Private Boosted Decision Trees via Smooth Re-Weighting

by

Bahar Salamatian

B.Sc., Sharif University of Technology, 2019

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Summer 2021

Declaration of Committee

Name:	**Bahar Salamatian**
Degree:	**Master of Science**
Thesis title:	**Private Boosted Decision Trees via Smooth Re-Weighting**
Committee:	**Chair:** Evgenia Ternovska Associate Professor, Computing Science
	Igor Shinkar Supervisor Assistant Professor, Computing Science
	Andrei A.Bulatov Committee Member Professor, Computing Science
	Valentine Kabanets Examiner Professor, Computing Science

Abstract

Protecting the privacy of people whose data is used by machine learning algorithms is important. Differential Privacy is the appropriate mathematical framework for formal guarantees of privacy, and boosted decision trees are a popular machine learning technique. So we propose and test a practical algorithm for boosting decision trees that guarantees differential privacy. Privacy is enforced because our booster never puts too much weight on any one example; this ensures that each individual's data never influences a single tree "too much." Experiments show that this boosting algorithm can produce better sparsity and accuracy than other differentially private ensemble classifiers.

Keywords: differential privacy, smooth boosting, decision tree, decision stump

Acknowledgements

I want to thank my patient and supportive supervisor, Professor Igor Shinkar. Thanks for letting me explore different projects to find a project that I am passionate about. I am grateful for all of your guidance and for helping me to become a better researcher. I can not imagine having any better supervisor. I also want to thank my collaborators. This thesis was not possible without them, Marco L. Carmosino, Akbar Rafiey, Mohammadmahdi Jahanara, and Vahid Reza Asadi. I am particularly grateful to Marco, who introduced me to the concept of differential privacy; I learned a lot from you. Finally, I would like to thank my partner, parents, and brother for their unconditional love and support during the completion of this thesis.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

A machine learning algorithm is an algorithm that gets a set S of n datapoints in some high dimensional space $\mathbb{R}^r$, and for each datapoint $x \in S$ it also gets its boolean label $y(x) \in \{\pm 1\}$. In addition, we assume that the elements of S are sampled i.i.d. from some unknown distribution $\mathcal{D}$, such that each point x in the support of $\mathcal{D}$ has some unique Boolean label $y = y(x)$ that is unknown to the algorithm.

The goal of the learning algorithm is given the points in S and their labels to output a functions $h\colon \mathbb{R}^r \to \{\pm 1\}$ from a certain restricted class of functions (called the hypothesis class), so that the hypothesis h outputs the correct label on as many points are possible from the (unknown) distribution $\mathcal{D}$. That is, by reading only the points in S we want to build a hypothesis h that maximizes $\Pr_{x\sim\mathcal{D}}[h(x) = y(x)]$.

One popular and widely used hypothesis class is the class of *Boolean decision trees*. Recently, this notion has been generalized to *Boosted decision trees*. These are defined by a distribution over some finite collection of Boolean decision trees. That is, given an input x the boosted decision tree computes the value of each decision tree on the input x, and decides to output $+1$ or -1 according to the weighted majority of the outputs of the trees.

Boosted decision trees are a popular, widely deployed, and successful machine learning technique. Boosting constructs an ensemble (distribution) of decision trees sequentially, by calling a decision tree *base learner* with sample weights that "concentrate attention" on training examples that are poorly classified by trees constructed so far [30].

Differential Privacy (DP) is a mathematical definition of privacy which ensures that the distribution over hypotheses produced by a learning algorithm does not depend *too much* on any one input example [7], so that an adversary cannot tell if any specific individual participated in a differentially private study or not [see 32, section IV.C.1].

Recent purely theoretical work of [4] used *Smooth Boosting* to give a simple and differentially private algorithm for learning large-margin half-spaces. Informally speaking, smooth boosting is an algorithmic technique that gets a differential private weak learning algorithm (that's is a learning algorithm that works only slightly better than a random guess), and converts it into a learning algorithm while preserving the differentially privacy property.

This is done by applying the weak learning algorithm repeatedly on the same training set with different distributions, making sure that never to concentrate too much sample weight on any one example. The boosting algorithm of [4] is generic, and does not depend on any specific features of the weak learner beyond differential privacy.

1.1 Our contributions

In this work we demonstrate that the smooth boosting algorithm of [4] is a practical and efficient differentially private classifier when paired with decision "stumps" – depth-1 trees, i.e., predicates that depend on only one feature of the dataset[1]. We compare on three classification tasks to DP logistic regression [6], DP bagging [18], DP gradient boosting [22], and smooth boosting over our own "reference implementation" of DP decision trees. In all cases, smooth-boosted decision *stumps* improve on other algorithms in accuracy, model sparsity, or both in the high-privacy regime. This is surprising; in the non-private setting somewhat deeper trees (depth 3 - 7) generally improve accuracy. It seems that stumps better tolerate the amount of noise that must be added to enforce privacy for small samples. Since many applications of DP (e.g., US Census sample surveys, genetic data) require simple and accurate models for small datasets, we regard the high utility of smooth-boosted DP-Stumps in these settings as a *pleasant* surprise.

1.2 Overview of our work

Below we provide an informal overview of our work by describing its algorithmic components, and the results we obtain from our experiments.

As explained above, the theoretical part of this work relies on the notion of *smooth boosting* which we describe next. Smooth boosting consists of two parts: (1) constructing a differentially private weak learning algorithm, and (2) boosting the learning algorithm while preserving privacy. In our case we study two hypothesis classes that the weak learning algorithms output: (i) 1-rules, also known as decision stumps or dictators, and (ii) decision trees of some fixed depth. Boosting of the weak learners output distributions over the basic hypotheses.

Weak learner: Given a training data set $S \in (\mathcal{X} \times \{\pm 1\})^n$ consisting of n data points $x = (x_{i,1}, \ldots, x_{i,r}) \in \mathbb{R}^r$ and their labels $y(x) \in \{\pm 1\}$, our weak learning algorithm constructs a decision tree as a hypothesis, where each node in the tree depends on one of the coordinates of the input x. The tree is constructed iteratively, where in each iteration given the tree so far we decide which of its leaves to split and which predicate (coordinate/feature of the

[1]Such predicates are known as the *dictators* in the literature on analysis of Boolean functions

input) to associate with this node. The decision is made by a greedy heuristic, namely we choose an split (leaf and coordinate pair) that has a high splitting criterion value. We formally define our notion of splitting criterion in Section 4.2 and Definition 4.2.1.

In particular, we study the 1-Rules or "Decision Stumps", which are degenerate trees with only one node. It turns out that these using them as weak learners in the boosting algorithm gives a learning algorithm that improves over DP-RL, which is the standard differentially private learning algorithms in the literature. See Section 4 for details.

In each iteration we use a simple differentially private algorithm named Weighted Exponential Mechanism, which is defined in 2.6.3, to find a split that with high probability has a high splitting criterion value. In order to prove privacy guarantees for our weak learners we use composition theorems for differentially privacy informally described below, and formally in Theorem 2.5.3 and Theorem 2.5.4.

Boosting algorithm: Given a training data set $S \in (\mathcal{X} \times \{\pm 1\})^n$ consisting of n data points $x = (x_{i,1}, \ldots, x_{i,r}) \in \mathbb{R}^r$ and their labels $y(x) \in \{\pm 1\}$, and a weak learning algorithm, we start by applying the learning algorithm on S with the uniform distribution $\mu_1 = U_n$. The learning algorithm returns us a hypothesis $h_1 \colon \mathcal{X} \to \{\pm 1\}$.

In the next iteration, we go over all data points in S, and see how many of them are misclassified by h_1, i.e., find all $x \in \mathcal{X}$ such that $h_1(x) \neq y(x)$, where $y(x)$ is the correct label of x. Then, we define a new distribution μ_2 that gives more weight to the x's misclassified by h_1, and would like to run the weak learning algorithm on this distribution μ_2. However, since we want our learning algorithm to preserve privacy, we need to make sure that μ_2 is κ-smooth for some parameter $\kappa \in [0, 1]$, where, roughly speaking "smooth" means close to uniform. This is achieved by "smoothing" μ_2 using the procedure called *Bregman projection* (Definition 2.2.4). We run the weak learning algorithm on this smooth distribution, and obtain another hypothesis h_2.

We continue in the same fashion to the next iteration, by finding all points that are misclassified by the average of the previous hypotheses $h_1, \ldots h_i$, defining the appropriate distribution, smoothing it using Bregman projection, and then running the weak algorithm again to obtain the next hypothesis h_{i+1}.

