving the privacy of each individual. This formal notion, combining the requirements of PAC learning [28] and Differential Privacy [16], was presented in 2008 by Kasiviswanathan et al. [21], who also gave a generic construction of private learners. However, the sample complexity of the learner of [21] is (generally) higher than what is needed for non-private learners.

This gap in the sample complexity was studied in several followup papers. For *pure* differential privacy, it was shown that in some cases this gap can be closed with the price of giving up proper learning – where the output hypothesis should be from the learned concept class – for *improper* learning. Indeed, it was shown that for the class of point functions over domain of size 2^d, the sample complexity of every proper learner is $\Omega(d)$ (matching the upper bound of [21]), whereas there exist improper private learners with sample complexity $O(1)$ that use pseudorandom or pairwise independent functions as their output hypotheses [6, 7].[1] For the case of threshold functions, it was shown that with pure differential privacy even improper learners require $\Omega(d)$ samples [19] (while there exists a non-private proper learner with sample complexity $O(1)$).

Another approach for reducing the sample complexity of private learners is to relax the privacy requirement to *approximate* differential privacy. This relaxation was shown to be significant as it allows privately and *properly* learning point functions with $O(1)$ sample complexity, and threshold functions with sample complexity $2^{O(\log^* d)}$ [8].

In this work we examine an alternative approach, inspired by the (non-private) models of *semi-supervised learning* [29] (where the learning algorithm uses a small batch of labeled examples and a large batch of unlabeled examples) and *active learning* [23] (where the learning algorithm chooses which examples should be labeled). In both approaches, the focus is on reducing the sample complexity of *labeled* examples whereas it is assumed that *unlabeled* examples can be obtained with a significantly lower cost. In this vein, a recent work by Balcan and Feldman [4] suggested a generic conversion of active learners in the model of statistical queries [22] into learners that also provide differential privacy. For example, Balcan and Feldman showed an active pure-private proper learner for the class of thresholds over $X_d = \{0,1\}^d$ that uses $O(1)$ labeled examples and $O(d)$ unlabeled examples.

We present two generic constructions of private semi-supervised learners via an approach that deviates from most of the research in semi-supervised and active learning: (1) Semi-supervised learning algorithms and heuristics often rely on strong assumptions about the data, e.g., that close points are likely to be labeled similarly, that the data is clustered, or that the data lies on a low dimensional subspace of the input space. In contrast, we work in the standard PAC learning model, and need not make any further assumptions. (2) Active learners examine their pool of unlabeled data and then choose (maybe adaptively) which data examples to label. Our learners have no control over which of the sample elements are labeled.

Our main result is that the labeled sample complexity of such learners is characterized by the VC dimension. Our first generic construction is of learners where the labeled sample complexity is proportional to the VC dimension of the concept class. However, the unlabeled sample complexity of the algorithm is as big as the representation length of domain elements. The learner for a class C starts with an unlabeled database and uses private sanitization to create a synthetic database, with roughly $\mathrm{VC}(C)$ points, that can answer queries in a class related to C. It then uses this database to choose a subset of the hypotheses of size $2^{O(\mathrm{VC}(C))}$ and then uses the exponential mechanism [24] to choose from these hypotheses using the

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

$O(\mathrm{VC}(C))$ labeled examples.

As an example, applying this technique with the private sanitizer for threshold functions from [8] we get a (semi-supervised) approximate-private proper-learner for thresholds over X_d with optimal $O(1)$ labeled sample complexity and near optimal $2^{O(\log^*(d))}$ unlabeled sample complexity. This matches the labeled sample complexity of Balcan and Feldman [4] (ignoring the dependency in all parameters except for d), and improves on the unlabeled sample complexity.[2]

Our second construction presents a new technique for decreasing the labeled sample complexity of a given private learner $\mathcal{A}$. At the heart of this construction is a technique for choosing (non-privately) a hypothesis using a small labeled database; this hypothesis is used to label a bigger database, which is given to the private learner $\mathcal{A}$.

Consider, for example, the concept class $\texttt{RECTANGLE}_d^\ell$ containing all axis-aligned rectangles over ℓ dimensions, where each dimension consists of 2^d points. Applying our techniques on the learner from [8] results in a non-active semi-supervised private learner with optimal $O(\ell)$ labeled sample complexity and with $O(\ell^3 \cdot 8^{\log^*(d)})$ unlabeled sample complexity. This matches the labeled sample complexity of Balcan and Feldman [4], and improves on the unlabeled sample complexity whenever $\ell \leq \sqrt{\frac{d}{8^{\log^*(d)}}}$.

In addition, we present a stronger definition for *private* active learners and show that (most of) our learners satisfy this stronger definition, while the learners of [4] do not. We also show that in some settings the labeled sample complexity does not depend on the privacy parameters of the learner.

Related Work. Differential privacy was defined in [16] and the relaxation to approximate differential privacy is from [15]. Most related to our work is the work on private learning and its sample complexity [9, 21, 12, 17, 5, 7, 8, 19] and the early work on sanitization [10]. Blum et al. [9] showed that computationally efficient private-learners exist for all concept classes that can be efficiently learned in the *statistical queries* model [22]. Kasiviswanathan et al. [21] 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. These positive results show that many "natural" learning tasks that are efficiently learned non-privately can be learned privately and efficiently.

Chaudhuri and Hsu [12] studied the sample complexity needed for private learning when the data is drawn from a continuous domain. They showed that under these settings there exists a simple concept class for which any proper learner that uses a finite number of examples and guarantees pure-privacy fails to satisfy accuracy guarantee for at least one data distribution.

A complete characterization for the sample complexity of pure-private learners was recently given in [7], in terms of a new dimension – the *Representation Dimension*, that is, given a class C, the number of samples needed and sufficient for privately learning C is $\Theta(\mathrm{RepDim}(C))$. Following [7], Feldman and Xiao [19] 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.

Dwork et al. [17] showed how to boost the accuracy of private learning algorithms. That is, given a *private* learning algorithm that has a big classification error, they produced a *private* learning algorithm with small error. Other tools for private learning include, e.g., private SVM [25], private logistic regression [13], and private empirical risk minimization [14].

[2] We remark that – unlike this work – the focus in [4] is on the dependency of the labeled sample complexity in the approximation parameter. As our learners are non-active, their labeled sample complexity is lower bounded by $\Omega(\frac{1}{\alpha})$ (where α is the approximation parameter).

2 Preliminaries

In this section we define differential privacy and semi-supervised (private) learning. Additional preliminaries on the VC dimension and on data sanitization are deferred to the appendix.

Notation. We use $O_\gamma(g(n))$ as a shorthand for $O(h(\gamma) \cdot g(n))$ for some non-negative function h. In informal discussions, we sometimes write $\widetilde{O}(g(n))$ to indicate that $g(n)$ is missing lower order terms. We use X to denote an arbitrary domain, and X_d for the domain $\{0,1\}^d$.

Differential Privacy. Consider a database where each entry contains information pertaining to an individual. An algorithm operating on such databases is said to preserve *differential privacy* if its outcome is insensitive to any modification in a single entry. Formally:

Definition 2.1 (Differential Privacy [16, 15])**.** *Databases $S_1 \in X^m$ and $S_2 \in X^m$ over a domain X are called* neighboring *if they differ in exactly one entry. A randomized algorithm $\mathcal{A}$ is (ϵ, δ)-differentially private if for all neighboring databases $S_1, S_2 \in X^m$, and for all sets F of outputs,*

$$\Pr[\mathcal{A}(S_1) \in F] \leq \exp(\epsilon) \cdot \Pr[\mathcal{A}(S_2) \in F] + \delta. \tag{1}$$

The probability is taken over the random coins of $\mathcal{A}$. When $\delta{=}0$ we omit it and say that $\mathcal{A}$ preserves pure *differential privacy, otherwise (when $\delta > 0$) we say that $\mathcal{A}$ preserves* approximate *differential privacy.*

See Appendices A and B for basic differentially private mechanisms.

Semi-Supervised PAC Learning. The standard PAC model (and similarly private PAC) focuses on learning a class of concepts from a sample of labeled examples. In a situation where labeled examples are significantly more costly than unlabeled ones, it is natural to attempt to use a combination of labeled and unlabeled data to reduce the number of labeled examples needed. Such learners may have no control over which of the examples are labeled, as in *semi-supervised learning*, or may specifically choose which examples to label, as in *active learning*. In this section we focus on semi-supervised learning. Active learning will be discussed in Section 5.

A concept $c : X \to \{0,1\}$ is a predicate that labels *examples* taken from the domain X by either 0 or 1. A *concept class* C over X is a set of concepts (predicates) mapping X to $\{0,1\}$. A semi-supervised learner is given n examples sampled according to an unknown probability distribution μ over X, where $m \leq n$ of these examples are labeled according to an unknown *target* concept $c \in C$. The learner succeeds if it outputs a hypothesis h that approximates the target concept well according to the distribution μ. Formally:

Definition 2.2. *Let c and μ be a concept and a distribution over a domain X. The* generalization error *of a hypothesis $h : X \to \{0,1\}$ w.r.t. c and μ is defined as $\operatorname{error}_\mu(c,h) = \Pr_{x\sim\mu}[h(x) \neq c(x)]$. When $\operatorname{error}_\mu(c,h) \leq \alpha$ we say that h is* α-good *for c and μ.*

Definition 2.3 (Semi-Supervised Learning [28, 29])**.** *Let C be a concept classes over a domain X, and let $\mathcal{A}$ be an algorithm operating on (partially) labeled databases. Algorithm $\mathcal{A}$ is an (α, β, n, m)-*SSL (semi-supervised learner) *for C if for all concepts $c \in C$ and all distributions μ on X the following holds.*

Let $D = (x_i, y_i)_{i=1}^n \in (X \times \{0,1,\perp\})^n$ be a database s.t. (1) each x_i is drawn i.i.d. from μ; (2) in the first m entries $y_i = c(x_i)$; (3) in the last $(n-m)$ entries $y_i = \perp$. Then,

$$\Pr[\mathcal{A}(D){=}h \text{ s.t. } \operatorname{error}_\mu(c,h) > \alpha] \leq \beta.$$

The probability is taken over the choice of the samples from μ and the coin tosses of $\mathcal{A}$.

If a semi-supervised learner is restricted to only output hypotheses from the target concept class C, then it is called a *proper* learner. Otherwise, it is called an *improper* learner. We sometimes refer to the input for a semi-supervised learner as two databases $D \in (X \times \{\perp\})^{n-m}$ and $S \in (X \times \{0,1\})^m$, where m and n are the *labeled* and *unlabeled* sample complexities of the learner.

Definition 2.4. *Given a* labeled *sample* $S = (x_i, y_i)_{i=1}^m$*, the* empirical error *of a hypothesis* h *on* S *is* $\mathrm{error}_S(h) = \frac{1}{m}|\{i : h(x_i) \neq y_i\}|$*. Given an* unlabeled *sample* $D = (x_i)_{i=1}^m$ *and a target concept* c*, the* empirical error *of* h *w.r.t.* D *and* c *is* $\mathrm{error}_D(h, c) = \frac{1}{m}|\{i : h(x_i) \neq c(x_i)\}|$.

Semi-supervised learning algorithms operate on a (partially) labeled sample with the goal of choosing a hypothesis with a small *generalization* error. Standard arguments in learning theory (see Appendix B) state that the generalization of a hypothesis h and its *empirical* error (observed on a large enough sample) are similar. Hence, in order to output a hypothesis with small generalization error it suffices to output a hypothesis with small empirical error.

Agnostic Learner. Consider an SSL for an *unknown* class C that uses a (known) hypotheses class H. If $H \neq C$, then a hypothesis with small empirical error might not exist in H. Such learners are referred to in the literature as *agnostic*-learners, and are only required to produce a hypothesis $f \in H$ (approximately) minimizing $\mathrm{error}_\mu(c, f)$, where c is the (unknown) target concept.

Definition 2.5 (Agnostic Semi-Supervised Learning)**.** *Let* H *be a concept classes over a domain* X*, and let* $\mathcal{A}$ *be an algorithm operating on (partially) labeled databases. Algorithm* $\mathcal{A}$ *is an* (α, β, n, m)-agnostic-SSL *using* H *if for all concepts* c *(not necessarily in* H*) and all distributions* μ *on* X *the following holds.*

Let $D = (x_i, y_i)_{i=1}^n \in (X \times \{0, 1, \perp\})^n$ *be a database s.t. (1) each* x_i *is drawn i.i.d. from* μ*; (2) in the first* m *entries* $y_i = c(x_i)$*; (3) in the last* $(n - m)$ *entries* $y_i = \perp$*. Then,* $\mathcal{A}(D)$ *outputs a hypothesis* $h \in H$ *satisfying* $\Pr[\mathrm{error}_\mu(c, h) \leq \min_{f \in H}\{\mathrm{error}_\mu(c, f)\} + \alpha] \geq 1 - \beta$*. The probability is taken over the choice of the samples from* μ *and the coin tosses of* $\mathcal{A}$*.*

Private Semi-Supervised PAC learning. Similarly to [21] we define private semi-supervised learning as the combination of Definitions 2.1 and 2.3.