The output of the learning algorithm is the sign function of the average of the hypotheses obtained in all the iterations. That is, $\text{sign}(\sum_i h_i(x))$, where $\text{sign}(r) = 1$ for all $r > 0$ and $\text{sign}(r) = -1$ for all $r \leq 0$. See Algorithm 2 and Theorem 3.0.1 in Section 1.3 for details.

Composition theorems: In order to prove that the smooth boosting defined above is indeed differentially private, we use the *sequential composition theorem* stated in Theorem 2.5.3, which we explain next. A composition of two private algorithms M_1 and M_2 is defined as follows: given an input dataset S we first run M_1 of S, and obtain an output. In our case the output is a hypothesis h_1. Then, we run M_2 by providing to it the dataset S as well

as h_1 the output of M_1. Assuming that each of M_1 and M_2 are differentially private with some parameters, the composition theorem provides the guarantees on the privacy of their composition. See Theorem 2.5.3 for details.

Similarly to the sequential composition theorem we also use the *parallel composition theorem*. For parallel composition we partition all data points (from both training dataset and testing dataset) into k parts $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2 \cup \cdots \cup \mathcal{D}_k$ according to some predefined partition function. (For example, the partition function can be a decision tree of height $\log_2(k)$ where each node depends on some coordinate/feature of the datapoint.) For each $i = 1, \ldots, k$ we have a differentially private learning algorithm M_i. The parallel composition of these M_i is obtained in the natural way: (1) Run each M_i independently and obtain a hypothesis $h_i \colon \mathcal{D}_i \to \{\pm 1\}$. (2) Then, given an input $x \in \mathcal{D}$ we first find i^* such that $x \in \mathcal{D}_{i^*}$, and then output $h_{i^*}(x)$. The parallel composition theorem states that the differential privacy of the composition is small assuming that each of the M_i's is differentially private with the appropriate privacy guarantee. See Theorem 2.5.4 for the precise statement.

Experiments and the quality our results: We implement our boosted learning algorithm, where hypothesis class of the weak learning algorithms output consists of 1-rules, also known as decision stumps or dictators. For these algorithm we show that smooth boosting of 1-rules yield improved model accuracy under privacy constraints when compared to the standard learning algorithms, such as differentially private logistic regression (DP-LR).

Specifically, run our experiments on three sets of datapoints described below. We explain

Adult This dataset contains the information from Census data, containing some statistical information about the participants in the survey, such as age range, marital status, languages the person speaks, etc. For this data set the goal is to build a hypothesis that predicts whether the income of a given person is above \$50K/yr based on census data. Here the dataset is publicly available, and has 32,561 training examples, that are standard and are used by all learning algorithms. Then the hypothesis is tested on additional 16,282 test examples. These test examples provided by the Census data are also standard, and are not to be used during when running the learning algorithm.

Cod-RNA This dataset is available from the LIBSVM website, and has 59,535 training examples. The goal of the learning algorithm here is to build a hypothesis for detection of non-coding RNA. Unlike in the Adult dataset, here we do not have the standard separation between training data and testing data. In order to test out hypothesis we use cross-valudation (Definition 2.3.5). In the cross-validation setting we choose, say 10 percent of random datapoint from the data set $T \subset \mathcal{X}$, and use the remaining points for training. Then we test the obtained hypothesis on the chosen 10 percent of the data points T, and measure on how many of these points the hypothesis provides a correct label.

This procedure is repeated several times, each time sampling an new set T of testing datapoints.

Mushrooms This dataset has 8124 training examples, each consisting of 117 features, and the goal is to output a hypothesis that identifies whether the mushroom is poisonous or not. As in the Cod-RNA, the dataset does not have a standard separation between training data and testing data, and we use cross-validation to measure the success rate of the hypothesis obtained from our learning algorithm.

For the case of high privacy regime (corresponding to low probability of distinguishing whether any particular data point belongs to the training set) our results show better accuracy on the test data for Adult and Mushrooms, when compared to the differential privacy linear regression (DR-LP) algortihms, which is the standard benchmark algorithm in this area.

See Figure 5.2 in Section 5.1 for plots and clarifications about the plots.

1.3 Related Work

Decision trees are among the most popular classifiers, that are often used for their efficiency and interpretability. Since the NP-completeness result of Hyafil and Rives [16], there has been an extensive body of research devoted to designing heuristic algorithms for inducing decision trees. These algorithms are efficient and successful in practice [28]. Notable examples are greedy procedures such as ID3, C4.5, and CART [26, 27, 3]. They construct the trees iteratively, "growing" a tree by adding children to some leaf node of an existing tree according to some *splitting criterion*.

Differentially Private Decision Trees. Many previous works explored differentially private algorithms for learning *single* decision trees. Authors in [2] showed how a traditional non-private algorithm (ID3) could be modified to achieve differential privacy by adding noise to the splitting criterion. Later, [12] empirically demonstrated the effectiveness of using the exponential mechanism to privately select splits for ID3 and C4.5.

Recent work modified the TopDown algorithm of Kearns and Mansour [20] to enforce differential privacy [31]. This is particularly interesting because TopDown is *not* a heuristic. Under a *weak learning* assumption — if the features considered for splitting have some advantage over random guessing — TopDown is guaranteed to learn a tree with low training error. Wang, Dick, and Balcan [31] preserve this guarantee under differential privacy by appealing to the utility of the Laplace Mechanism. Here, we implement a simpler DP-TopDown algorithm — as the goal of our work is to test differentially private *boosting,* weaker tree induction is perferable.

Differentially Private Boosting. Differentially private boosting is less well-studied because the iterative structure of boosting algorithms complicates the task of enforcing privacy while maintaining utility. In theory, [11] designed the first differentially private boosting algorithm. Later, Bun, Carmosino, and Sorrell [4] offered a much simpler private algorithm based on the hard-core lemma of [1]. Both algorithms preserved privacy by using "smooth" distributions over the sample to limit the "attention" any one example receives from a base learner. Our `LazyBB` (Algorithm 2) is an implementation of the algorithm of [4] over decison trees and stumps.

Boosting by reweighting updates an explicit distributions over the data, where the probability mass on an example reflects how difficult it is to classify. *Gradient Boosting* iteratively fits the residuals of the combined voting classifier — it alters the labels instead of explicit weights on each sample.

One very recent experimental work studies differentially private *gradient* tree boosting [22]. Their base learner is an ensemble of greedily-constructed decision trees on disjoint subsets of the data, so that parallel composition may be used *inside* the base learner to save privacy. They deal with the "too much attention" problem by *clipping* the pseudo-residuals at each round, so that outliers do not compromise privacy by over-influencing the hypothesis at any round. They use composition to spread the privacy budget across each round of boosting.

Our algorithm is boosting by *reweighting* and uses much simpler base learners. Our update rule is just multiplicative weights, and we enforce privacy by *projecting* the resulting distribution over examples into the space of smooth distributions. Our algorithm remains accurate in the *high-privacy* ($\epsilon < 1$) setting; whereas [22] did not explore this regime.

Chapter 2

Preliminaries

2.1 Notation

We use the following notation for better exposition and readability.

Definition 2.1.1. *We denote the exponential function* $\exp(x) = e^x$.

2.2 Distributions and Smoothness

To preserve privacy of the data, we will never concentrate too much attention on a single example. This can be enforced by only using *smooth distributions* — where no example is allowed to have too much relative weight.

Definition 2.2.1 (κ-smooth distributions)**.** *For a parameter* $\kappa \in [0,1]$ *a probability distribution* μ *over a domain* X *is* κ-smooth *if for each* $x \in X$ *we have* $\mu(x) \leq \frac{1}{\kappa|X|}$*, where* $\kappa \in [0,1]$.

In addition we will need the related notion of κ-*dense* measures, defined below. A *measure* on a set X is a function $\mu\colon X \to [0,1]$ that need not sum to one. Note that normalizing measures to total weight naturally results in a distribution over X.

Definition 2.2.2 (κ-dense measure)**.** *For a parameter* $\kappa \in [0,1]$ *a measure* μ *overa domain* X *is said to be* κ-dense *if for each* $x \in X$ *we have* $\mu(x) \leq \frac{|\mu|}{\kappa|X|}$*, where* $|\mu| = \sum_{x \in X} \mu(x)$.

Next we define the notion of Kullback-Leibler (KL) divergence. Informally speaking KL divergence measure how one probability distribution is different from another.