Definition 2.6 (Private Semi-Supervised Learning)**.** *Let* $\mathcal{A}$ *be an algorithm that gets an input* $S \in (X \times \{0, 1, \perp\})^n$*. Algorithm* $\mathcal{A}$ *is an* $(\alpha, \beta, \epsilon, \delta, n, m)$-PSSL (private SSL) *for a concept class* C *over* X *if* $\mathcal{A}$ *is an* (α, β, n, m)*-SSL for* C and $\mathcal{A}$ *is* (ϵ, δ)*-differentially private.*

Active Learning. Semi-supervised learners are a subset of the larger family of *active learners*. Such learners can adaptively request to reveal the labels of specific examples. See formal definition and discussion in Section 5.

3 A Generic Construction Achieving Low Labeled Sample Complexity

We next study the labeled sample complexity of private semi-supervised learners. We begin with a generic algorithm showing that for every concept class C there exist a pure-private proper-learner with labeled sample complexity (roughly) $\mathrm{VC}(C)$. This algorithm, called $GenericLearner$, is described in Algorithm 1. The algorithm operates on a labeled database S and on an unlabeled database D. First, the algorithm produces a sanitization $\widetilde{D}$ of the unlabeled database D w.r.t. $C^\oplus$ (to be defined). Afterwards, the algorithm

uses $\widetilde{D}$ to construct a small set of hypotheses H (we will show that H contains at least one good hypothesis). Finally, the algorithm uses the exponential mechanism to choose a hypothesis out of H.

Definition 3.1. *Given two concepts $h, f \in C$, we denote $(h \oplus f) : X_d \to \{0,1\}$, where $(h \oplus f)(x) = 1$ if and only if $h(x) \neq f(x)$. Let $C^{\oplus} = \{(h \oplus f) \ : \ h, f \in C\}$.*

To preserve the privacy of the examples in D, we first create a sanitized version of it – $\widetilde{D}$. If the entries of D are drawn i.i.d. according to the underlying distribution (and if D is big enough), then a hypothesis with small empirical error on D also has small generalization error (see Theorem B.6). Our learner classifies the sanitized database $\widetilde{D}$ with small error, thus we require that a small error on $\widetilde{D}$ implies a small error on D. Specifically, if c is the target concept, then we require that for every $f \in C$, $\operatorname{error}_D(f,c) = \frac{1}{|D|}\left|\{x \in D \ : \ f(x) \neq c(x)\}\right|$ is approximately the same as $\operatorname{error}_{\widetilde{D}}(f,c) = \frac{1}{|\widetilde{D}|}\left|\{x \in \widetilde{D} \ : \ f(x) \neq c(x)\}\right|$. Observe that this is exactly what we would get from a sanitization of D w.r.t. the concept class $C^{\oplus c} = \{(f \oplus c) \ : \ f \in C\}$. As the target concept c is unknown, we let $\widetilde{D}$ be a sanitization of D w.r.t. $C^{\oplus}$, which contains $C^{\oplus c}$.

To apply the sanitization of Blum et al. [10] to D w.r.t. the class $C^{\oplus}$, we analyze the VC dimension of $C^{\oplus}$ in the next observation, whose proof appears in Appendix C.

Observation 3.2. *For any concept class C over X_d it holds that $\mathrm{VC}(C^{\oplus}) = O(\mathrm{VC}(C))$.*

Theorem 3.3. *Let C be a concept class over X_d. For every α, β, ϵ, there exists an $(\alpha, \beta, \epsilon, \delta{=}0, n, m)$-private semi-supervised proper-learner for C, where $m = O\left(\frac{\mathrm{VC}(C)}{\alpha^3\epsilon}\log(\frac{1}{\alpha}) + \frac{1}{\alpha\epsilon}\log(\frac{1}{\beta})\right)$, and $n = O\left(\frac{d \cdot \mathrm{VC}(C)}{\alpha^3\epsilon}\log(\frac{1}{\alpha}) + \frac{1}{\alpha\epsilon}\log(\frac{1}{\beta})\right)$. The learner might not be efficient.*

The proof of Theorem 3.3 is via the construction of algorithm $GenericLearner$ (Algorithm 1). See Appendix C for the complete proof.

Algorithm 1 $GenericLearner$

Input: parameter ϵ, an unlabeled database $D = (x_i)_{i=1}^{n-m}$, and a labeled database $S = (x_i, y_i)_{i=1}^{m}$.

1. Initialize $H = \emptyset$.
2. Construct an ϵ-private sanitization $\widetilde{D}$ of D w.r.t. $C^{\oplus}$, where $|\widetilde{D}| = O\left(\frac{\mathrm{VC}(C^{\oplus})}{\alpha^2}\log(\frac{1}{\alpha})\right) = O\left(\frac{\mathrm{VC}(C)}{\alpha^2}\log(\frac{1}{\alpha})\right)$ (e.g., using Theorem A.3).
3. Let $B = \{b_1, \ldots, b_\ell\}$ be the set of all (unlabeled) points appearing at least once in $\widetilde{D}$.
4. For every $(z_1, \ldots, z_\ell) \in \Pi_C(B) = \{(c(j_1), \ldots, c(j_\ell)) : c \in C\}$, add to H an arbitrary concept $c \in C$ s.t. $c(b_i) = z_i$ for every $1 \leq i \leq \ell$.
5. Choose and return $h \in H$ using the exponential mechanism with inputs ϵ, H, S.

Note that the labeled sample complexity in Theorem 3.3 is optimal (ignoring the dependency in α, β, ϵ), as even without the privacy requirement every PAC learner for a class C must have *labeled* sample complexity $\Omega(\mathrm{VC}(C))$. However, the unlabeled sample complexity is as big as the representation length of domain

elements, that is, $O(d \cdot \mathrm{VC}(C))$. Such a blowup in the unlabeled sample complexity is unavoidable in any generic construction of pure-private learners.[3]

To show the usefulness of Theorem 3.3, we consider the concept class $\mathtt{THRESH}_d$ defined as follows. For $0 \leq j \leq 2^d$ let $c_j : X_d \to \{0,1\}$ be defined as $c_j(x) = 1$ if $x < j$ and $c_j(x) = 0$ otherwise. Define the concept class $\mathtt{THRESH}_d = \{c_j : 0 \leq j \leq 2^d\}$. Balcan and Feldman [4] showed an efficient pure-private proper-learner for $\mathtt{THRESH}_d$ with labeled sample complexity $O_{\alpha,\beta,\epsilon}(1)$ and unlabeled sample complexity $O_{\alpha,\beta,\epsilon}(d)$. At the cost of preserving approximate-privacy, and using the efficient approximate-private sanitizer for intervals from [8] (in Step 2 of Algorithm $GenericLearner$) instead on the sanitizer of [10], we get the following lemma (as $GenericLearner$ requires unlabeled examples only in Step 2, and the sanitizer of [8] requires a database of size $\widetilde{O}_{\alpha,\beta,\epsilon,\delta}(8^{\log^* d})$).

Lemma 3.4. *There exists an efficient approximate-private proper-learner for* $\mathtt{THRESH}_d$ *with labeled sample complexity* $O_{\alpha,\beta,\epsilon}(1)$ *and unlabeled sample complexity* $\widetilde{O}_{\alpha,\beta,\epsilon,\delta}(8^{\log^* d})$.

Beimel et al. [8] showed an efficient approximate-private proper-learner for $\mathtt{THRESH}_d$ with (both labeled and unlabeled) sample complexity $\widetilde{O}_{\alpha,\beta,\epsilon,\delta}(16^{\log^* d})$. The learner from Lemma 3.4 has similar unlabeled sample complexity, but improves on the labeled complexity.

4 Boosting the Labeled Sample Complexity of Private Learners

We now show a generic transformation of a private learning algorithm $\mathcal{A}$ for a class C into a private learner with reduced labeled sample complexity (roughly $\mathrm{VC}(C)$), while maintaining its unlabeled sample complexity. This transformation could be applied to a proper or an improper learner, and to a learner that preserves pure or approximated privacy.

The main ingredient of the transformation is algorithm $LabelBoost$ (Algorithm 2), where the labeled sample complexity is reduced logarithmically. In Algorithm $IterLabelBoost$, presented in Appendix C, we use this transformation iteratively to get our learner with label complexity $O_{\alpha,\beta,\epsilon}(\mathrm{VC}(C))$.

Given a partially labeled sample B of size n, algorithm $LabelBoost$ chooses a small subset H of C that strongly depends on the points in B so outputting a hypothesis $h \in H$ may breach privacy. Nevertheless, $LabelBoost$ does choose a good hypothesis $h \in H$ (using the exponential mechanism), but instead of outputting h it relabels part of the sample B using h and applies $\mathcal{A}$ on the relabeled sample. In Lemma 4.1, we analyze the privacy guarantees of Algorithm $LabelBoost$. We do not know if Algorithm $LabelBoost$ is a learner. To achieve a learner we add sampling stages to the algorithm. This is done in Algorithm $InterLabelBoost$ appearing Appendix C. Algorithm $InterLabelBoost$, in addition, also applyies Algorithm $LabelBoost$ iteratively in order to further reduce the labeled sample complexoty.

Lemma 4.1. *If* $\mathcal{A}$ *is* (ϵ, δ)*-differentially private, then* $LabelBoost$ *is* $(\epsilon + 3, 4e\delta)$*-differentially private.*

Proof. Consider the executions of $LabelBoost$ on two neighboring inputs $S_1 \circ T_1 \circ D_1$ and $S_2 \circ T_2 \circ D_2$. If these two neighboring inputs differ (only) on the last portion D then the execution of $LabelBoost$ on these neighboring inputs differs only in Step 6, and hence Inequality (1) (approximate differential privacy) follows from the privacy of $\mathcal{A}$. We, therefore, assume that $D_1 = D_2 = D$ (and that $S_1 \circ T_1, S_2 \circ T_2$ differ in at most one entry).

[3] Feldman and Xiao [19] showed an example of a concept class C over X_d for which every pure-private learner must have unlabeled sample complexity $\Omega(\mathrm{VC}(C) \cdot d)$. Hence, as a function of d and $\mathrm{VC}(C)$, the unlabeled sample complexity in Theorem 3.3 is the best possible for a generic construction of pure-private learners.

Algorithm 2 $LabelBoost$

Setting: Algorithm $\mathcal{A}$ operating on partially labeled databases of size n.

Input: A partially labeled databases $B = S \circ T \circ D \in (X \times \{0, 1, \perp\})^n$.

% We assume that the first portion of B (denoted as S) contains labeled examples. Our goal is to apply $\mathcal{A}$ on a similar database where both S and T are labeled.

1. Initialize $H = \emptyset$.
2. Let $P = \{p_1, \ldots, p_\ell\}$ be the set of all points $p \in X$ appearing at least once in $S \circ T$.
3. For every $(z_1, \ldots, z_\ell) \in \Pi_C(P) = \{(c(p_1), \ldots, c(p_\ell)) : c \in C\}$, add to H an arbitrary concept $c \in C$ s.t. $c(p_i) = z_i$ for every $1 \leq i \leq \ell$.
4. Choose $h \in H$ using the exponential mechanism with privacy parameter ϵ=1, solution set H, and the database S.
5. Relabel $S \circ T$ using h, and denote this relabeled database as $(S \circ T)^h$, that is, if $S \circ T = (x_i, y_i)_{i=1}^t$ then $(S \circ T)^h = (x_i, y'_i)_{i=1}^t$ where $y'_i = h(x_i)$.
6. Execute $\mathcal{A}$ on $(S \circ T)^h \circ D$.

Denote by H_1, P_1 and by H_2, P_2 the elements H, P as they are in the executions of $LabelBoost$ on $S_1 \circ T_1 \circ D$ and on $S_2 \circ T_2 \circ D$. The main difficulty in proving differential privacy is that H_1 and H_2 can significantly differ. We show, however, that the distribution on relabeled databases $(S \circ T)^h$ generated in Step 5 of the two executions are similar in the sense that for each relabeled database in one of the distributions there exist one or two databases in the other s.t. (1) all these databases have, roughly, the same probability, and (2) they differ on at most one entry. Thus, executing the differentially private algorithm $\mathcal{A}$ in Step 6 preserves differential privacy. We now make this argument formal.

Note that $|P_1 \setminus P_2| \in \{0, 1\}$, and let p_1 be the element in $P_1 \setminus P_2$ if such an element exists. If this is the case, then p_1 appears exactly once in $S_1 \circ T_1$. Similarly, let p_2 be the element in $P_2 \setminus P_1$ if such an element exists. Let $K = P_1 \cap P_2$, hence $P_i = K$ or $P_i = K \cup \{\mathsf{p}_i\}$. Therefore, $|\Pi_C(K)| \leq |\Pi_C(P_i)| \leq 2|\Pi_C(K)|$. Thus, $|H_1| \leq 2|H_2|$ and similarly $|H_2| \leq 2|H_1|$.

More specifically, for every $\vec{z} \in \Pi_C(K)$ there are either one or two (but not more) hypotheses in H_1 that agree with $\vec{z}$ on K. We denote these one or two hypotheses by $h_{1,\vec{z}}$ and $h'_{1,\vec{z}}$, which may be identical if only one unique hypothesis exists. Similarly, we denote $h_{2,\vec{z}}$ and $h'_{2,\vec{z}}$ as the hypotheses corresponding to H_2. For every $\vec{z} \in \Pi_C(K)$ we have that $|q(S_i, h_{i,\vec{z}}) - q(S_i, h'_{i,\vec{z}})| \leq 1$ because if $h_{i,\vec{z}} = h'_{i,\vec{z}}$ then the difference is clearly zero and otherwise they differ only on p_i, which appears at most once in S_i. Moreover, for every $\vec{z} \in \Pi_C(K)$ we have that $|q(S_1, h_{1,\vec{z}}) - q(S_2, h_{2,\vec{z}})| \leq 1$ because $h_{1,\vec{z}}$ and $h_{2,\vec{z}}$ disagree on at most two points $\mathsf{p}_1, \mathsf{p}_2$ such that at most one of them appears in S_1 and at most one of them appears in S_2. The same is true for every pair in $\{h_{1,\vec{z}}, h'_{1,\vec{z}}\} \times \{h_{2,\vec{z}}, h'_{2,\vec{z}}\}$.

Let $w_{i,\vec{z}}$ be the probability that the exponential mechanism chooses $h_{i,\vec{z}}$ or $h'_{i,\vec{z}}$ in Step 4 of the execution

on $S_i \circ T_i \circ D$. We get that for every $\vec{z} \in \Pi_C(K)$,

$$\begin{aligned}
w_{1,\vec{z}} &\leq \frac{\exp(\frac{1}{2} \cdot q(S_1, h_{1,\vec{z}})) + \exp(\frac{1}{2} \cdot q(S_1, h'_{1,\vec{z}}))}{\sum_{f \in H_1} \exp(\frac{1}{2} \cdot q(S_1, f))} \\
&\leq \frac{\exp(\frac{1}{2} \cdot q(S_1, h_{1,\vec{z}})) + \exp(\frac{1}{2} \cdot q(S_1, h'_{1,\vec{z}}))}{\sum_{\vec{r} \in \Pi_C(K)} \exp(\frac{1}{2} \cdot q(S_1, h_{1,\vec{r}}))} \\
&\leq \frac{\exp(\frac{1}{2} \cdot [q(S_2, h_{2,\vec{z}}) + 1]) + \exp(\frac{1}{2} \cdot [q(S_2, h'_{2,\vec{z}}) + 1])}{\frac{1}{2} \sum_{\vec{r} \in \Pi_C(K)} \left(\exp(\frac{1}{2} \cdot [q(S_2, h_{2,\vec{r}}) - 1]) + \exp(\frac{1}{2} \cdot [q(S_2, h'_{2,\vec{r}}) - 1]) \right)} \\
&\leq 2e \cdot \frac{\exp(\frac{1}{2} \cdot [q(S_2, h_{2,\vec{z}})]) + \exp(\frac{1}{2} \cdot [q(S_2, h'_{2,\vec{z}})])}{\sum_{f \in H_2} \exp(\frac{1}{2} \cdot q(S_2, f))} \quad \leq \quad 4e \cdot w_{2,\vec{z}}.
\end{aligned}$$

We can now conclude the proof by noting that for every $\vec{z} \in \Pi_C(K)$ the databases $(S_1 \circ T_1)^{h_{1,\vec{z}}}$ and $(S_2 \circ T_2)^{h_{2,\vec{z}}}$ are neighboring, and, therefore, $(S_1 \circ T_1)^{h_{1,\vec{z}}} \circ D$ and $(S_2 \circ T_2)^{h_{2,\vec{z}}} \circ D$ are neighboring. Hence, by the privacy properties of algorithm $\mathcal{A}$ we have that for any set F of possible outputs of algorithm $LabelBoost$

$$\begin{aligned}
\Pr[LabelBoost\,(S_1 \circ T_1 \circ D) \in F] &= \sum_{\vec{z} \in \Pi_C(K)} w_{1,\vec{z}} \cdot \Pr[\mathcal{A}\left((S_1 \circ T_1)^h \circ D\right) \in F | h \in \{h_{1,\vec{z}}, h'_{1,\vec{z}}\}] \\
&\leq \sum_{\vec{z} \in \Pi_C(K)} 4e \cdot w_{2,\vec{z}} \cdot \left(e^{\epsilon} \cdot \Pr[\mathcal{A}\left((S_2 \circ T_2)^h \circ D\right) \in F | h \in \{h_{2,\vec{z}}, h'_{2,\vec{z}}\}] + \delta \right) \\
&\leq e^{\epsilon+3} \cdot \Pr[LabelBoost\,(S_2 \circ T_2 \circ D) \in F] + 4e\delta.
\end{aligned}$$

□

Consider an execution of algorithm $LabelBoost$ on a database $S \circ T \circ D$, and assume that the examples in S are labeled by some target concept $c \in C$. Recall that for every possible labeling $\vec{z}$ of the elements in S and in T, algorithm $LabelBoost$ adds to H a hypothesis from C that agrees with $\vec{z}$. In particular, H contains a hypothesis that agrees with the target concept c on S (and on T). That is, $\exists f \in H$ s.t. $\mathrm{error}_S(f) = 0$. Hence, the exponential mechanism (on Step 4) chooses (w.h.p.) a hypothesis $h \in H$ s.t. $\mathrm{error}_S(h)$ is small, provided that $|S|$ is roughly $\log |H|$, which is roughly $\mathrm{VC}(C) \cdot \log(|S| + |T|)$ by Sauer's lemma. So, algorithm $LabelBoost$ takes an input database where only a small portion of it is labeled, and applies $\mathcal{A}$ a similar database in which the labeled portion grows exponentially.

It Appendix C we embed algorithm $LabelBoost$ in a wrapper algorithm, called $IterLabelBoost$, that iteratively applies $LabelBoost$ in order to enlarge the labeled portion of the database. This results in the following theorem.

Theorem 4.2. *Fix α, β, ϵ. Applying $IterLabelBoost$ on an $(\alpha', \beta', \epsilon'=1, \delta, n, n)$-PSSL for a class C results in an $(\alpha + \alpha', \beta + \beta', \epsilon, \delta, O(\frac{n}{\epsilon}), m)$-PSSL for C, where $m = O(\frac{1}{\alpha\epsilon} \mathrm{VC}(C) \log(\frac{1}{\alpha\beta}))$.*

See Appendix C for the details. In a nutshell, the learner $IterLabelBoost$ could be described as follows. It starts by training on the given labeled data. In each step, a part of the unlabeled points is labeled using the current hypothesis (previously labeled points are also relabeled); then the learner retrains using its own predictions as a (larger) labeled sample. Variants of this idea (known as self-training) have appeared in the literature for non-private learners (e.g., [27, 20, 1]). As stated in Theorem 4.2, in the context of *private* learners, this technique provably reduces the labeled sample complexity (while maintaining utility).

Remark 4.3. *Let $\mathcal{B}$ be the learner resulting from applying $IterLabelBoost$ on a learner $\mathcal{A}$. Then (1) If $\mathcal{A}$ preserves pure-privacy, then so does $\mathcal{B}$; and (2) If $\mathcal{A}$ is a proper-learner, then so is $\mathcal{B}$.*

Algorithm $IterLabelBoost$ can also be used as an *agnostic* learner, where the target class C is unknown, and the learner outputs a hypothesis out of a set $F \neq C$. Note that given a labeled sample, a consistent hypothesis might not exist in F. Minor changes in the proof of Theorem 4.2 show the following theorem.

Theorem 4.4. *Fix α, β, ϵ. Applying $IterLabelBoost$ on an $(\alpha', \beta', \epsilon'{=}1, \delta, n, n)$-PSSL for a class F results in an $(\alpha + \alpha', \beta + \beta', \epsilon, \delta, O(\frac{n}{\epsilon}), m)$-agnostic-PSSL using F, where $m = O(\frac{1}{\alpha^2 \epsilon} \operatorname{VC}(F) \log(\frac{1}{\alpha\beta}))$.*

To show the usefulness of Theorem 4.2, we consider (a discrete version of) the class of all axis-aligned rectangles (or hyperrectangles) in ℓ dimensions. Formally, let $X_d^\ell = (\{0,1\}^d)^\ell$ denote a discrete ℓ-dimensional domain, in which every axis consists of 2^d points. For every $\vec{a} = (a_1, \ldots, a_\ell), \vec{b} = (b_1, \ldots, b_\ell) \in X_d^\ell$ define the concept $c_{[\vec{a},\vec{b}]} : X_d^\ell \to \{0,1\}$ where $c_{[\vec{a},\vec{b}]}(\vec{x}) = 1$ if and only if for every $1 \leq i \leq \ell$ it holds that $a_i \leq x_i \leq b_i$. Define the concept class of all axis-aligned rectangles over X_d^ℓ as $\texttt{RECTANGLE}_d^\ell = \{c_{[\vec{a},\vec{b}]}\}_{\vec{a},\vec{b} \in X_d^\ell}$. The VC dimension of this class is 2ℓ, and, thus, it can be learned non-privately with (labeled and unlabeled) sample complexity $O_{\alpha,\beta}(\ell)$. The best currently known private PAC learner for this class [8] has (labeled and unlabeled) sample complexity $\widetilde{O}_{\alpha,\beta,\epsilon,\delta}(\ell^3 \cdot 8^{\log^*(d)})$. Using $IterLabelBoost$ with the construction of [8] reduces the labeled sample complexity while maintaining the unlabeled sample complexity.

Corollary 4.5. *There exists a private semi-supervised learner for $\texttt{RECTANGLE}_d^\ell$ with unlabeled sample complexity $\widetilde{O}_{\alpha,\beta,\epsilon,\delta}(\ell^3 \cdot 8^{\log^*(d)})$ and labeled sample complexity $O_{\alpha,\beta,\epsilon}(\ell)$. The learner is efficient (runs in polynomial time) whenever the dimension ℓ is small enough (roughly, $\ell \leq \log^{\frac{1}{3}}(d)$).*

The *labeled* sample complexity in Theorem 4.2 has no dependency in δ.[4] It would be helpful if we could also reduce the dependency on ϵ. As we will later see, this can be achieved in the active learning model.

IterLabelBoost vs. GenericLearner. While both constructions result in learners with labeled sample complexity proportional to the VC dimension, their constructions are different and result in different unlabeled sample complexity.

We can apply $IterLabelBoost$ to the generic construction for private PAC learners of Kasiviswanathan et al. [21], in which the (labeled and unlabeled) sample complexity is logarithmic in the size of the target concept class C (better constructions are known for many specific cases). Using Algorithm $IterLabelBoost$ with their generic construction results in a private semi-supervised learner with unlabeled sample complexity (roughly) $\log|C|$, which is better than the bound achieved by $GenericLearner$ (whose unlabeled sample complexity is $O(\log|X| \cdot \operatorname{VC}(C))$). In cases where a sample-efficient private-PAC learner is known, applying $IterLabelBoost$ would give even better bounds.

Another difference is that (a direct use of) $GenericLearner$ only yields pure-private proper-learners, where as $IterLabelBoost$ could be applied to every private learner (proper or improper, preserving pure or approximated privacy). To emphasize this difference, recall that the sample complexity of pure-private improper-PAC-learners is characterized by the Representation Dimension [7].

Corollary 4.6. *For every concept class C there exists a pure-private semi-supervised improper-learner with labeled sample complexity $O_{\alpha,\beta,\epsilon}(\operatorname{VC}(C))$ and unlabeled sample complexity $\widetilde{O}_{\alpha,\beta,\epsilon}(\operatorname{RepDim}(C))$.*

[4] The unlabeled sample complexity depends on δ as n depends on δ.

5 Private Active Learners

Semi-supervised learners are a subset of the larger family of active learners. Such learners can adaptively request to reveal the labels of specific examples. An active learner is given access to a pool of n unlabeled examples, and adaptively chooses to label m examples.

Definition 5.1 (Active Learning [23])**.** *Let C be a concept classes over a domain X. Let $\mathcal{A}$ be an interactive (stateful) algorithm that holds an initial input database $D = (x_i)_{i=1}^n \in (X)^n$. For at most m rounds, algorithm $\mathcal{A}$ outputs an index $i \in \{1, 2, \ldots, n\}$ and receives an answer $y_i \in \{0, 1\}$. Afterwards, algorithm $\mathcal{A}$ outputs a hypothesis h, and terminates.*

Algorithm $\mathcal{A}$ is an (α, β, n, m)-AL (Active learner) *for C if for all concepts $c \in C$ and all distributions μ on X: If $\mathcal{A}$ is initiated on an input $D = (x_i)_{i=1}^n$, where each x_i is drawn i.i.d. from μ, and if every index i queried by $\mathcal{A}$ is answered by $y_i = c(x_i)$, then algorithm $\mathcal{A}$ outputs a hypothesis h satisfying $\Pr[\mathrm{error}_\mu(c, h) \leq \alpha] \geq 1 - \beta$. The probability is taken over the random choice of the samples from μ and the coin tosses of the learner $\mathcal{A}$.*

Remark 5.2. *In the standard definition of active learners the learners specify examples by their value (whereas in Definition 5.1 the learner queries the labels of examples by their index). E.g., if $x_5 = x_9 = p$ then instead of asking for the label of p, algorithm $\mathcal{A}$ asks for the label example 5 (or 9). This deviation from the standard definition is because when privacy is introduced, every entry in D corresponds to a single individual, and can be changed arbitrarily (and regardless of the other entries).*

Definition 5.3 (Private Active Learner [4])**.** *An algorithm $\mathcal{A}$ is an* $(\alpha, \beta, \epsilon, \delta, n, m)$-PAL *(Private Active Learner) for a concept class C if Algorithm $\mathcal{A}$ is an (α, β, n, m)-active learner for C and $\mathcal{A}$ is (ϵ, δ)-differentially private, where in the definition of privacy we consider the input of $\mathcal{A}$ to be a fully labeled sample $S = (x_i, y_i)_{i=1}^n \in (X \times \{0, 1\})^n$ (and limit the number of labels y_i it can access to m).*

Note that the queries an active learner makes depend on individual data and hence, if exposed, may breach privacy. An example of how such an exposure may occur is a medical research of a new disease – a hospital may posses background information about individuals and hence can access a large pool of unlabeled examples, but to label an example a medical test is needed. Partial information about the labeling queries would hence be leaked to the tested individuals. More information about the queries may be leaked to an observer of the testing site. The following definition remedies this potential breach of privacy.