Definition 2.2.3 (Kullback-Leibler divergence)**.** *Let* μ_1 *and* μ_2 *be bounded measures over the same domain* X*. The* Kullback-Leibler divergence *between* μ_1 *and* μ_2 *is defined:*

$$\mathrm{KL}(\mu_1 \parallel \mu_2) = \sum_{x \in X} \mu_1(x) \ln\left(\frac{\mu_1(x)}{\mu_2(x)}\right) + \mu_2(x) - \mu_1(x)$$

Next we define the notion of Bregman projection. This will be one of the key notions required for our boosting algorithm.

Definition 2.2.4 (Bregman projection)**.** *Let* $\Gamma \subseteq \mathbb{R}^{|S|}$ *be a non-empty closed convex set of measures over* S*. The* Bregman projection *of* $\tilde{\mu}$ *onto* Γ *is defined as:* $\Pi_\Gamma \tilde{\mu} = \arg\min_{\mu \in \Gamma} \mathrm{KL}(\mu \parallel \tilde{\mu})$.

The result of Bregman 1967 says Bregman projections do not badly "distort" KL-divergence. Moreover, when Γ is the set of κ-dense measures we can compute $\Pi_\Gamma \tilde{\mu}$ for measure $\tilde{\mu}$ with $|\tilde{\mu}| < \kappa |X|$ [1]. Finally, we require the following notion of similarity.

Definition 2.2.5 (Statistical Distance)**.** *The* statistical distance*, a.k.a.* total variation distance*, between two distributions* μ *and* ν *on* Ω*, denoted* $\mathtt{d}(\mu, \nu)$*, is defined as* $\mathtt{d}(\mu, \nu) = \max_{S \subset \Omega} |\mu(S) - \nu(S)|$.

For finite sets Ω, $\mathtt{d}(\mu, \nu) = 1/2 \sum_{x \in \Omega} |\mu(x) - \nu(x)|$ e.g., see Proposition 4.2 in [21]. Finally we define Laplace distribution, which will be used in order to sample noise that helps us guarantee privacy.

Definition 2.2.6 (Laplace distribution)**.** *For scale parameter* $b > 0$*, the Laplace distribution is defined with the following probability density function:*

$$p(x) = \frac{1}{2b} \exp(-\frac{x}{b}) \ .$$

2.3 Learning

Throughout the thesis we let $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}^n$, where each $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,r}) \in \mathbb{R}^r$ is a data point represented by a vector of its features, and each $y_i = y(\mathbf{x}_i) \in \{\pm 1\}$ denotes the unique Boolean label of $\mathbf{x}_i$. Though our techniques readily extend to continuous-feature or multi-label learning, studying this restricted Boolean classification setting simplifies the presentation and experiments.

Definition 2.3.1 (Hypothesis Class)**.** *A hypothesis class, denoted by* H*, is a family of functions from the domain* $\mathcal{X}$ *to the labels* $\{\pm 1\}$*. We call any member of the hypothesis class* $h \in H$ *a hypothesis.*

Definition 2.3.2 (A learning algorithm)**.** *Let* $S \in (\mathcal{X} \times \{\pm 1\})^n$ *be a training set of size* n*, and assume the elements of* S *are sampled i.i.d from an unknown distribution* $\mathcal{D}$*. Let* H *be a hypothesis class of functions* $h \colon \mathcal{X} \to \{\pm 1\}$*. A* learning algorithm *takes the random variable* S *as an input, and outputs a hypothesis* h*.*

We hope/expect that the hypothesis h *obtained from the learning algorithm will output the correct labels on the unknown underlying distribution of data. One way to formalize this,*

is the $(\alpha, \beta)-$Probably Approximately Correct *(PAC) guarantees, that is:*

$$\Pr_{S\sim\mathcal{D}^n}\left[\Pr_{(x,y)\sim\mathcal{D}}[h(x)\neq y] < \alpha\right] > 1-\beta \ .$$

We remark that while the general definition of PAC learning is commonly used in theory, for practical applications we usually just run our learning algorithm on a training set, and then estimate the quality of the hypothesis only on uniform distribution over the *given test examples*, while the test examples are sampled from an unknown distribution $\mathcal{D}$ (which is typically very far from uniform over all possible examples). This estimation serves as an estimation of $\Pr_{(x,y)\sim\mathcal{D}}[h(x)\neq y(x)]$.

Definition 2.3.3 (Weak Learner)**.** *Let* $S \in (\mathcal{X}\times\{\pm 1\})^n$ *be a training set of size* n*. Let* μ *be a distribution over* $[n]$*. A* weak learning algorithm *with* advantage γ *takes* (S,μ) *as input and outputs a hypothesis* $h : \mathcal{X}\to\{\pm 1\}$ *such that:* $\Pr_{x\sim\mu}[h(x)=y(x)] \geq 1/2+\gamma$*, where* $y(x)$ *is the unique label of the data point* $x\in\mathcal{X}$*.*

It is worth emphasizing that the weak learner is expected to output a hypothesis h that has advantage γ over the trivial (random) hypothesis only for the training set, and is not required to have any guarantees about the test dataset.

Definition 2.3.4 (Margin)**.** *For binary classification, the* margin *(denoted* σ*) of an ensemble* $H = h_1,\ldots,h_\tau$ *consisting of* τ *hypotheses on an example* (x,y) *is a number between* $-\tau$ *and* τ *that captures how "right" the classifier as a whole is* $\sigma(H,x,y) = y\sum_{j=1}^{\tau} h(x)$*.*

As mentioned earlier, the goal of any learner is to achieve high accuracy (or low error) with respect to the underlying unknown distribution of data, often called the *population accuracy*. In practice we can only estimate this quantity. When we are provided with a test dataset that is separate from the training dataset, we can use the accuracy of the hypothesis with respect to the test data as an estimate its accuracy with respect to the underlying distribution of data. When we do not have such an additional dataset, we have to re-use our learning data; however the accuracy of the hypothesis with respect to the training data does not ensure generalization and is often too optimistic as an estimate for the population accuracy. One approach to guarantee the generalization in this situations is cross-validation.

Definition 2.3.5 (k-fold cross-validation)**.** *Given a learning algorithm* A*, and training sample* S*, first partition* S *into* $S_1,...,S_k$ *of equal sizes (if* $|S|$ *is not divisible by* k *just put the reminder in the last partition). For each round* $1\leq i\leq k$*, we use all the partitions but* S_i *to find a hypothesis and then use* S_i *to estimate its accuracy, then average all. Let's denote by* $S_{-i} = S\setminus S_i$*, and* $h_i = A(S_{-i})$*. Then* k*-fold cross-validation is defined as follows:*

$$CV_k(A,S) = \frac{1}{k}\sum_{i\in[k]}\frac{1}{|S_i|}\sum_{(x,y)\in S_i}\mathbb{1}_{\{h_i(x)=y\}} \ .$$

2.4 Decision Trees

Definition 2.4.1 (Binary Decision Tree)**.** *A decision tree $T : \mathcal{X} \to \{\pm 1\}$ is a simple classifier represented as a rooted directed tree, in which any node has no more than 2 children. Each leaf $\ell \in T$ has a label $y(\ell) \in \{\pm 1\}$ assigned to it. Moreover, each internal node v is equipped with a simple predicate $pred_v : \mathcal{X} \to \{\pm 1\}$ that decides whether an incoming sample should go to the right or left sub-tree.*

For an input $x \in \mathcal{X}$, in order to obtain $T(x)$ one starts from the root, and follows the predicates rule until reaching a leaf ℓ. The output the label of T on x is the label of the leaf $y(\ell)$.

In general, depending on the knowledge domain for the task at hand, the predicate functions of the internal nodes are selected from a simple family of functions like thresholds on single features. Here we only consider the case in which $\mathcal{X} = \{0,1\}^m$ and the predicates of internal nodes v are of the form $pred_v(x) = x_i$ for some $i \in [m]$.

2.5 Differential Privacy

The definition of differential privacy relies on the notion of neighboring datasets. We say two datasets are neighboring if they differ in a single record. We write $D \sim D'$ when two datasets D, D' are neighboring.