Definition 5.4. *We define the transcript in an execution of an active learner $\mathcal{A}$ as the ordered sequence $L = (\ell_i)_{i=1}^m \in \{1, 2, \ldots, n\}^m$ of indices that $\mathcal{A}$ outputs throughout the execution. We say that a learner $\mathcal{A}$ is (ϵ, δ)-transcript-differentially private if the algorithm whose input is the labeled sample and whose output is the output of $\mathcal{A}$ together with the transcript of the execution is (ϵ, δ)-differentially private. An algorithm $\mathcal{A}$ is an* $(\alpha, \beta, \epsilon, \delta, n, m)$-TPAL (transcript-private active-learner) *for a concept class C if Algorithm $\mathcal{A}$ is an (α, β, n, m)-Active learner for C and $\mathcal{A}$ is (ϵ, δ)-transcript-differentially private.*

Recall that a semi-supervised learner has no control over which of its examples are labeled, and the indices of the labeled examples are publicly known. Hence, a private semi-supervised learner is, in particular, a transcript-private active learner.

Theorem 5.5. *If $\mathcal{A}$ is an $(\alpha, \beta, \epsilon, \delta, n, m)$-PSSL, then $\mathcal{A}$ is an $(\alpha, \beta, \epsilon, \delta, n, m)$-TPAL.*

In particular, our algorithms from Sections 3 and 4 satisfy Definition 5.4, suggesting that the strong privacy guarantees of Definition 5.4 are achievable. See Appendix C for a discussion about Definition 5.4. The private active learners presented in [4] as well as the algorithm described in the next section only satisfy the weaker Definition 5.3.

5.1 Removing the Dependency on the Privacy Parameters

We next show how to transform a semi-supervised private learner $\mathcal{A}$ into an active learner $\mathcal{B}$ with better privacy guarantees without increasing the labeled sample complexity. Algorithm $\mathcal{B}$, on input an unlabeled database D, randomly chooses a subset of the inputs $D' \subseteq D$ and asks for the labels of the examples in D' (denote the resulting labeled database as S). Algorithm $\mathcal{B}$ then applies $\mathcal{A}$ on D, S. As the next claim states, this eliminates the $\frac{1}{\epsilon}$ factor from the labeled sample complexity as the (perhaps adversarial) choice for the input database is independent of the queries chosen.

Claim 5.6. *If there exists an $(\alpha, \beta, \epsilon^*, \delta, n, m)$-PSSL for a concept class C, then for every ϵ there exists an $\left(\alpha, \beta, \epsilon, \frac{7+e^{\epsilon^*}}{3+e^{2\epsilon^*}}\epsilon\delta, t, m\right)$-PAL (private active learner) for C, where $t = \frac{n}{\epsilon}(3 + \exp(2\epsilon^*))$.*

The proof of Claim 5.6 is similar to the sub-sampling technique for boosting the privacy parameters [21, 5] (see Appendix C for the details). This transformation preserves the efficiency of the base (non-active) learner. Hence, a given (efficient) non-active private learner for a class C could always be transformed into an (efficient) active private learner whose labeled sample complexity does not depend on ϵ. Applying Claim 5.6 to the learner from Theorem 4.2 result in the following theorem, showing that the labeled sample complexity of private active learners has no dependency in the privacy parameters ϵ and δ.

Theorem 5.7. *For every $\alpha, \alpha', \beta, \beta', \epsilon, \delta, n$, if $\mathcal{A}$ is an $(\alpha', \beta', \epsilon'{=}1, \delta, n, n)$-PSSL for a concept class C, then there exists an $(\alpha + \alpha', \beta + \beta', \epsilon, \delta, O(\frac{n}{\epsilon}), m)$-PAL for C where $m = O\left(\frac{1}{\alpha}\operatorname{VC}(C) \cdot \log(\frac{1}{\alpha\beta})\right)$.*

Acknowledgments. We thank Aryeh Kontorovich, Adam Smith, and Salil Vadhan for helpful discussions of ideas in this work.

References

[1] A. Agrawala. Learning with a probabilistic teacher. *Information Theory, IEEE Transactions on*, 16(4):373–379, Jul 1970.

[2] Martin Anthony and John Shawe-Taylor. A result of Vapnik with applications. *Discrete Applied Mathematics*, 47(3):207–217, 1993.

[3] Matin Anthony and Peter L. Bartlett. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press, 2009.

[4] Maria-Florina Balcan and Vitaly Feldman. Statistical active learning algorithms. In *Advances in Neural Information Processing Systems 26*, pages 1295–1303, 2013.

[5] 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.

[6] Amos Beimel, Shiva Prasad Kasiviswanathan, and Kobbi Nissim. Bounds on the sample complexity for private learning and private data release. In *TCC*, volume 5978 of *LNCS*, pages 437–454. Springer, 2010.

[7] Amos Beimel, Kobbi Nissim, and Uri Stemmer. Characterizing the sample complexity of private learners. In Robert D. Kleinberg, editor, *ITCS*, pages 97–110. ACM, 2013.

[8] Amos Beimel, Kobbi Nissim, and Uri Stemmer. Private learning and sanitization: Pure vs. approximate differential privacy. In Prasad Raghavendra, Sofya Raskhodnikova, Klaus Jansen, and José D. P. Rolim, editors, *APPROX-RANDOM*, volume 8096 of *Lecture Notes in Computer Science*, pages 363–378. Springer, 2013.

[9] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: The SuLQ framework. In Chen Li, editor, *PODS*, pages 128–138. ACM, 2005.

[10] Avrim Blum, Katrina Ligett, and Aaron Roth. A learning theory approach to noninteractive database privacy. *J. ACM*, 60(2):12, 2013.

[11] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K. Warmuth. Learnability and the vapnik-chervonenkis dimension. *J. ACM*, 36(4):929–965, 1989.

[12] 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.

[13] Kamalika Chaudhuri and Claire Monteleoni. Privacy-preserving logistic regression. In Daphne Koller, Dale Schuurmans, Yoshua Bengio, and Léon Bottou, editors, *NIPS*. MIT Press, 2008.

[14] Kamalika Chaudhuri, Claire Monteleoni, and Anand D. Sarwate. Differentially private empirical risk minimization. *J. Mach. Learn. Res.*, 12:1069–1109, July 2011.

[15] 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.

[16] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *TCC*, volume 3876 of *Lecture Notes in Computer Science*, pages 265–284. Springer, 2006.

[17] Cynthia Dwork, Guy N. Rothblum, and Salil P. Vadhan. Boosting and differential privacy. In *FOCS*, pages 51–60. IEEE Computer Society, 2010.

[18] 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.

[19] Vitaly Feldman and David Xiao. Sample complexity bounds on differentially private learning via communication complexity. *CoRR*, abs/1402.6278, 2014.

[20] S. Fralick. Learning to recognize patterns without a teacher. *IEEE Trans. Inf. Theor.*, 13(1):57–64, September 2006.

[21] 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.

[22] Michael J. Kearns. Efficient noise-tolerant learning from statistical queries. *J. ACM*, 45(6):983–1006, 1998.

[23] 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.

[24] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *FOCS*, pages 94–103. IEEE Computer Society, 2007.

[25] Benjamin I. P. Rubinstein, Peter L. Bartlett, Ling Huang, and Nina Taft. Learning in a large function space: Privacy-preserving mechanisms for svm learning. *CoRR*, abs/0911.5708, 2009.

[26] N Sauer. On the density of families of sets. *Journal of Combinatorial Theory, Series A*, 13(1):145 – 147, 1972.

[27] III Scudder, H. Probability of error of some adaptive pattern-recognition machines. *Information Theory, IEEE Transactions on*, 11(3):363–371, Jul 1965.

[28] Leslie G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984.

[29] Vladimir Vapnik and Alexey Chervonenkis. Theory of pattern recognition [in russian]. *Nauka, Moscow*, 1974.

[30] Vladimir N. Vapnik and Alexey Y. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

A Some Differentially Private Mechanisms

A.1 The Exponential Mechanism [24]

We next describe the exponential mechanism of McSherry and Talwar [24]. We present its private learning variant; however, it can be used in more general scenarios. The goal here is to chooses a hypothesis $h \in H$ approximately minimizing the empirical error. The choice is probabilistic, where the probability mass that is assigned to each hypothesis decreases exponentially with its empirical error.

Algorithm 3 Exponential Mechanism

Inputs: Privacy parameter ϵ, finite hypothesis class H, and m labeled examples $S = (x_i, y_i)_{i=1}^m$.

1. $\forall h \in H$ define $q(S, h) = |\{i : h(x_i) = y_i\}|$.
2. Randomly choose $h \in H$ with probability $\frac{\exp(\epsilon \cdot q(S,h)/2)}{\sum_{f \in H} \exp(\epsilon \cdot q(S,f)/2)}$.
3. Output h.

Proposition A.1 (Properties of the Exponential Mechanism)**.** *(i) The exponential mechanism is ϵ-differentially private. (ii) Let $\hat{e} \triangleq \min_{f \in H}\{\mathrm{error}_S(f)\}$. For every $\Delta > 0$, the probability that the exponential mechanism outputs a hypothesis h such that $\mathrm{error}_S(h) > \hat{e} + \Delta$ is at most $|H| \cdot \exp(-\epsilon \Delta m/2)$.*

A.2 Data Sanitization

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

For a concept $c : X \to \{0, 1\}$ define $Q_c : X^* \to [0, 1]$ as $Q_c(S) = \frac{1}{|S|} \cdot \left|\{i \ : \ c(x_i) = 1\}\right|$. That is, $Q_c(S)$ is the fraction of the entries in S that satisfy c. A sanitizer for a concept class C is a differentially private algorithm that given a database S outputs a database $\hat{S}$ s.t. $Q_c(S) \approx Q_c(\hat{S})$ for every $c \in C$.

Definition A.2 (Sanitization [10])**.** *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 another database $\hat{S} \in X^*$. Algorithm $\mathcal{A}$ is an $(\alpha, \beta, \epsilon, \delta, m)$-sanitizer for predicates in the class C, if*

1. *$\mathcal{A}$ is (ϵ, δ)-differentially private;*
2. *For every input $S \in X^m$, the size of the database m satisfies:*

$$\Pr_{\mathcal{A}}\left[\exists c \in C \text{ s.t. } |Q_c(S) - Q_c(\hat{S})| > \alpha\right] \leq \beta.$$

The probability is over the coin tosses of algorithm $\mathcal{A}$. As before, when δ=0 (pure privacy) we omit it from the set of parameters.

Theorem A.3 (Blum et al. [10])**.** *For any class of predicates C over a domain X, and any parameters α, β, ϵ, there exists an $(\alpha, \beta, \epsilon, m)$-sanitizer for C, provided that the size of the database m satisfies:*

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

The returned sanitized database contains $O(\frac{\mathrm{VC}(C)}{\alpha^2}\log(\frac{1}{\alpha}))$ *elements.*

B The Vapnik-Chervonenkis Dimension

The Vapnik-Chervonenkis (VC) Dimension is a combinatorial measure of concept classes that characterizes the sample size of PAC learners. 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 on B that are realized by C is $\Pi_C(B) = \Big\{(c(b_1), \ldots, c(b_\ell)) : c \in C\Big\}$. A set $B \subseteq X$ is *shattered* by C if C realizes all possible dichotomies over B, i.e., $\Pi_C(B) = \{0,1\}^{|B|}$.

Definition B.1 (VC-Dimension [30])**.** *The* $\mathrm{VC}(C)$ *is the cardinality of the largest set* $B \subseteq X$ *shattered by* C*. If arbitrarily large finite sets can be shattered by* C*, then* $\mathrm{VC}(C) = \infty$*.*

Sauer's lemma bounds the cardinality of $\Pi_C(B)$ in termes of $\mathrm{VC}(C)$ and $|B|$.

Theorem B.2 ([26])**.** *Let* C *be a concept class over a domain* X*, and let* $B \subseteq X$ *such that* $|B| > \mathrm{VC}(C)$*. It holds that* $\Pi_C(B) \leq \left(\frac{e|B|}{\mathrm{VC}(C)}\right)^{\mathrm{VC}(C)}$*.*

B.1 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 B.3 ([18])**.** *For any* $(\alpha, \beta{<}\frac{1}{2}, n, m)$*-SSL for a class* C *it holds that* $m \geq \frac{\mathrm{VC}(C)-1}{16\alpha}$*.*

Theorem B.4 (VC-Dimension Generalization Bound [30, 11])**.** *Let* C *and* $\mathcal{D}$ *be a concept class and a distribution over a domain* X*. Let* $\alpha, \beta > 0$*, and* $m \geq \frac{8}{\alpha}(\mathrm{VC}(C)\ln(\frac{16}{\alpha}) + \ln(\frac{2}{\beta}))$*. Fix a concept* $c \in C$*, and suppose that we draw a sample* $S = (x_i, y_i)_{i=1}^m$*, where* x_i *are drawn i.i.d. from* $\mathcal{D}$ *and* $y_i = c(x_i)$*. Then,*

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

Hence, an algorithm that takes a sample of $m = \Omega_{\alpha,\beta}(\mathrm{VC}(C))$ labeled examples and outputs a concept $h \in C$ that agrees with the sample is a PAC learner for C. The following is a simple generalization of Theorem B.4.