Definition 2.5.1 ((ϵ, δ)-Differential Privacy [8])**.** *For $\epsilon, \delta \in \mathbb{R}_+$, we say that a randomized computation M is* (ϵ, δ)-differentially private *if for any neighboring datasets $D \sim D'$, and for any set of outcomes $S \subseteq \text{range}(M)$,*

$$\Pr[M(D) \in S] \leq \exp(\epsilon) \Pr[M(D') \in S] + \delta.$$

When $\delta = 0$, we say M is ϵ-differentially private*.*

Remark 2.5.1 (δ – the probability of catastrophic failure)**.** Let M be randomized algorithm that is (ε, δ) differentially private. Now let D, D' be two neighbouring input for M, and O a potential output for M. Note that if $\Pr[M(D) = O] = 0$, then the only guarantee we have is that $\Pr[M(D') = 0] \leq \delta$, which means if we only consider D and D' it can be the case that with probability δ it would be completely clear from the output of M which one was used as the input. Hence we call δ the probability of catastrophic failure and it should be very small for the DP guarantee to be meaningful.

Now, consider an algorithm that gets a sensitive dataset D of n records as input. Then randomly select δn records from D and remove them, and finally release the remaining dataset as output. It is easy to see that this algorithm is $(\varepsilon = 0, \delta)$-DP. This simple observation suggests that we should keep $\delta \leq \frac{1}{n}$.

Differentially private algorithms must be calibrated to the sensitivity of the function of interest with respect to small changes in the input dataset, defined formally as follows.

Definition 2.5.2 (Sensitivity). *The sensitivity of a function* $F\colon X \to \mathbb{R}^k$ *is* $\max_{D\sim D'\in X} ||F(D)-F(D')||_1$. *A function with sensitivity* Δ *is called* Δ*-sensitive.*

In the context of differential privacy, a *mechanism* is any randomized algorithm that acts as an interface to the data, and answer specific queries. In following subsection we generalize a few standard differential private mechanisms that will serve as building blocks of our private weak learners.

An important feature of the differential privacy framework is its ability to guarantee privacy under composition of multiple mechanisms. Two privacy composition theorems, sequential composition and parallel composition, are widely used in the design of mechanisms.

A sequential composition of mechanisms $M_1, ..., M_k$ is a new mechanisms M which for any given input dataset D runs M_i for $1 \leq i \leq k$ one-by-one, providing D and the output of all previous mechanisms as the input to M_i. Finally M outputs the result of M_k. Notice that in each round M_i gets a chance to learn even more about D given the results of previous rounds. We see our boosting algorithm as a sequential composition of single rounds of boosting, and we use sequential composition theorem to analyse the aggregated privacy cost of the boosting algorithm.

A parallel composition of mechanisms $M_1, ..., M_k$ is new mechanism M which for any given input dataset D first deterministically partition D to $D_1, ..., D_k$ and then run each M_i independently on D_i. Finally M outputs the aggregated result of all the mechanisms, that is $(M_1(D_1), ..., M_k(D_k))$. Notice that for any neighboring datasets D, D' only of the corresponding partitions can be different. We use parallel composition to analyse the aggregate privacy cost of different branches of our private decision trees.

Theorem 2.5.3 (Sequential Composition [5, 9, 11, 24]). *Suppose a set of privacy mechanisms* $M = \{M_1, \ldots, M_k\}$ *are sequentially performed on a dataset, and each* M_i *is* (ϵ_i, δ_i)*-differentially private with* $\epsilon_i \leq \epsilon_0$ *and* $\delta_i \leq \delta_0$ *for every* $1 \leq i \leq k$. *Then mechanism* M *satisfies* (ϵ, δ)*-differential privacy where*

- $\epsilon = k\epsilon_0$ *and* $\delta = k\delta_0$ *(the basic composition), or*
- $\epsilon = \sqrt{2k \ln 1/\delta'}\epsilon_0 + k\epsilon_0(\mathrm{e}^{\epsilon_0} - 1)$ *and* $\delta = \delta' + k\delta_0$ *for any* $\delta' > 0$ *(the advanced composition).*

We emphasise that in Theorem 2.5.3 both the basic and the advanced composition refer to the same algorithm, and the only difference is in the way the privacy of the algorithm is analyzed. Advanced composition allows us to tune δ' in order to trade privacy budget ε with probability of catastrophic failure δ.

Theorem 2.5.4 (Parallel Composition [23]). *Let* $D_1, \ldots, D_k$ *be a partition of the input domain and suppose* $M_1, \ldots, M_k$ *are mechanisms so that* M_i *satisfies* ϵ_i*-differential privacy.*

Then the mechanism $M(S) = (M_1(S \cap D_1), \ldots, M_k(S \cap D_k))$ *satisfies* $(\max_i \epsilon_i)$*-differential privacy.*

2.6 Differentially Private Learning

For the sake of our differentially private boosting algorithm we require a slightly different notion of privacy for weak learners. Given two neighboring datasets and *almost* the same distributions on them, we require weak learners to output the same hypothesis with high probability.

Definition 2.6.1 (DP Weak Learning)**.** *A weak learning algorithm* $\texttt{WkL} : S \times \mathcal{D}(S) \to \mathcal{H}$ *is* $(\epsilon, \delta, \zeta)$*-differentially private if for all neighboring samples* $S \sim S' \in (\mathcal{X}^n \times \{\pm 1\})$ *and all* $H \subseteq \mathcal{H}$*, and any pair of distributions* $\hat{\mu}, \hat{\mu}'$ *on* $[n]$ *with* $\mathtt{d}(\hat{\mu}, \hat{\mu}') < \zeta$*, we have:*

$$\Pr[\texttt{WKL}(S, \hat{\mu}) \in H] \leq \exp(\epsilon) \Pr[\texttt{WKL}(S', \hat{\mu}') \in H] + \delta.$$

Note that the notion of sensitivity for differentially private weak learners depends on the promised total variation distance ζ. Hence, differentially private weak learners must be calibrated to the sensitivity of the function of interest with respect to small changes in the distribution on the dataset.

Definition 2.6.2 (Robust Sensitivity)**.** *The robust sensitivity of a function* $F \colon (X, \mathsf{M}) \to \mathbb{R}^k$ *where* M *is the set of all distributions on* X *is defined as*

$$\max_{\substack{D \sim D' \in X \\ \hat{\mu}, \hat{\mu}' \in \mathsf{M} : \mathtt{d}(\hat{\mu}, \hat{\mu}') < \zeta}} ||F(D, \hat{\mu}(D)) - F(D', \hat{\mu}'(D'))||_1.$$

A function with robust sensitivity Δ_ζ *is called* Δ_ζ *robustly sensitive.*

Exponential Mechanism is an algorithmic technique in differential privacy that given a set of all possible outputs of some algorithm and scores $q_h \in \mathbb{R}$ associated with each solution $h \in H$, where each score q_h depends on some input parameters, outputs a random solution h with probability that is proportional to $\exp(\eta q_h)$ for some parameter η. The standard Exponential Mechanism [24] typically considers only the uniform distribution over the possible inputs, and does not consider utility functions with an auxiliary weighting μ. But for weak learning we only demand privacy (close output distributions) when *both* the dataset and measures are "close." When both promises hold and μ is fixed, the Exponential Mechanism is indeed a differentially private weak learner.

Definition 2.6.3 (Weighted Exponential Mechanism)**.** *Let* $\eta > 0$ *and let* $q_{D,\mu} \colon \mathcal{H} \to \mathbb{R}$ *be a quality score. Then, the* Weighted Exponential Mechanism $WEM(\eta, q_{D,\mu})$ *outputs* $h \in \mathcal{H}$ *with probability proportional to* $\exp\left(\eta \cdot q_{D,\mu}(h)\right)$.

Similar to the Exponential Mechanism one can prove privacy and utility guarantee for the Weighted Exponential Mechanism.

Theorem 2.6.4. *Suppose the quality score $q_{D,\mu}\colon \mathcal{H} \to \mathbb{R}$ has robust sensitivity Δ_ζ. Then, $WEM(\eta, q_{D,\mu})$ is $(2\eta\Delta_\zeta, 0, \zeta)$-differentially private weak learner. Moreover, for every $\beta \in (0,1)$, $WEM(\eta, q_D)$ outputs $h \in \mathcal{H}$ so that*

$$\Pr\left[q_{D,\mu}(h) \geq \max_{h' \in \mathcal{H}} q_{D,\mu}(h') - \ln\left(|\mathcal{H}|/\beta\right)/\eta\right] \geq 1 - \beta.$$

Proof. The goal is to prove, given two neighboring datasets and two similar distributions on them, the Weighted Exponential Mechanism outputs the same hypothesis with high probability.

In what follows let M denote the Weighted Exponential Mechanism, and let $\mathcal{H} = \text{Range}(M)$. Suppose $\mathcal{S}, \mathcal{S}'$ are two neighboring datasets of size n and μ, μ' are distributions over $[n]$ such that $\mathtt{d}(\mu, \mu') < \zeta$. Furthermore, let $q_{D,\mu}\colon \mathcal{H} \to \mathbb{R}$ be a quality score that has robust sensitivity Δ_ζ. That is, for every hypothesis $h \in \mathcal{H}$, we have

$$\max_{\substack{D \sim D' \\ \mu,\mu' : \mathtt{d}(\mu,\mu') < \zeta}} |q_{D,\mu}(h) - q_{D',\mu'}(h)| \leq \Delta_\zeta. \tag{2.1}$$

We proceed to prove that for any $h \in \mathcal{H}$ the following holds

$$\Pr[M(S,\mu) = h] \leq \exp(2\eta\Delta_\zeta)\Pr[M(S',\mu') = h].$$

Recall that M outputs a hypothesis h with probability proportional to $\exp(\eta \cdot q_{D,\mu})$ with $\eta = \frac{\epsilon}{2\Delta_\zeta}$. Let us expand the probabilities above,

$$\begin{aligned}\frac{\Pr[M(\mathcal{S},\mu) = h]}{\Pr[M(\mathcal{S}',\mu') = h]} &= \frac{\exp\left(\eta \cdot q_{\mathcal{S},\mu}(h)\right)}{\exp\left(\eta \cdot q_{\mathcal{S}',\mu'}(h)\right)} \\ &\quad \times \frac{\sum_{h \in \mathcal{H}} \exp\left(\eta \cdot q_{\mathcal{S}',\mu'}(h)\right)}{\sum_{h \in \mathcal{H}} \exp\left(\eta \cdot q_{\mathcal{S},\mu}(h)\right)}.\end{aligned}$$

Consider the first term, then

$$\begin{aligned}\frac{\exp\left(\eta \cdot q_{\mathcal{S},\mu}(h)\right)}{\exp\left(\eta \cdot q_{\mathcal{S}',\mu'}(h)\right)} &= \exp\left(\eta[q_{\mathcal{S},\mu}(h) - q_{\mathcal{S}',\mu'}(h)]\right) \\ &\leq \exp\left(\eta \cdot \Delta_\zeta\right). \qquad \text{(By (2.1))}\end{aligned}$$

Now consider the second term, then

$$\frac{\sum_{h\in\mathcal{H}} \exp\left(\eta \cdot q_{\mathcal{S}',\mu'}(h)\right)}{\sum_{h\in\mathcal{H}} \exp\left(\eta \cdot q_{\mathcal{S},\mu}(h)\right)} \leq \frac{\sum_{h\in\mathcal{H}} \exp\left(\eta \cdot [q_{\mathcal{S},\mu}(h) + \Delta_\zeta]\right)}{\sum_{h\in\mathcal{H}} \exp\left(\eta \cdot q_{\mathcal{S},\mu}(h)\right)}$$
$$= \frac{\exp\left(\eta\Delta_\zeta\right) \sum_{h\in\mathcal{H}} \exp\left(\eta \cdot q_{\mathcal{S},\mu}(h)\right)}{\sum_{h\in\mathcal{H}} \exp\left(\eta \cdot q_{\mathcal{S},\mu}(h)\right)}$$
$$= \exp\left(\eta\Delta_\zeta\right).$$

Hence, it follows that

$$\frac{\Pr[M(\mathcal{S},\mu) = h]}{\Pr[M(\mathcal{S}',\mu') = h]} \leq \exp\left(\eta\Delta_\zeta\right) \cdot \exp\left(\eta\Delta_\zeta\right)$$
$$= \exp\left(2\eta\Delta_\zeta\right).$$

This implies that, for $\eta > 0$, WEM is a $(2\eta\Delta_\zeta, 0, \zeta)$-differentially private weak learner. (Note that setting $\eta = \frac{2\epsilon}{2\Delta_\zeta}$ yields a $(\epsilon, 0, \zeta)$-differentially private weak learner.) □

We point out that the proof for the utility guarantee of Theorem 2.6.4 is identical to the proof of the utility guarantee in standard Exponential Mechanism [24].

Another differentially private mechanism that we use is Weighted Return Noisy Max (WRNM). Let $f_1, \ldots, f_k$ be k quality functions where each $f_i : S \times \mathcal{D}(S) \to \mathbb{R}$ maps datasets and distributions over them to real numbers. For a dataset S and distribution μ over S, WRNM adds independently generated Laplace noise $Lap(1/\eta)$ to each f_i and returns the index of the largest noisy function i.e. $i^* = \underset{i}{\text{argmax}}(f_i + Z_i)$ where each Z_i denotes a random variable drawn independently from the Laplace distribution with scale parameter $1/\eta$.

Theorem 2.6.5. *Suppose each f_i has robust sensitivity at most Δ_ζ. Then WRNM is a $(2\eta\Delta_\zeta, 0, \zeta)$-differentially private weak learner.*

Proof. Our proof follows the same steps as the standard Return Noisy Max explained in [10] with slight modification. Again, our goal is to prove, given two neighboring datasets and two similar distributions on them, the WRNM outputs the same hypothesis index.

Let $f_1, \ldots, f_k$ be k quality functions where each $f_i : \mathcal{S} \times \mathcal{D}(S) \to \mathbb{R}$ maps datasets and distributions over them to real numbers. For a dataset S and distribution μ over S, WRNM adds independently generated Laplace noise $Lap(1/\eta)$ to each f_i and returns the index of the largest noisy function i.e. $i^* = \underset{i}{\text{argmax}}(f_i + Z_i)$ where each Z_i denotes a random variable drawn independently from the Laplace distribution with scale parameter $1/\eta$. In what follows let M denote the WRNM.

Suppose $\mathcal{S}, \mathcal{S}'$ are two neighboring datasets of size n and μ, μ' are distributions over $[n]$ such that $\mathtt{d}(\mu, \mu') < \zeta$. Furthermore, suppose each f_i has robust sensitivity at most Δ_ζ. That

is, for every index $i \in \{1, \ldots, k\}$, we have

$$\max_{\substack{D \sim D' \\ \mu, \mu' : \mathtt{d}(\mu, \mu') < \zeta}} |f_i(\mathcal{S}, \mu) - f_i(\mathcal{S}', \mu')| \leq \Delta_\zeta. \tag{2.2}$$

Fix any $i \in \{1, \ldots, k\}$. We will bound the ratio of the probabilities that i is selected by M with inputs $\mathcal{S}, \mathcal{S}'$ and distributions μ, μ'.

Fix $Z_{-i} = (Z_1, \ldots, Z_{i-1}, Z_{i+1}, \ldots, Z_k)$, where each $Z_j \in Z_{-i}$ is drawn from $Lap(1/\eta)$. We first argue that

$$\frac{\Pr[M(\mathcal{S}, \mu) = i \mid Z_{-i}]}{\Pr[M(\mathcal{S}', \mu') = i \mid Z_{-i}]} \leq \mathrm{e}^{2\eta \cdot \Delta_\zeta}.$$

Define Z^* to be the minimum Z_i such that

$$f_i(\mathcal{S}, \mu) + Z^* > f_j(\mathcal{S}, \mu) + Z_j \quad \forall j \neq i.$$

Note that, having fixed Z_{-i}, M will output i only if $Z_i \geq Z^*$. Recalling (2.2), for all $j \neq i$, we have the following,

$$\begin{aligned} f_i(\mathcal{S}', \mu') + Z^* + \Delta_\zeta \geq f_i(\mathcal{S}, \mu) + Z^* &> f_j(\mathcal{S}, \mu) + Z_j \\ &\geq f_j(\mathcal{S}', \mu') + Z_j - \Delta_\zeta. \end{aligned}$$

This implies that

$$f_i(\mathcal{S}', \mu') + Z^* + 2\Delta_\zeta \geq f_j(\mathcal{S}', \mu') + Z_j.$$

Now, for dataset $\mathcal{S}'$, distribution μ', and Z_{-i}, mechanism M selects the i-th index if Z_i, drawn from $Lap(1/\eta)$, satisfies $Z_i \geq Z^* + 2\Delta_\zeta$.

$$\begin{aligned} &\Pr_{Z_i \sim Lap(1/\eta)}[M(\mathcal{S}', \mu') = i \mid Z_{-i}] \\ &\geq \Pr_{Z_i \sim Lap(1/\eta)}[Z_i \geq Z^* + 2\Delta_\zeta] \\ &\geq \mathrm{e}^{-(2\eta\Delta_\zeta)} \Pr_{Z_i \sim Lap(1/\eta)}[Z_i \geq Z^*] \\ &= \mathrm{e}^{-(2\eta\Delta_\zeta)} \Pr_{Z_i \sim Lap(1/\eta)}[M(\mathcal{S}, \mu) = i \mid Z_{-i}]. \end{aligned}$$