Theorem B.5 (VC-Dimension Generalization Bound)**.** *Let* C *and* μ *be a concept class and a distribution over a domain* X*. Let* $\alpha, \beta > 0$*, and* $m \geq \frac{48}{\alpha}\left(10\,\mathrm{VC}(C)\log(\frac{48e}{\alpha}) + \log(\frac{5}{\beta}))\right)$*. Suppose that we draw a sample* $S = (x_i)_{i=1}^m$*, where each* x_i *is drawn i.i.d. from* μ*. Then,*

$$\Pr[\exists c, h \in C \text{ s.t. } \mathrm{error}_{\mathcal{D}}(c,h) \geq \alpha \ \wedge \ \mathrm{error}_S(c,h) \leq \alpha/10] \leq \beta.$$

The above theorem generalizes Theorem B.4 in two aspects. First, it holds simultaneously for every pair $c, h \in C$, where as in Theorem B.4 the target concept c is fixed before generating the sample. Second, Theorem B.4 only ensures that a hypothesis h has small generalization error if $\mathrm{error}_S(h) = 0$. In Theorem B.5 on the other hand, this is guaranteed even if $\mathrm{error}_S(h)$ is small (but non-zero).

The next theorem handles (in particular) the agnostic case, in which the concept class C is unknown and the learner uses a hypotheses class H. In particular, given a labeled sample S there may be no $h \in H$ for which $\mathrm{error}_S(h)$ is small.

Theorem B.6 (VC-Dimension Agnostic Generalization Bound [3, 2])**.** *Let H and μ be a concept class and a distribution over a domain X, and let $f : X \to \{0,1\}$ be some concept, not necessarily in H. For a sample $S = (x_i, f(x_i))_{i=1}^m$ where $m \geq \frac{50\,\mathrm{VC}(H)}{\alpha^2}\ln(\frac{1}{\alpha\beta})$ and each x_i is drawn i.i.d. from μ, it holds that*

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

Notice that the sample size in Theorem B.6 is larger than the sample size in Theorem B.5, where, basically, the former is proportional to $\frac{1}{\alpha^2}$ and the latter is proportional to $\frac{1}{\alpha}$.

C Omitted Proofs

C.1 Proving the Correctness of Algorithm GenericLearner

Let C be a concept class over X_d, and let $B = \{b_1, \ldots, b_\ell\} \subseteq X_d$. Recall that the projection of C on B is $\Pi_C(B) = \{\langle c(b_1), \ldots, c(b_\ell)\rangle : c \in C\}$.

Observation 3.2. *For any concept class C over X_d it holds that $\mathrm{VC}(C^\oplus) = O(\mathrm{VC}(C))$.*

Proof. First note that for every $B = \{b_1, \ldots, b_\ell\} \subseteq X_d$

$$\begin{aligned}
\Pi_{C^\oplus}(B) &= \{\langle (h \oplus f)(b_1), \ldots, (h \oplus f)(b_\ell)\rangle : h, f \in C\} \\
&= \{\langle h(b_1), \ldots, h(b_\ell)\rangle \oplus \langle f(b_1), \ldots, f(b_\ell)\rangle : h, f \in C\} \\
&= \{\langle h(b_1), \ldots, h(b_\ell)\rangle : h \in C\} \oplus \{\langle f(b_1), \ldots, f(b_\ell)\rangle : f \in C\} \\
&= \Pi_C(B) \oplus \Pi_C(B).
\end{aligned}$$

Therefore, by Sauer's lemma B.2, $|\Pi_{C^\oplus}(B)| \leq |\Pi_C(B)|^2 \leq \left(\frac{e\ell}{\mathrm{VC}(C)}\right)^{2\,\mathrm{VC}(C)}$. Hence, for $C^\oplus$ to shatter a subset $B \subseteq X_d$ of size ℓ it must be that $\left(\frac{e\ell}{\mathrm{VC}(C)}\right)^{2\,\mathrm{VC}(C)} \geq 2^\ell$. For $\ell \geq 10\,\mathrm{VC}(C)$ this inequality does not hold, and we can conclude that $\mathrm{VC}(C^\oplus) \leq 10\,\mathrm{VC}(C)$. □

Theorem 3.3. *Let C be a concept class over the domain $\{0,1\}^d$. For every α, β, ϵ, there exists an $(\alpha, \beta, \epsilon, \delta{=}0, n, m)$-private semi-supervised proper-learner for C, where $n = O_{\alpha,\beta,\epsilon}(d \cdot \mathrm{VC}(C))$, and $m \geq O\left(\frac{\mathrm{VC}(C)}{\alpha^3\epsilon}\log(\frac{1}{\alpha}) + \frac{1}{\alpha\epsilon}\log(\frac{1}{\beta})\right)$. The learner might not be efficient.*

Proof. Note that $GenericLearner$ only accesses D via a sanitizer, and only accesses S using the exponential mechanism (on Step 5). As each of those two mechanisms is ϵ-differentially private, and as D and S are two disjoint samples, $GenericLearner$ is ϵ-differentially private. We, thus, only need to prove that with high probability the learner returns a good hypothesis.

Fix a target concept $c \in C$ and a distribution μ over X, and define the following three "good" events:

E_1 : For every $h \in C$ it holds that $|\mathrm{error}_S(h) - \mathrm{error}_{\widetilde{D}}(h,c)| \leq \frac{3\alpha}{5}$.

E_2 : The exponential mechanism chooses an $h \in H$ such that $\mathrm{error}_S(h) \leq \frac{\alpha}{5} + \min_{f\in H}\{\mathrm{error}_S(f)\}$.

E_3 : For every $h \in H$ s.t. $\mathrm{error}_S(h) \leq \frac{4\alpha}{5}$, it holds that $\mathrm{error}_\mu(c,h) \leq \alpha$.

We first observe that when these three events happen algorithm $GenericLearner$ returns an α-good hypothesis: For every $(y_1, \ldots, y_\ell) \in \Pi_C(B)$, algorithm $GenericLearner$ adds to H a hypothesis f s.t. $\forall 1 \leq i \leq \ell,\ f(b_i) = y_i$. In particular, H contains a hypothesis h^* s.t. $h^*(x) = c(x)$ for every $x \in B$, that is, a hypothesis h^* s.t. $\mathrm{error}_{\widetilde{D}}(h^*, c) = 0$. As event E_1 has occur we have that this h^* satisfies $\mathrm{error}_S(h^*) \leq \frac{3\alpha}{5}$. Thus, event $E_1 \cap E_2$ ensures that algorithm $GenericLearner$ chooses (using the exponential mechanism) a hypothesis $h \in H$ s.t. $\mathrm{error}_S(h) \leq \frac{4\alpha}{5}$. Event E_3 ensures, therefore, that this h satisfies $\mathrm{error}_\mu(c, h) \leq \alpha$. We will now show $E_1 \cap E_2 \cap E_3$ happens with high probability.

Standard arguments in learning theory state that (w.h.p.) the empirical error on a (large enough) random sample is close to the generalization error (see Theorem B.6). Specifically, by setting n and m to be at least $\frac{1250}{\alpha^2}\,\mathrm{VC}(C)\ln(\frac{25}{\alpha\beta})$, Theorem B.6 ensures that with probability at least $(1 - \frac{2}{5}\beta)$, for every $h \in C$ the following two inequalities hold.

$$|\mathrm{error}_S(h) - \mathrm{error}_\mu(h, c)| \leq \frac{\alpha}{5} \tag{2}$$

$$|\mathrm{error}_D(h, c) - \mathrm{error}_\mu(h, c)| \leq \frac{\alpha}{5} \tag{3}$$

Note that Event E_3 occurs whenever Inequality (2) holds (since $H \subseteq C$). Moreover, by setting the size of the unlabeled database $(n - m)$ to be at least

$$(n-m) \geq O\left(\frac{d \cdot \mathrm{VC}(C^\oplus)\log(\frac{1}{\alpha})}{\alpha^3\epsilon} + \frac{\log(\frac{1}{\beta})}{\epsilon\alpha}\right) = O\left(\frac{d \cdot \mathrm{VC}(C)\log(\frac{1}{\alpha})}{\alpha^3\epsilon} + \frac{\log(\frac{1}{\beta})}{\epsilon\alpha}\right)$$

Theorem A.3 ensures that with probability at least $(1 - \frac{\beta}{5})$ for every $(h \oplus f) \in C^\oplus$ (i.e., for every $h, f \in C$) it holds that

$$\begin{aligned}
\frac{\alpha}{5} &\geq |Q_{(h\oplus f)}(D) - Q_{(h\oplus f)}(\widetilde{D})| \\
&= \left|\frac{1}{|D|}|\{x \in D\ :\ (h \oplus f)(x) = 1\}| - \frac{1}{|\widetilde{D}|}|\{x \in \widetilde{D}\ :\ (h \oplus f)(x) = 1\}|\right| \\
&= \left|\frac{1}{|D|}|\{x \in D\ :\ h(x) \neq f(x)\}| - \frac{1}{|\widetilde{D}|}|\{x \in \widetilde{D}\ :\ h(x) \neq f(x)\}|\right| \\
&= \left|\mathrm{error}_D(h, f) - \mathrm{error}_{\widetilde{D}}(h, f)\right|.
\end{aligned}$$

In particular, for every $h \in C$ it holds that

$$\left|\mathrm{error}_D(h, c) - \mathrm{error}_{\widetilde{D}}(h, c)\right| \leq \frac{\alpha}{5}. \tag{4}$$

Therefore (using Inqeualities (2),(3),(4) and the triangle inequality), Event $E_1 \cap E_3$ occurs with probability at least $(1 - \frac{3\beta}{5})$.

The exponential mechanism ensures that the probability of event E_2 is at least $1 - |H| \cdot \exp(-\epsilon\alpha m/10)$ (see Proposition A.1). Note that $\log|H| \leq |B| \leq |\widetilde{D}| = O\left(\frac{\mathrm{VC}(C)}{\alpha^2}\log(\frac{1}{\alpha})\right)$. Therefore, for $m \geq O\left(\frac{\mathrm{VC}(C)}{\alpha^3\epsilon}\log(\frac{1}{\alpha}) + \frac{1}{\alpha\epsilon}\log(\frac{1}{\beta})\right)$, Event E_2 occurs with probability at least $(1 - \frac{\beta}{5})$.

All in all, by setting $m \geq O\left(\frac{\mathrm{VC}(C)}{\alpha^3\epsilon}\log(\frac{1}{\alpha}) + \frac{1}{\alpha\epsilon}\log(\frac{1}{\beta})\right)$, and $n \geq O\left(\frac{d\cdot\mathrm{VC}(C)\log(\frac{1}{\alpha})}{\alpha^3\epsilon} + \frac{\log(\frac{1}{\beta})}{\epsilon\alpha}\right)$, we ensure that the probability of $GenericLearner$ failing to output an α-good hypothesis is at most β. □

C.2 Proving the Correctness of Algorithm LabelBoost and Algorithm IterLabelBoost

In this section we present the iterative version of algorithm $LabalBoost$, and proove its proerties (stated in Theorem 4.2). Consider algorithm $IterLabelBoost$ (Algorithm 4). Recall that applying algorithm $LabelBoost$ on an algorithm $\mathcal{A}$ reduces the labeled sample complexity (logarithmically). Basically, Algorithm $IterLabelBoost$ simply applies $labelBoost$ iteratively in order to further reduce the labeled sample complexity. Every such application deteriorates the privacy parameters, and hence, every iteration includes a sub-sampling step, which compensates for those privacy losses.

Before analyzing algorithm $IterLabelBoost$ we recall the sub-sampling technique from [21, 5].

Claim C.1 ([21, 5])**.** *Let $\mathcal{A}$ be an (ϵ^*, δ)-differentially private algorithm operating on databases of size n. Fix $\epsilon \leq 1$, and denote $t = \frac{n}{\epsilon}(3 + \exp(\epsilon^*))$. Construct an algorithm $\mathcal{B}$ that on input a database $D = (z_i)_{i=1}^t$ uniformly at random selects a subset $J \subseteq \{1, 2, ..., t\}$ of size n, and runs $\mathcal{A}$ on the multiset $D_J = (z_i)_{i \in J}$. Then, $\mathcal{B}$ is $\left(\epsilon, \frac{4\epsilon}{3+\exp(\epsilon^*)}\delta\right)$-differentially private.*

Remark C.2. *In Claim C.1 we assume that $\mathcal{A}$ treats its input as a multiset. If this is not the case, then algorithm $\mathcal{B}$ should be modified to randomly shuffle the elements in D_J before applying $\mathcal{A}$ on D_j.*

Claim C.1 boosts privacy by selecting random elements from the database and ignoring the rest of the database. The intuition is simple: Fix two neighboring databases D, D' differing (only) on their i^{th} entry. If the i^{th} entry is ignored (which happens with high probability), then the executions on D and on D' are the same (i.e., perfect privacy). Otherwise, (ϵ^*, δ)-privacy is preserved.