Multiplying both sides by $\mathrm{e}^{(2\eta\Delta_\zeta)}$ yields the desire bound.

$$\frac{\Pr_{Z_i \sim Lap(1/\eta)}[M(\mathcal{S}, \mu) = i \mid Z_{-i}]}{\Pr_{Z_i \sim Lap(1/\eta)}[M(\mathcal{S}', \mu') = i \mid Z_{-i}]} \leq \mathrm{e}^{(2\eta\Delta_\zeta)}.$$

This implies that, for $\eta > 0$, WRNM is a $(2\eta\Delta_\zeta, 0, \zeta)$-differentially private weak learner. (Note that setting $\eta = \frac{2\epsilon}{2\Delta_\zeta}$ yields a $(\epsilon, 0, \zeta)$-differentially private weak learner.) $\square$

2.7 DP Learning of 1-Rules

1-Rules are the simplest kind of decision trees, that is trees with only one node! Throughout this section let $\mathcal{S} = \{(\mathbf{x}_i, y_i)\}^n$, where each $\mathbf{x}_i = (x_{i,1}, \ldots, x_{i,r}) \in \{\pm 1\}^r$ denotes a datapoint, and let μ be a distribution over $[n]$. We will brute-force "1-Rules," also known as Decision Stumps[1] [17, 15]. Here, these simply evaluate a single Boolean literal such as $\neg x_{17}$ — an input variable that may or may not be negated. We also admit the constants True and False as literals.

A brutally simple, but surprisingly effective weak learner returns the literal with optimal weighted agreement to the labels. For any 1-Rule h define $\mathrm{err}(\mathcal{S}, \mu, h)$ to be:

$$\mathrm{err}(\mathcal{S}, \mu, h) = \sum_{(\mathbf{x}_i, y_i) \in \mathcal{S}} \mu(i) \chi\{h(\mathbf{x}_i) \neq y\}.$$

For learning 1-Rules under DP constraints, the natural approach is to use the Exponential Mechanism to noisily select the best possible literal. There is a small type error: the standard Exponential Mechanism does not consider utility functions with an auxiliary weighting μ. But for weak learning we only demand privacy (close output distributions) when *both* the dataset and measures are "close." When both promises hold and μ is fixed, the Exponential Mechanism is indeed a differentially private 1-Rule learner. We show this formally below.

[1] Decision Stumps are also known as the *Dictator functions* in the context of analysis of boolean functions.

Algorithm 1 Differentially Private 1-Rule Induction$(\mathcal{S}, \mu, \eta)$

Require: Dataset $\mathcal{S}$, distribution μ over $[1, \ldots, |\mathcal{S}|]$, and $\eta > 0$.

1: Let $\mathcal{H}$ be the set of all literals over $\mathcal{S}$ plus the constants True and False
2: **for** $h \in \mathcal{H}$ **do**
3: $\quad q_{\mathcal{S},\mu}(h) \leftarrow -\mathrm{err}(\mathcal{S}, \mu, h)$.
4: **end for**
5: $h_{out} \leftarrow$ select a hypothesis $h \in \mathcal{H}$ with probability proportional to $\exp(\eta \cdot q_{\mathcal{S},\mu}(h))$
6: **return** h_{out}

Observation 2.7.1. *Let $\mathcal{S} \sim \mathcal{S}'$ be any two neighboring datasets and set $I = \mathcal{S} \cap \mathcal{S}'$. Then, for any two distributions μ, μ' over $[n]$, we have*

$$
\begin{aligned}
&\left|\mathrm{err}(\mathcal{S}, \mu, T) - \mathrm{err}(\mathcal{S}', \mu', T)\right| \\
&= \Bigg| \sum_{(\mathbf{x}_i, y_i) \in \mathcal{S}} \mu(i) \chi\{T(\mathbf{x}_i) \neq y\} \\
&\qquad - \sum_{(\mathbf{x}_i, y_i) \in \mathcal{S}'} \mu'(i) \chi\{T(\mathbf{x}_i) \neq y\} \Bigg| \\
&= \Bigg| \sum_{(\mathbf{x}_i, y_i) \in \mathcal{S} \cap \mathcal{S}'} [\mu(i) - \mu'(i)] \chi\{T(\mathbf{x}_i) \neq y\} \\
&\qquad + \sum_{(\mathbf{x}_i, y_i) \in \mathcal{S} \triangle \mathcal{S}'} [\mu(i) - \mu'(i)] \chi\{T(\mathbf{x}_i) \neq y\} \Bigg| \\
&\leq \sum_{(\mathbf{x}_i, y_i) \in \mathcal{S} \cap \mathcal{S}'} |\mu(i) - \mu'(i)| + \sum_{(\mathbf{x}_i, y_i) \in \mathcal{S} \triangle \mathcal{S}'} |\mu(i) - \mu'(i)| \\
&= \sum_{i=1}^{n} |\mu(i) - \mu'(i)| = 2\mathtt{d}(\mu, \mu').
\end{aligned}
$$

Theorem 2.7.2. *Algorithm 1 is a $(4\eta\zeta, 0, \zeta)$-differentially private weak learner.*

Proof. Suppose $\mathcal{S}, \mathcal{S}'$ are two neighboring datasets of size n and μ, μ' are distributions over $[n]$ such that $\mathtt{d}(\mu, \mu') < \zeta$. Observation 2.7.1 tells us that the quality score $q_{\mathcal{S},\mu}(h) = -\mathrm{err}(\mathcal{S}, \mu, h)$ has robust sensitivity 2ζ. Hence, by Theorem 2.6.4, we have that Algorithm 1 is a $(4\eta\zeta, 0, \zeta)$-differentially private weak learner. □

Theorem 2.7.3. *Let h_{opt} denote the optimal hypothesis in $\mathcal{H}$. Then Algorithm 1, with probability at least $1 - \beta$, returns $h_{out} \in \mathcal{H}$ such that*

$$
\mathrm{err}(h_{out}) \leq \mathrm{err}(h_{opt}) + \frac{1}{\eta} \ln \frac{|\mathcal{H}|}{\beta}.
$$

Proof. By Theorem 2.6.4, with probability at least $1-\beta$, we have

$$q_{\mathcal{S},\mu}(h_{out}) \geq \max_{h\in\mathcal{H}} q_{\mathcal{S},\mu}(h) - \frac{1}{\eta}\ln\frac{|\mathcal{H}|}{\beta}. \tag{2.3}$$

Note that $q_{\mathcal{S},\mu}(h) = -\text{err}(\mathcal{S},\mu,h)$ for all $h\in\mathcal{H}$ and $\max_{h\in\mathcal{H}} q_{\mathcal{S},\mu}(h) = -\text{err}(h_{opt})$. This gives us

$$\begin{aligned} -\text{err}(h_{out}) &\geq -\text{err}(h_{opt}) - \frac{1}{\eta}\ln\frac{|H|}{\beta} \implies \\ \text{err}(h_{out}) &\leq \text{err}(h_{opt}) + \frac{1}{\eta}\ln\frac{|\mathcal{H}|}{\beta}. \end{aligned}$$

□

As we already discussed, in order to construct PAC learners by boosting weak learners we need weak learners that only beat random guessing on any distribution over the training set. Here, we wish to use Algorithm 1 as a weak learner. That is, we show that Algorithm 1 (with high probability) is better than random guessing. In what follows we have Theorem 2.7.4 and its proof.

Theorem 2.7.4. *Under a weak learner assumption with advantage γ, Algorithm 1, with probability at least $1-\beta$, is a weak learner with advantage at least $\gamma - \frac{1}{\eta}\ln\frac{|\mathcal{H}|}{\beta}$. That is, for any distribution μ over $\{1,\ldots,|\mathcal{S}|\}$, we have*

$$\sum_{(\mathbf{x}_i,y_i)\in\mathcal{S}} \mu(i)\chi\{h_{out}(\mathbf{x}_i)\neq y\} \leq 1/2 - \left(\gamma - \frac{1}{\eta}\ln\frac{|\mathcal{H}|}{\beta}\right).$$

Proof. By Theorem 2.7.3, Algorithm 1 with probability at least $1-\beta$ outputs a hypothesis h_{out} such that

$$\text{err}(h_{out}) \leq \text{err}(h_{opt}) + \frac{1}{\eta}\ln\frac{|\mathcal{H}|}{\beta}.$$

Under a *weak learner assumption*, we assume that an optimal hypothesis h_{opt} is at least as good as random guessing. That is $\text{err}(h_{opt}) < 1/2 - \gamma$. This yields the desired result. □

Chapter 3

Private Boosting

Our boosting algorithm simply calculates the current margin of each example at each round, exponentially weights the sample accordingly, and then calls a private base learner with smoothed sample weights. The hypothesis returned by this base learner is added to the ensemble H, then the process repeats. Privacy follows from (advanced) composition and the definitions of differentially private weak learning. Utility (low training error) follows from regret bounds for lazy projected mirror descent and a reduction of boosting to zero-sum games. Theorem 3.0.1 formalizes these guarantees; for the proof, see [4]. Next, we discuss the role of each parameter.

Round Count τ. The number of base hypotheses. In the non-private setting, τ is like a regularization parameter — we increase it until just before overfitting is observed. In the private setting, there is an additional trade-off: more rounds *could* decrease training error until the amount of noise we must inject into the weak learner at each round (to preserve privacy) overwhelms progress.

Learning rate λ. Exponential weighting is attenuated by a *learning rate* λ to ensure that weights do not shift too dramatically between calls to the base learner. λ appears negatively because the margin is negative when the ensemble is incorrect. Signs cancel to make the weight on an example *larger* when the committee is bad, as desired.

Smoothness κ. Base learners attempt to maximize their probability of correctness over each intermediate distribution. Suppose the t-th distribution was a point mass on example x_i — this would pose a serious threat to privacy, as hypothesis h_t would only contain information about individual x_i! We ensure this never happens by invoking the weak learner only over κ-smooth distributions: each example has probability mass "capped" at $\frac{1}{\kappa n}$. For larger samples, we have smaller mass caps, and so can inject less noise to enforce privacy. Note that by setting $\kappa = 1$, we force each intermediate distribution to be uniform, which would entirely negate the effects of boosting: reweighting would simply be impossible. Conversely, taking $\kappa \to 0$ will entirely remove the smoothness constraint.

Algorithm 2 `LazyBB`: Weighted Lazy-Bregman Boosting

Parameters: $\kappa \in (0,1)$, desired training error; $\lambda \in (0,1)$, learning rate; $\tau \in \mathbb{N}$ number of rounds

Input: $S \in X^n$, the sample;

$H \leftarrow \emptyset$ and $\mu_1(i) \leftarrow \kappa \ \ \forall i \in [n]$ {Uniform bounded measure}

for $t = 1$ to τ **do**

$\hat{\mu}_t \leftarrow$ Normalize μ_t to a distribution {Obtaining a κ-smooth distribution}

$h_t \leftarrow \texttt{WkL}(S, \hat{\mu}_t)$

$H \leftarrow H \cup \{h_t\}$

$\sigma_t(i) \leftarrow y_i \sum_{j=1}^{t} h_j(x_i) \ \ \forall i \in [n]$ {Normalized score of current majority vote}

$\tilde{\mu}_{t+1}(i) \leftarrow \exp\left(-\lambda \sigma_t(i)\right) \kappa \ \ \forall i \in [n]$

$\mu_{t+1} \leftarrow \Pi_{\Gamma}(\tilde{\mu}_{t+1})$ {Bregman project to a κ-dense measure}

end for

Output: $\hat{f}(x) = \mathrm{Maj}_{h_j \in H}\left[h_j(x)\right]$

Theorem 3.0.1 (Privacy & Utility of `LazyBB`)**.** *Let L be a $(\epsilon_b, \delta_b, (1/\kappa n))$-DP weak learner with advantage γ and failure probability β for concept class $\mathcal{H}$. Running `LazyBB` with L for $\tau \geq \frac{16 \log(1/\kappa)}{\gamma^2}$ rounds on a sample of size n with $\lambda = \gamma/4$ guarantees:*

Privacy *`LazyBB` is (ϵ_A, δ_A)-DP, where*

- *$\epsilon_A = \tau \cdot \epsilon_b$ and $\delta_A = \tau \cdot \delta_b$ (using basic composition) or,*
- *$\epsilon_A = \sqrt{2\tau \cdot \ln(1/\delta')} \cdot \epsilon_b + \tau \cdot \epsilon_b \cdot (\exp(\epsilon_b) - 1)$ and $\delta_A = \tau \cdot \delta_b + \delta'$ for every $\delta' > 0$ (using advanced composition).*

Utility *With all but $(\tau \cdot \beta)$ probability, H has at least γ-good normalized margin on a $(1-\kappa)$ fraction of S i.e.,* $\Pr_{(x,y)\sim S}\left[y/\tau \sum_{j=1}^{\tau} h_j(x) \leq \gamma\right] \leq \kappa$.

Note that Theorem 3.0.1 above "opportunistically" switches between basic and advanced composition in the privacy guarantee. This actually matters for our experiments, because when the number of rounds is small budgeting privacy according to *basic* composition is actually better for the weak learner! Of course, for larger numbers of rounds, advanced composition is better — assuming we are willing to give up pure differential privacy for a small possibility of catastrophic privacy failure.

Weak Learner failure probability β is critical to admit because whatever "noise" process a DP weak learner uses to ensure privacy may ruin utility on some round. So, we must union bound over this event in the training error guarantee.

Chapter 4

Concrete Private Boosting

Here we specify concrete weak learners and give privacy guarantees for `LazyBB` combined with these weak learners.

4.1 Baseline: 1-Rules

To establish a baseline for performance of both private and non-private learning, we use the simplest possible hypothesis class: 1-Rules or "Decision Stumps" [17, 15]. In the Boolean feature and classification setting, these are just constants or signed literals (e.g. $-x_{17}$) over the data domain.

$$\texttt{1R}(\mathcal{S}) = \{x_i\}_{i\in[d]} \cup \{-x_i\}_{i\in[d]} \cup \{+1, -1\} \quad \text{and} \quad \text{err}(\mathcal{S}, \mu, h) = \sum_{(\mathbf{x}_i, y_i)\in\mathcal{S}} \mu(i)\chi\{h(\mathbf{x}_i) \neq y\}.$$

To learn a 1-Rule given a distribution over the training set, return the signed feature or constant with minimum weighted error. Naturally, we use the Weighted Exponential Mechanism with noise rate η to privatize selection. This is simply the Generic Private Agnostic Learner of [19], finessing the issue that "weighted error" is actually a *set* of utility functions (analysis in Section 2.7). We denote the baseline and differentially private versions of this algorithm as `1R` and `DP-1R`, respectively. We introduced the private version in Algorithm 1 and analysed its privacy in Theorem 2.7.2.

Given a total privacy budget of ϵ, we divide it uniformly across rounds of boosting. Then, by Theorem 3.0.1, we solve $\epsilon = 4\tau \cdot \eta \cdot \zeta$ for η to determine how much noise `DP-1R` must inject at each round. Note that privacy depends on the statistical distance ζ between distributions over neighboring datasets. `LazyBB` furnishes the promise that $\zeta \leq 1/\kappa n$. It is natural for ζ to depend on the number of samples: the larger the dataset, the easier it is to "hide" dependence on a single individual, and the less noise we can inject at each round. Hence the following is a direct corollary of Theorem 2.7.2 and the privacy section of Theorem 3.0.1:

Corollary 4.1.1. *`LazyBB` runs for τ rounds using `DP-1R` at noise rate $\eta = \frac{\epsilon\kappa n}{4\tau}$ is ϵ-DP.*

4.2 TopDown Decision Trees

TopDown heuristics are a family of decision tree learning algorithms that are employed by widely used software packages such as C4.5, CART, and scikit-learn. We present a differentially private TopDown algorithm that is a modification of decision tree learning algorithms given by Kearns and Mansour [20]. At a high level, TopDown induces decision trees by repeatedly *splitting* a leaf node in the tree built so far. On each iteration, the algorithm *greedily* finds the leaf and splitting function that maximally reduces an upper bound on the error of the tree. The selected leaf is replaced by an internal node labeled with the chosen splitting function, which partitions the data at the node into two new children leaves. Once the tree is built, the leaves of the tree are labeled by the label of the most common class that reaches the leaf. Algorithm 3, `DP-TopDown`, is a "reference implementation" of the differentially private version of this algorithm. `DP-TopDown`, instead of choosing the best leaf and splitting function, applies the Exponential Mechanism to noisily select a leaf and splitting function in the built tree so far. The Exponential Mechanism is applied on the set of all possible leaves and splitting functions in the current tree; this is computationally feasible in our Boolean-feature setting. Next we introduce necessary notation and discuss the privacy guarantee of our algorithm, and how it is used as a weak learner for our boosting algorithm.