In algorithm $IterLabelBoost$ we apply the learner $\mathcal{A}$ on a database containing n i.i.d. samples from the database S (Steps 4). Consider two neighboring databases D, D' differing on their i^{th} entry. Unlike in Claim C.1, the risk is that this entry will appear several times in the database on which $\mathcal{A}$ is executed. As the next claim states, the affects on the privacy guarantees are small. The intuition is that the probability of the i^{th} entry appearing "too many" times is negligible.

Claim C.3. *Let $\epsilon \leq 1$ and $\mathcal{A}$ be an (ϵ, δ)-differentially private algorithm operating on databases of size n. Applying algorithm $\mathcal{B}$ (algorithm 5) on $\mathcal{A}$ results in a $(\ln(244), 2467\delta)$-differentially private algorithm.*

Proof. Fix two neighboring databases $D, D' \in X^n$ differing on their i^{th} entry, and fix a set of outcomes F. Consider an execution of $\mathcal{B}$ on D (and on D'), let V denote the vector chosen on Step 1, and let L denote the number of appearances of i in the vector V.

First note that since D and D' differ in just the i^{th} entry, it holds that $\Pr[\mathcal{B}(D) \in F | L = 0] = \Pr[\mathcal{B}(D') \in F | L = 0]$. Moreover, observe that

$$\Pr[L = \ell] = \frac{\binom{n}{\ell}(n-1)^{n-\ell}}{n^n} \leq \binom{n}{\ell}\left(\frac{1}{n}\right)^\ell \leq \left(\frac{en}{\ell}\right)^\ell \left(\frac{1}{n}\right)^\ell = \left(\frac{e}{\ell}\right)^\ell.$$

Now,

$$\begin{aligned}
\Pr[\mathcal{B}(D) \in F] &= \sum_{\ell=0}^{n} \Pr[L = \ell] \cdot \Pr[\mathcal{B}(D) \in F | L = \ell] \\
&= \sum_{\ell=0}^{n} \Pr[L = \ell] \sum_{v} \Pr[V = v | L = \ell] \cdot \Pr[\mathcal{A}(D_v) \in F].
\end{aligned}$$

Algorithm 4 $IterLabelBoost$

Setting: Algorithm $\mathcal{A}$ with (labeled and unlabeled) sample complexity n.
Input: An unlabeled database $D \in X^{90000n}$ and a labeled database $S \in (X \times \{0,1\})^m$.

1. Set $i = 1$.
2. While $|S| < 300n$:
   % S denotes the currently labeled portion of the database. In each iteration, $|S|$ grows exponentially. The loop ends when S is big enough s.t. we can apply the base learner $\mathcal{A}$ on S.
   (a) Denote $\alpha_i = \frac{\alpha}{24 \cdot 2^i}$, and $\beta_i = \frac{\beta}{4 \cdot 2^i}$.
   (b) Let T be the first $\min\left\{30000n\,,\ \frac{\beta_i}{e}\,\mathrm{VC}(C)\exp(\frac{\alpha_i|S|}{200\,\mathrm{VC}(C)}) - 100|S|\right\}$ elements of D, and remove T from D. Fail if there are not enough elements in D.
      % We consider the input as a one database $(S \circ T \circ D) \in (X \times \{0,1,\perp\})^*$. The functionality of this step can, therefore, be viewed as changing the index in which T ends and D begins.
   (c) Delete (permanently) $\frac{99}{100}|T|$ random entries from T, and $\frac{99}{100}|S|$ random entries from S.
      % Every iteration deteriorates the privacy parameters. We, therefore, boost the privacy guarantees using sub-sampling.
   (d) Initialize $H = \emptyset$.
   (e) Let $P = \{p_1, \ldots, p_\ell\}$ be the set of all points $p \in X$ appearing at least once in T or in S.
   (f) For every $(z_1, \ldots, z_\ell) \in \Pi_C(P) = \{(c(p_1), \ldots, c(p_\ell)) : c \in C\}$, add to H an arbitrary concept $c \in C$ s.t. $c(p_j) = z_j$ for every $1 \leq j \leq \ell$.
   (g) Choose $h \in H$ using the exponential mechanism with privacy parameter ϵ=1, solution set H, and the database S.
   (h) Label T using h, and relabel S using h.
   (i) Add every element of T to S.
   (j) Set $i = i + 1$.
3. Delete $\frac{299}{300}|S|$ random entries from S.
   % Boosting privacy guarantees.
4. Let S' denote the outcome of $|S|$ i.i.d. samples from S.
   % We apply $\mathcal{A}$ on n i.i.d. samples from S. As $\mathcal{A}$ is a learner, it is required to output (w.h.p.) a hypothesis with small error on S. We will show that this suffices to achieve low generalization error (w.h.p.).
5. Execute $\mathcal{A}$ on S'.

Algorithm 5 $\mathcal{B}$

Inputs: Algorithm $\mathcal{A}$ and a database $D = (x_i)_{i=1}^n$.

1. Uniformly at random select $V = (v_1, v_2, \ldots, v_n)$, where each v_i is chosen i.i.d. from $\{1, 2, \ldots, n\}$.
2. Let $D_V = (x_{v_i})_{i=1}^n$.
3. Run $\mathcal{A}$ on D_V.

Let $v \in \{1, \ldots, n\}^n$ be s.t. the number of appearances of i in v is ℓ, and let $w \in (\{1, \ldots, n\} \setminus \{i\})^\ell$. We denote by v_w the vector v where every appearance of i in it is replaced with its corresponding entry from w. For example, if $i = \mathbf{5}$ and $v = (9, 2, \mathbf{5}, 8, \mathbf{5}, 4, 2, \mathbf{5}, 1)$, then for $w = (2, 9, 4)$ we get $v_w = (9, 2, \mathbf{2}, 8, \mathbf{9}, 4, 2, \mathbf{4}, 1)$. Moreover, let W_ℓ denote a random $w \in (\{1, \ldots, n\} \setminus \{i\})^\ell$.

$$\begin{aligned}
&\Pr[\mathcal{B}(D) \in F] \\
&= \sum_{\ell=0}^{n} \Pr[L = \ell] \sum_{v} \Pr[V = v | L = \ell] \sum_{w \in (\{1,\ldots,n\} \setminus \{i\})^\ell} \Pr[W_\ell = w] \cdot \Pr[\mathcal{A}(D_v) \in F] \\
&\leq \sum_{\ell=0}^{n} \Pr[L = \ell] \sum_{v} \Pr[V = v | L = \ell] \sum_{w \in (\{1,\ldots,n\} \setminus \{i\})^\ell} \Pr[W_\ell = w] \cdot \left(e^{\ell\epsilon} \cdot \Pr[\mathcal{A}(D_{v_w}) \in F] + \frac{e^{\ell\epsilon} - 1}{e^\epsilon - 1}\delta \right) \\
&= \sum_{\ell=0}^{n} \Pr[L = \ell] \sum_{v} \frac{1}{\binom{n}{\ell}(n-1)^{n-\ell}} \sum_{w \in (\{1,\ldots,n\} \setminus \{i\})^\ell} \frac{1}{(n-1)^\ell} \cdot \left(e^{\ell\epsilon} \cdot \Pr[\mathcal{A}(D_{v_w}) \in F] + \frac{e^{\ell\epsilon} - 1}{e^\epsilon - 1}\delta \right)
\end{aligned}$$

Note that for every ℓ, every choice for v_w appears in the above sum exactly $\binom{n}{\ell}$ times (as the number of choice for a matching v). Therefore,

$$\begin{aligned}
&\Pr[\mathcal{B}(D) \in F] \\
&\leq \sum_{\ell=0}^{n} \Pr[L = \ell] \sum_{v_w \in (\{1,\ldots,n\} \setminus \{i\})^n} \frac{1}{(n-1)^n} \cdot \left(e^{\ell\epsilon} \cdot \Pr[\mathcal{A}(D_{v_w}) \in F] + \frac{e^{\ell\epsilon} - 1}{e^\epsilon - 1}\delta \right) \\
&= \sum_{\ell=0}^{n} \Pr[L = \ell] \sum_{v_w \in (\{1,\ldots,n\} \setminus \{i\})^n} \Pr[v_w] \cdot \left(e^{\ell\epsilon} \cdot \Pr[\mathcal{A}(D_{v_w}) \in F] + \frac{e^{\ell\epsilon} - 1}{e^\epsilon - 1}\delta \right) \\
&= \sum_{\ell=0}^{n} \Pr[L = \ell] \left(e^{\ell\epsilon} \Pr[\mathcal{B}(D) \in F | L = 0] + \frac{e^{\ell\epsilon} - 1}{e^\epsilon - 1}\delta \right) \\
&= \Pr[\mathcal{B}(D') \in F | L = 0] \sum_{\ell=0}^{n} \Pr[L = \ell] e^{\ell\epsilon} + \sum_{\ell=0}^{n} \Pr[L = \ell] \frac{e^{\ell\epsilon} - 1}{e^\epsilon - 1}\delta.
\end{aligned}$$

Similar arguments show that

$$\Pr[\mathcal{B}(D') \in F] \geq \Pr[\mathcal{B}(D') \in F | L = 0] \sum_{\ell=0}^{n} \Pr[L = \ell] e^{-\ell\epsilon} - \sum_{\ell=0}^{n} \Pr[L = \ell] \ell\delta.$$

Hence,

$$\begin{aligned}\Pr[\mathcal{B}(D)\in F] &\leq \frac{\Pr[\mathcal{B}(D')\in F]+\sum_{\ell=0}^{n}\Pr[L=\ell]\ell\delta}{\sum_{\ell=0}^{n}\Pr[L=\ell]e^{-\ell\epsilon}}\cdot\sum_{\ell=0}^{n}\Pr[L=\ell]e^{\ell\epsilon}+\sum_{\ell=0}^{n}\Pr[L=\ell]\frac{e^{\ell\epsilon}-1}{e^{\epsilon}-1}\delta \\ &\leq \frac{\Pr[\mathcal{B}(D')\in F]+\sum_{\ell=0}^{n}(\frac{e}{\ell})^{\ell}\ell\delta}{1/4}\cdot\sum_{\ell=0}^{n}\left(\frac{e}{\ell}\right)^{\ell}e^{\ell\epsilon}+\sum_{\ell=0}^{n}\left(\frac{e}{\ell}\right)^{\ell}\frac{e^{\ell\epsilon}-1}{e^{\epsilon}-1}\delta.\end{aligned}$$

For $\epsilon\leq 1$ we get that

$$\begin{aligned}\Pr[\mathcal{B}(D)\in F] &\leq \frac{\Pr[\mathcal{B}(D')\in F]+10\delta}{1/4}\cdot 61+36\delta \\ &= 244\Pr[\mathcal{B}(D')\in F]+2476\delta.\end{aligned}$$

□

We next prove the privacy properties of algorithm $IterLabelBoost$.

Lemma C.4. *If $\mathcal{A}$ is $(1,\delta)$-differentially private, then $IterLabelBoost$ is $(1,41\delta)$-differentially private.*

Proof. We think of the input of $IterLabelBoost$ as one database $B\in(X\times\{0,1,\perp\})^{90000n+m}$. Note that the number of iterations performed on neighboring databases is identical (determined by the parameters α,β,n,m), and denote this number as N. Throughout the execution, random elements from the input database are deleted (on Step 2c). Note however, that the size of the database at any moment throughout the execution does not depend on the database content (determined by the parameters α,β,n,m). We denote the size of the database at the beginning of the i^{th} iteration as $n(i)$, e.g., $n(1)=90000n+m$.

Let $\mathcal{L}_t$ denote an algorithm similar to $IterLabelBoost$, except that only the last t iterations are performed. The input of $\mathcal{L}_t$ is a database in $(X\times\{0,1,\perp\})^{n(N-t+1)}$. We next show (by induction on t) that $\mathcal{L}_t$ is $(1,41\delta)$-differentially private. To this end, note that an execution of $\mathcal{L}_0$ consists sub-sampling (as in Claim C.1), i.i.d. sampling (as in Claim C.3), and applying the $(1,\delta)$-private algorithm $\mathcal{A}$. By Claim C.3, steps 4–5 preserve $(\ln(244),2476)$-differential privacy, and, hence, by Claim C.1, we have that $\mathcal{L}_0$ is $(1,41\delta)$-differentially private.

Assume that $\mathcal{L}_{t-1}$ is $(1,41\delta)$-differentially private, and observe that $\mathcal{L}_t$ could be restated as an algorithm that first performs one iteration of algorithm $IterLabelBoost$ and then applies $\mathcal{L}_{t-1}$ on the databases D,S as they are at the end of that iteration. Now fix two neighboring databases B_1,B_2 and consider the execution of $\mathcal{L}_t$ on B_1 and on B_2.

Let S_1^b,T_1^b,D_1^b and S_2^b,T_2^b,D_2^b be the databases S,T,D after Step 2b of the first iteration of $\mathcal{L}_t$ on B_1 and on B_2 (note that $B_1=S_1^b\circ T_1^b\circ D_1^b$ and $B_2=S_2^b\circ T_2^b\circ D_2^b$). If B_1 and B_2 differ (only) on their last portion, denoted as D_1^b,D_2^b, then the execution of $\mathcal{L}_t$ on these neighboring inputs differs only in the execution of $\mathcal{L}_{t-1}$, and hence Inequality (1) (approximate differential privacy) follows from the privacy of $\mathcal{L}_{t-1}$. We, therefore, assume that $D_1^b=D_2^b$ (and that $S_1^b\circ T_1^b$ and $S_2^b\circ T_2^b$ differ in at most one entry). Now, note that $\mathcal{L}_t$ begins by subsampling $\frac{1}{100}$ fraction of the elements in S_1^b and in T_1^b (or in S_2^b and in T_2^b). Denote the resulting databases (i.e., the databases S,T as they are after Step 2c) as S_1^c,T_1^c (and as S_2^c,T_2^c). Let $\mathcal{L}_t'$ denote an algorithm similar to $\mathcal{L}_t$ except without Step 2c ($\mathcal{L}_t'$ operates on partially labeled databases of size $|S_1^c|+|T_2^c|+|D_1^b|$). By Claim C.1, it suffices to show that $\mathcal{L}_t'$ is $(4,590\delta)$-differentially private. Finally, note that $\mathcal{L}_t'$ is identical to $LabelBoost$ with $\mathcal{L}_{t-1}$ as the base learner. As $\mathcal{L}_{t-1}$ is $(1,41\delta)$-differentially private, Lemme 4.1 states that $\mathcal{L}_t'$ is $(4,446\delta)$-differentially private. □

Before proceeding with the utility analysis, we introduce to following notations.

Notation. Consider the i^{th} iteration of $IterLabelBoost$. We let S_i and T_i denote the elements S, T as they are after Step 2c, and let h_i denote the the hypothesis h chosen on Step 2g.

Claim C.5. *With probability at least* $(1-\beta_i)$ *we have that* $\text{error}_{S_i}(h_i) \leq \alpha_i$.

Proof. Let H_i, P_i denote the elements H, P as they are in the i^{th} iteration, and note that by Sauer's lemma,

$$\begin{aligned}
|H_i| &= |\Pi_C(P_i)| \\
&\leq \left(\frac{e|P_i|}{\text{VC}(C)}\right)^{\text{VC}(C)} \\
&\leq \left(\frac{e(|T_i|+|S_i|)}{\text{VC}(C)}\right)^{\text{VC}(C)} \\
&= \left(\frac{e\frac{\beta_i}{100e}\,\text{VC}(C)\exp(\frac{\alpha_i|S_i|}{2\,\text{VC}(C)})}{\text{VC}(C)}\right)^{\text{VC}(C)} \\
&= \left(\frac{\beta_i}{100}\exp(\frac{\alpha_i|S_i|}{2\,\text{VC}(C)})\right)^{\text{VC}(C)} \\
&\leq \beta_i \exp(\frac{\alpha_i|S_i|}{2}).
\end{aligned}$$

For every $(z_1,\ldots,z_\ell) \in \Pi_C(P_i)$, algorithm $IterLabelBoost$ adds to H_i a hypothesis f s.t. $\forall 1 \leq j \leq \ell,\ f(p_j) = z_j$. In particular, H_i contains a hypothesis f^* s.t. $\text{error}_{S_i}(f^*) = 0$. Hence, Proposition A.1 (properties of the exponential mechanism) ensures that the probability of the exponential mechanism choosing an h_i s.t. $\text{error}_{S_i}(h_i) > \alpha_i$ is at most

$$|H_i| \cdot \exp(-\frac{\alpha_i|S_i|}{2}) \leq \beta_i.$$

□

Claim C.6. *Let* $IterLabelBoost$ *be executed with a base learner with sample complexity* n*, and on databases* D, S*. If* $|D| \geq 90000n$*, then* $IterLabelBoost$ *never fails on Step 2b.*

Proof. Let T_i^b denote the database T as it is after Step 2b of the i^{th} iteration, and denote the number of iterations throughout the execution as N. We need to show that $\sum_{i=1}^{N} T_i^b \leq 90000n$. Clearly, $|T_N^b|, |T_{N-1}^b| \leq 30000n$. Moreover, for every $1 < i < N$ we have that $|T_i^b| \geq 2|T_{i-1}^b|$. Hence,

$$\sum_{i=1}^{N} T_i^b \leq 30000n + 30000n \sum_{i=0}^{\infty} \frac{1}{2^i} = 90000n.$$

□

Notation. We use $\exp^{[i]}(\cdot)$ to denote the outcome of i repeated applications of the function $\exp(\cdot)$. For example, $\exp^{[0]}(3) = 3$, and $\exp^{[2]}(3) = e^{e^3}$.

Claim C.7. *Fix* α, β*. Let* $IterLabelBoost$ *be executed on a base learner with sample complexity* n*, and on databases* D, S*, where* $|D| \geq 90000n$ *and* $|S| \geq \frac{576000}{\alpha}\,\text{VC}(C)\log(\frac{768}{\alpha\beta})$*. In every iteration* i

$$|S_i| \geq \frac{40}{\alpha_i}\,\text{VC}(C)\log(\frac{2}{\alpha_i\beta_i})\exp^{[i-1]}(3).$$

Proof. The proof is by induction on i. Note that the base case (for $i = 1$) trivially holds, and assume that the claim holds for $i - 1$. We have that

$$\begin{aligned}
|S_i| &= \frac{1}{100}(|S_{i-1}| + |T_{i-1}|) \\
&= \frac{1}{10000}\frac{\beta_{i-1}}{e}\operatorname{VC}(C)\exp(\frac{\alpha_{i-1}|S_{i-1}|}{2\operatorname{VC}(C)}) \\
&\geq \frac{1}{10000}\frac{\beta_{i-1}}{e}\operatorname{VC}(C)\exp(20\log(\frac{2}{\alpha_{i-1}\beta_{i-1}})\exp^{[i-2]}(3)) \\
&\geq \frac{1}{10000}\frac{\beta_{i-1}}{e}\operatorname{VC}(C)(\frac{2}{\alpha_{i-1}\beta_{i-1}})^{20\exp^{[i-2]}(3)} \\
&\geq \frac{1}{10000}\frac{\beta_{i-1}}{e}\operatorname{VC}(C)(\frac{2}{\alpha_{i-1}\beta_{i-1}})^{18\exp^{[i-2]}(3)}\exp^{[i-1]}(3) \\
&\geq \frac{1}{10000}\frac{\beta_{i-1}}{e}\operatorname{VC}(C)(\frac{2}{\alpha_{i-1}\beta_{i-1}})^{54}\exp^{[i-1]}(3) \\
&\geq \frac{40}{\alpha_i}\operatorname{VC}(C)\log(\frac{2}{\alpha_i\beta_i})\exp^{[i-1]}(3).
\end{aligned}$$

□

Recall that the loop of Step 2 in Algorithm $IterLabelBoost$ stops when $S \geq 300n$. Hence, the above claim implies the following corollary.

Corollary C.8. *There are at most $O(\log^* n)$ iterations throughout the execution of $IterLabelBoost$ on a base learner $\mathcal{A}$ with sample complexity n.*

Claim C.9. *Let $IterLabelBoost$ be executed on databases D, S containing i.i.d. samples from a fixed distribution μ, where the examples in S are labeled by some fixed target concept $c \in C$, and $|S| \geq \frac{576000}{\alpha}\operatorname{VC}(C)\log(\frac{768}{\alpha\beta})$. The probability that $\operatorname{error}_\mu(c, h_i) > 12\sum_{j=1}^{i}\alpha_j$ is at most $2\sum_{j=1}^{i}\beta_j$.*

Proof. The proof is by induction on i. Note that for $i = 1$ we have that S_1 contains $\frac{120}{\alpha_1}\operatorname{VC}(C)\log(\frac{2}{\alpha_1\beta_1})$ i.i.d. samples from μ that are labeled by the target concept c. By Claim C.5, with probability at least $(1 - \beta_1)$, we have that $\operatorname{error}_{S_1}(h_1) \leq \alpha_1$. In that case, Theorem B.5 (the VC dimension bound) states that with probability at least $(1 - \beta_1)$ it holds that $\operatorname{error}_\mu(c, h_1) \leq 12\alpha_1$.

Now assume that the claim holds for $(i - 1)$, and consider the i^{th} iteration. Note that S_i contains i.i.d. samples from μ that are labeled by h_{i-1}. Moreover, by Claim C.7, we have that $|S_i| \geq \frac{40}{\alpha_i}\operatorname{VC}(C)\log(\frac{2}{\alpha_i\beta_i})$. By Claim C.5, with probability at least $(1 - \beta_i)$, we have that $\operatorname{error}_{S_i}(h_i) \leq \alpha_i$. If that is the case, Theorem B.5 states that with probability at least $(1 - \beta_i)$ it holds that $\operatorname{error}_\mu(h_{i-1}, h_i) \leq 12\alpha_i$. So, with probability at least $(1 - 2\beta_i)$ we have that $\operatorname{error}_\mu(h_{i-1}, h_i) \leq 12\alpha_i$. Using the inductive assumption, the probability that $\operatorname{error}_\mu(c, h_i) \leq \operatorname{error}_\mu(c, h_{i-1}) + \operatorname{error}_\mu(h_{i-1}, h_i) \leq 12\sum_{j=1}^{i}\alpha_j$ is at least $(1 - 2\sum_{j=1}^{i}\beta_j)$. □

Lemma C.10. *Fix α, β. Applying $IterLabelBoost$ on an (α', β', n, n)-SSL for a class C results in an $(\alpha + \alpha', \beta + \beta', O(n), m)$-SSL for C, where $m = O(\frac{1}{\alpha}\operatorname{VC}(C)\log(\frac{1}{\alpha\beta}))$.*

Proof. Let $IterLabelBoost$ be executed on databases D, S containing i.i.d. samples from a fixed distribution μ, where $|D| \geq 90000n$ and $|S| \geq \frac{576000}{\alpha}\operatorname{VC}(C)\log(\frac{768}{\alpha\beta})$. Moreover, assume that the examples in S are labeled by some fixed target concept $c \in C$.

Consider the last iteration of Algorithm $IterLabelBoost$ (say $i = N$) on these inputs. The intuition is that after the last iteration, when reaching Step 4, the database S is big enough s.t. $\mathcal{A}$ returns (w.h.p.) a hypothesis with small error on S. This hypothesis also has small generalization error as S is labeled by h_N which is close to the target concept (by Claim C.9).

Formally, let S^3 denote the database S as it after Step 3 of the execution, and let h_{fin} denote the hypothesis returned by the base learner $\mathcal{A}$ on Step 5. By the while condition on Step 2, we have that $|S^3| \geq n$. Hence, by the utility guarantees of the base learner $\mathcal{A}$, with probability at least $(1 - \beta')$ we have that $\text{error}_{S^3}(h_{\text{fin}}) \leq \alpha'$. As $|S^3| \geq \frac{1}{300}|S| \geq \frac{1920}{\alpha}\text{VC}(C)\log(\frac{768}{\alpha\beta})$, and as S^3 contains i.i.d. samples from μ labeled by h_N, Theorem B.5 states that with probability at least $(1 - \frac{\beta}{2})$ it holds that $\text{error}_\mu(h_{\text{fin}}, h_N) \leq \frac{\alpha}{2}$. By Claim C.9, with probability at least $(1 - 2\sum_{i=1}^{N}\beta_i) \geq (1 - \frac{\beta}{2})$ it holds that $\text{error}_\mu(c, h_N) \leq 12\sum_{j=1}^{i}\alpha_i \leq \frac{\alpha}{2}$. All in all (using the triangle inequality), with probability at least $(1 - \beta' - \beta)$ we get that $\text{error}_\mu(c, h_{\text{fin}}) \leq \alpha' + \alpha$. □

Combining Lemma C.4 and Lemma C.10 we get the following theorem.

Theorem C.11. *Fix α, β. Applying $IterLabelBoost$ on an $(\alpha', \beta', \epsilon'{=}1, \delta, n, n)$-PSSL for a class C results in an $(\alpha + \alpha', \beta + \beta', \epsilon'{=}1, 41\delta, O(n), m)$-PSSL for C, where $m = O(\frac{1}{\alpha}\text{VC}(C)\log(\frac{1}{\alpha\beta}))$.*

Using Claim C.1 to boost the privacy guarantees of the learner resulting from Theorem C.11, proves Theorem 4.2:

Theorem 4.2 *Fix α, β, ϵ. Applying $IterLabelBoost$ on an $(\alpha', \beta', \epsilon'{=}1, \delta, n, n)$-PSSL for a class C results in an $(\alpha + \alpha', \beta + \beta', \epsilon, \delta, O(\frac{n}{\epsilon}), m)$-PSSL for C, where $m = O(\frac{1}{\alpha\epsilon}\text{VC}(C)\log(\frac{1}{\alpha\beta}))$.*

C.3 A Lower Bound for Transcript-Private Active-Learners

Recall Definition 5.4 from Section 5.

Definition 5.4 *We define the transcript in an execution of an active learner $\mathcal{A}$ as the ordered sequence $L = (\ell_i)_{i=1}^m \in \{1, 2, \ldots, n\}^m$ of indices that $\mathcal{A}$ outputs throughout the execution. We say that a learner $\mathcal{A}$ is (ϵ, δ)-transcript-differentially private if the algorithm whose input is the labeled sample and whose output is the output of $\mathcal{A}$ together with the transcript of the execution is (ϵ, δ)-differentially private. An algorithm $\mathcal{A}$ is an* $(\alpha, \beta, \epsilon, \delta, n, m)$-TPAL (transcript-private active-learner) *for a concept class C if Algorithm $\mathcal{A}$ is an (α, β, n, m)-Active learner for C and $\mathcal{A}$ is (ϵ, δ)-transcript-differentially private.*

As mentioned in Section 5, our algorithms from Sections 3 and 4 satisfy Definition 5.4, suggesting that the strong privacy guarantees of Definition 5.4 are achievable. However, as we will now see, this comes with a price. The work on (non-private) active learning has mainly focused on reducing the dependency of the labeled sample complexity in α (the approximation parameter). The classic result in this regime states that the labeled sample complexity of learning $\texttt{THRESH}_d$ is $O(\log(\frac{1}{\alpha}))$, exhibiting an exponential improvement over the $\Omega(\frac{1}{\alpha})$ labeled sample complexity in the non-active model. As the next theorem states the labeled sample complexity of every transcript-private active-learner for $\texttt{THRESH}_d$ is lower bounded by $\Omega(\frac{1}{\alpha})$.

Theorem C.12. *For every $(\alpha, \beta, \epsilon, \delta, n, m)$-TPAL for $\texttt{THRESH}_d$ it holds that $m = \Omega\left(\frac{1}{\alpha}\right)$.*

Proof. Let $\mathcal{A}$ be an $(\alpha, \beta, \epsilon, \delta, n, m)$-TPAL for $\texttt{THRESH}_d$. Without loss of generality, we can assume that $n \geq \frac{100}{\alpha^2}\ln(\frac{1}{\alpha\beta})$ (since $\mathcal{A}$ can ignore part of the sample). Denote $B = \{0, 1, \ldots, 8\alpha n\}$, and consider the following thought experiment for randomly generating a labeled sample of size n.

1. Let $D = (x_1, x_2, \ldots, x_n)$ denote the outcome of n uniform iid draws from X_d.
2. Uniformly at random choose $t \in B$, and let $c_t \in \texttt{THRESH}_d$ be s.t. $c_t(x) = 1$ iff $x < t$.
3. Return $S = (x_i, c_t(x_i))_{i=1}^n$.

The above process induces a distribution on labeled samples of size n, denoted as $\mathcal{P}$. Let $S \sim \mathcal{P}$, and consider the execution of $\mathcal{A}$ on S. Recall that $\mathcal{A}$ operates on the unlabeled portion of S and actively queries for labels.

We first show that $\mathcal{A}$ must (w.h.p.) ask for the label of at least one example in B. To this end, note that even given the labels of all $x \notin B$, the target concept is distributed uniformly on B, and the probability that $\mathcal{A}$ fails to output an α-good hypothesis is at least $\frac{3}{4}$. Hence,

$$\begin{aligned}
\beta &\geq \Pr_{S,\mathcal{A}}[\mathcal{A} \text{ fails}] \\
&\geq \Pr_{S,\mathcal{A}}\left[\begin{array}{c}\mathcal{A} \text{ does not ask for the label} \\ \text{of any point in } B \text{ and fails}\end{array}\right] \\
&= \Pr_{S,\mathcal{A}}\left[\begin{array}{c}\mathcal{A} \text{ does not ask for the} \\ \text{label of any point in } B\end{array}\right] \cdot \Pr_{S,\mathcal{A}}\left[\mathcal{A} \text{ fails} \,\middle|\, \begin{array}{c}\mathcal{A} \text{ does not ask for the} \\ \text{label of any point in } B\end{array}\right] \\
&\geq \Pr_{S,\mathcal{A}}\left[\begin{array}{c}\mathcal{A} \text{ does not ask for the} \\ \text{label of any point in } B\end{array}\right] \cdot \frac{3}{4}
\end{aligned}$$

Thus, $\mathcal{A}$ asks for the label of a point in B with probability at least $(1 - \frac{4}{3}\beta)$.

Let b denote the the number of elements from B in the database S. Standard arguments in learning theory (see Theorem B.6) state that with all but β probability it holds that $7\alpha n \leq b \leq 9\alpha n$. We continue with the proof assuming that this is the case. Now choose a random x^* from S s.t. $x^* \in B$. Note that

$$\begin{aligned}
\Pr_{S,x^*,\mathcal{A}}[\mathcal{A}(S) \text{ asks for the label of } x^*] &\geq \Pr_S[b \leq 9\alpha n] \cdot \Pr_{S,x^*,\mathcal{A}}\left[\mathcal{A}(S) \text{ asks for the label of } x^* \middle| b \leq 9\alpha n\right] \\
&\geq (1-\beta) \cdot \frac{(1 - \frac{4}{3}\beta)}{9\alpha n} \\
&\geq \frac{1 - \frac{7}{3}\beta}{9\alpha n}.
\end{aligned}$$

Choose a random $\hat{x}$ from S (uniformly), and construct a labeled sample S' by swapping the entries $(x^*, c(x^*))$ and $(\hat{x}, c(\hat{x}))$ in S. Note that S' is also distributed according to $\mathcal{P}$, and that $\hat{x}$ is a uniformly random element of S'. Therefore,

$$\Pr_{S,x^*,\hat{x},\mathcal{A}}\left[\mathcal{A}(S') \text{ asks for the label of } \hat{x}\right] \leq \frac{m}{n}.$$

As S and S' differ in at most 2 entries, differential privacy states that

$$\begin{aligned}
\frac{m}{n} &\geq \Pr_{S,x^*,\hat{x},\mathcal{A}}\left[\mathcal{A}(S') \text{ asks for the label of } \hat{x}\right] \\
&= \sum_{S,x^*,\hat{x}} \Pr[S,x^*,\hat{x}] \cdot \Pr_{\mathcal{A}}\left[\mathcal{A}(S') \text{ asks for the label of } \hat{x}\right] \\
&\geq \sum_{S,x^*,\hat{x}} \Pr[S,x^*,\hat{x}] \cdot e^{-2\epsilon} \cdot \Pr_{\mathcal{A}}\left[\mathcal{A}(S) \text{ asks for the label of } x^*\right] - \delta(1+e^{-\epsilon}) \\
&= e^{-2\epsilon} \cdot \Pr_{S,x^*,\mathcal{A}}\left[\mathcal{A}(S) \text{ asks for the label of } x^*\right] - \delta(1+e^{-\epsilon}) \\
&\geq e^{-2\epsilon} \cdot \frac{1-\frac{7}{3}\beta}{9\alpha n} - \delta(1+e^{-\epsilon}).
\end{aligned}$$

Solving for m, this yields $m = \Omega(\frac{1}{\alpha})$.

□

C.4 Removing the Dependency on the Privacy Parameters

Claim 5.6. *If there exists an $(\alpha,\beta,\epsilon^*,\delta,n,m)$-PSSL for a concept class C, then for every ϵ there exists an $\left(\alpha,\beta,\epsilon,\frac{7+e^{\epsilon^*}}{3+e^{2\epsilon^*}}\epsilon\delta,t,m\right)$-PAL (private active learner) for C ,where $t=\frac{n}{\epsilon}(3+\exp(2\epsilon^*))$.*

Algorithm 6 $SubSampling$

Inputs: Base learner $\mathcal{A}$, privacy parameters ϵ^*,ϵ, and a database $D=(x_i)_{i=1}^t$ of t unlabeled examples.

1. Uniformly at random select a subset $J \subseteq \{1,2,...,t\}$ of size n, and let $K \subseteq J$ denote the smallest m indices in J.
2. Request the label of every index $i \in K$, and let $\{y_i \ : \ i \in K\}$ denote the received answers.
3. Run $\mathcal{A}$ an the multiset $D_J = \{(x_i,\perp) : i \in J \setminus K\} \cup \{(x_i,y_i) : i \in K\}$.

Proof. The proof is via the construction of Algorithm $SubSampling$ (Algorithm 6). The utility analysis is straight forward. Fix a target concept c and a distribution μ. Assume that D contains t i.i.d. samples from μ and that every query on an index i is answered by $c(x_i)$. Therefore, algorithm $\mathcal{A}$ is executed on a multiset D_J containing n i.i.d. samples from μ where m of those samples are labeled by c. By the utility properties of $\mathcal{A}$, an α-good hypothesis is returned with probability at least $(1-\beta)$.

For the privacy analysis, fix two neighboring databases $S,S' \in (X \times \{0,1\})^t$ differing on their i^{th} entry, and let $D,D' \in X^t$ denote the restriction of those two databases to X (that is, D contains an entry x for every entry (x,y) in S). Consider an execution of $SubSampling$ on D (and on D'), and let $J \subseteq \{1,\ldots,n\}$ denote the random subset of size n chosen on Step 1. Moreover, and let D_J denote the multiset on which $\mathcal{A}$ in executed.

Since S and S' differ in just the i^{th} entry, for any set of outcomes F it holds that $\Pr[\mathcal{A}(D_J) \in F | i \notin J] = \Pr[\mathcal{A}(D'_J) \in F | i \notin J]$. When $i \in J$ we have that

$$\Pr[SubSampling(D) \in F \wedge i \in J] = \sum_{\substack{R \subseteq [t]\setminus\{i\} \\ |R|=n-1}} \Pr[J = R \cup \{i\}] \cdot \Pr[\mathcal{A}(D_J) \in F | J = R \cup \{i\}].$$

Note that for every choice of $R \subseteq [t] \setminus \{i\}$ s.t. $|R| = (n-1)$, there are exactly $(t-n)$ choices for $Q \subseteq [t] \setminus \{i\}$ s.t. $|Q| = n$ and $R \subseteq Q$. Hence,

$$\begin{aligned}
&\Pr[SubSampling(D) \in F \wedge i \in J] \\
&\quad = \sum_{\substack{R \subseteq [t]\setminus\{i\} \\ |R|=n-1}} \frac{1}{t-n} \sum_{\substack{Q \subseteq [t]\setminus\{i\} \\ |Q|=n \\ R \subseteq Q}} \Pr[J = R \cup \{i\}] \cdot \Pr[\mathcal{A}(D_J) \in F | J = R \cup \{i\}] \\
&\quad \leq \sum_{\substack{R \subseteq [t]\setminus\{i\} \\ |R|=n-1}} \frac{1}{t-n} \sum_{\substack{Q \subseteq [t]\setminus\{i\} \\ |Q|=n \\ R \subseteq Q}} \Pr[J = Q] \cdot \left(\mathrm{e}^{2\epsilon^*} \cdot \Pr[\mathcal{A}(D_J) \in F | J = Q] + (1 + e^{\epsilon^*})\delta \right).
\end{aligned}$$

For the last inequality, note that D_Q and $D_{R \cup \{i\}}$ differ in at most two entries, as they differ in one unlabeled example, and possibly one other example that is labeled in one multiset and unlabeled on the other. Now note that every choice of Q will appear in the above sum exactly n times (as the number of choices for appropriate R's s.t. $R \subseteq Q$). Hence,

$$\begin{aligned}
&\Pr\left[\{SubSampling(D) \in F\} \wedge \{i \in J\}\right] \\
&\quad \leq \frac{n}{t-n} \sum_{\substack{Q \subseteq [t]\setminus\{i\} \\ |Q|=n}} \Pr[J = Q] \cdot \left(\mathrm{e}^{2\epsilon^*} \cdot \Pr[\mathcal{A}(D_J) \in F | J = Q] + (1 + e^{\epsilon^*})\delta \right) \\
&\quad = \frac{n}{t-n} \cdot \Pr[i \notin J] \cdot e^{2\epsilon^*} \cdot \Pr[\mathcal{A}(D_J) \in F | i \notin J] + \frac{n}{t-n} \cdot \Pr[i \notin J] \cdot (1 + e^{\epsilon^*})\delta \\
&\quad = \frac{n}{t} e^{2\epsilon^*} \cdot \Pr[\mathcal{A}(D_J) \in F | i \notin J] + \frac{n}{t}(1 + e^{\epsilon^*})\delta \\
&\quad = \frac{n}{t} e^{2\epsilon^*} \cdot \Pr[\mathcal{A}(D'_J) \in F | i \notin J] + \frac{n}{t}(1 + e^{\epsilon^*})\delta.
\end{aligned}$$

Therefore,

$$\begin{aligned}
\Pr[SubSampling(D) \in F] &= \Pr\left[\{SubSampling \in F\} \wedge \{i \in J\}\right] + \Pr[i \notin J] \cdot \Pr[\mathcal{A}(D'_J) \in F | i \notin J] \\
&\leq \left(\frac{n}{t} e^{2\epsilon^*} + \frac{t-n}{t} \right) \cdot \Pr[\mathcal{A}(D'_J) \in F | i \notin J] + \frac{n}{t}(1 + e^{\epsilon^*})\delta.
\end{aligned}$$

Similar arguments show that

$$\Pr[SubSampling(D') \in F] \geq \left(\frac{n}{t} e^{-2\epsilon^*} + \frac{t-n}{t} \right) \cdot \Pr[\mathcal{A}(D'_J) \in F | i \notin J] - \frac{n}{t} 2\delta.$$

For $t \geq \frac{n}{\epsilon}(3 + \exp(2\epsilon^*))$, this yields

$$\Pr[SubSampling(D) \in F] \leq e^{\epsilon} \cdot \Pr[SubSampling(D') \in F] + \frac{7 + e^{\epsilon^*}}{3 + e^{2\epsilon^*}} \epsilon\delta.$$

□