DP TopDown Decision Tree. Let F denote a class of Boolean splitting functions with input domain $\mathcal{S}$. Each internal node is labeled by a splitting function $h : \mathcal{S} \to \{0, 1\}$. These splitting functions route each example $x \in \mathcal{S}$ to exactly one leaf of the tree. That is, at each internal node if the splitting function $h(x) = 0$ then x is routed to the left subtree, and x is routed to the right subree otherwise. Furthermore, let G denote the *splitting criterion*. $G : [0, 1] \to [0, 1]$ is a concave function which is symmetric about $1/2$ and $G(1/2) = 1$. Typical examples of splitting criterion function are Gini and Entropy. Here we use Gini as our splitting criterion of choice.

Definition 4.2.1. *We denote Gini function by $G(x) = 4x(1 - x)$. Note that this function is symmetric about $1/2$ and $G(1/2) = 1$*

Algorithm 3 builds decision trees in which the internal nodes are labeled by functions in F, and the splitting criterion G is used to determine which leaf should be split next, and which function $h \in F$ should be used for the split.

Let T be a decision tree whose leaves are labeled by $\{0, 1\}$ and μ be a distribution on $\mathcal{S}$. The weight of a leaf $\ell \in \mathit{leaves}(T)$ is defined to be the weighted fraction of data that reaches ℓ i.e., $w(\ell, \mu) = \Pr_\mu[x \text{ reaches } \ell]$. The weighted fraction of data with label 1 at leaf ℓ

Algorithm 3 Differentially Private TopDown-DT

Require: Data sample $\mathcal{S}$, distribution $\hat{\mu}$ over $\mathcal{S}$, number of internal nodes t, and $\eta > 0$.

$T \leftarrow$ the single-leaf tree.
$\mathcal{C} \leftarrow \mathit{leaves}(T) \times F$
while T has fewer than t internal node **do**
  $(\ell^*, h^*) \leftarrow$ select a candidate from $\mathcal{C}$ w.p. $\propto \exp(\eta \cdot \mathrm{im}_{\ell,h,\hat{\mu}})$
  $T \leftarrow T(\ell^*, h^*)$
  for each new pair $\ell \times h \in \mathit{leaves}(T) \times F$ **do**
    $\mathrm{im}_{\ell,h,\hat{\mu}} \leftarrow G(T, \hat{\mu}) - G(T(\ell, h), \hat{\mu})$
    Add $\mathrm{im}_{\ell,h,\hat{\mu}}$ to $\mathcal{C}$
  end for
end while
Label leaves by majority label [WRNM with privacy budget $8t \cdot \eta \cdot \zeta$]
Output: T

is denoted by $q(\ell, \mu)$. Given these we define error of T as follows.

$$\mathrm{err}(T, \mu) = \sum_{\ell \in \mathit{leaves}(T)} w(\ell, \mu) \min\{q(\ell, \mu), 1 - q(\ell, \mu)\}.$$

Noting that $G(q(\ell, \mu)) \geq \min\{q(\ell, \mu), 1 - q(\ell, \mu)\}$, we have an upper bound for $\mathrm{err}(T, \mu)$.

$$\mathrm{err}(T, \mu) \leq \mathcal{G}(T, \mu) = \sum_{\ell \in \mathit{leaves}(T)} w(\ell, \mu) G(q(\ell, \mu)).$$

For $\ell \in \mathit{leaves}(T)$ and $h \in F$ let $T(\ell, h)$ denote the tree obtained from T by replacing ℓ by an internal node that splits subset of data that reaches ℓ, say $\mathcal{S}_\ell$, into two children leaves ℓ_0, ℓ_1. Note that any data x satisfying $h(x) = i$ goes to ℓ_i. The quality of a pair (ℓ, h) is the improvement we achieve by splitting at ℓ according to h. Formally,

$$\mathrm{im}_{\ell,h,\mu} = \mathcal{G}(T, \mu) - \mathcal{G}(T(\ell, h), \mu).$$

At each iteration, Algorithm 3 chooses a pair (ℓ^*, h^*) according to the Exponential Mechanism with probability proportional to $\mathrm{im}_{\ell,h,\mu}$. By Theorem 2.6.4, the quality of the chosen pair (ℓ^*, h^*) is close to the optimal split with high probability.

Theorem 4.2.2 (Privacy guarantee)**.** *`DP-TopDown`, Algorithm 3, is a* $(16t \cdot \eta \cdot \zeta, 0, \zeta)$*-DP weak learner.*

Throughout this section, $\mathcal{S} \sim \mathcal{S}'$ are two neighboring datasets of size n and μ, μ' are distributions over $[n]$ such that $\mathtt{d}(\mu, \mu') < \zeta$. Observe that for a decision tree T we have $|w(\ell, \mu) - w(\ell, \mu')| \leq \zeta$ and $|q(\ell, \mu) - q(\ell, \mu')| \leq \zeta$. Before proceeding to provide an upper bound on the sensitivity of $\mathrm{im}_{\ell,h,\mu}$, we prove some useful lemmas.

Lemma 4.2.3. *The following holds.* $4\Big|w(\ell,\mu)q(\ell,\mu)(1-q(\ell,\mu)) - w(\ell,\mu')q(\ell,\mu)(1-q(\ell,\mu'))\Big| \leq \frac{5}{4}\zeta$

Proof. As the Gini criterion $G(q) = 4q(1-q)$ is symmetric about $1/2$, without loss of generality, we assume $q(\ell) \leq 1/2$. Furthermore, suppose $w(\ell,\mu)q(\ell,\mu)(1-q(\ell,\mu))$ is greater than $w(\ell,\mu')q(\ell,\mu')(1-q(\ell,\mu'))$. The arguments for the other cases are analogous.

$$\begin{aligned}
&w(\ell,\mu)q(\ell,\mu)(1-q(\ell,\mu)) - w(\ell,\mu')q(\ell,\mu')(1-q(\ell,\mu')) \\
&\leq w(\ell,\mu)q(\ell,\mu)(1-q(\ell,\mu)) \\
&\quad - w(\ell,\mu')(q(\ell,\mu)-\zeta)(1-q(\ell,\mu)+\zeta) \\
&= w(\ell,\mu)q(\ell,\mu)(1-q(\ell,\mu)) \\
&\quad - w(\ell,\mu')q(\ell,\mu)(1-q(\ell,\mu)+\zeta) \\
&\quad + w(\ell,\mu')\zeta(1-q(\ell,\mu)+\zeta) \\
&\leq w(\ell,\mu)q(\ell,\mu)(1-q(\ell,\mu)) \\
&\quad - w(\ell,\mu')q(\ell,\mu)(1-q(\ell,\mu)) + \zeta \\
&\leq |w(\ell,\mu) - w(\ell,\mu')|q(\ell,\mu)(1-q(\ell,\mu)) + \zeta \leq \frac{5}{4}\zeta.
\end{aligned}$$

□

Lemma 4.2.4. *For a decision tree T and $(\ell,h) \in leaves(T) \times F$ we have*

$$\Big|\mathrm{im}_{\ell,h,\mu}(\mathcal{S}) - \mathrm{im}_{\ell,h,\mu'}(\mathcal{S}')\Big| \leq 4\zeta.$$

Proof. For dataset $\mathcal{S}$ let $\mathcal{G}(T) = \sum_{\ell \in leaves(T)} w(\ell)G(q(\ell))$. Recall the definition of $\mathrm{im}_{\ell,\mathrm{h},\mu}$,

$$\begin{aligned}
\mathrm{im}_{\ell,h,\mu}(\mathcal{S}) &= \mathcal{G}(T,\mu) - \mathcal{G}(T(\ell,h),\mu) \\
&= w(\ell,\mu)G(q(\ell,\mu)) - w(\ell_0,\mu)G(q(\ell_0,\mu)) \\
&\quad - w(\ell_1,\mu)G(q(\ell_1,\mu)).
\end{aligned}$$

Similarly, for dataset $\mathcal{S}'$ let $\mathcal{G}(T,\mu') = \sum_{\ell \in leaves(T)} w(\ell,\mu')G(q(\ell,\mu'))$. Then we have

$$\begin{aligned}
&\mathrm{im}_{\ell,h,\mu'}(\mathcal{S}') = \mathcal{G}(T,\mu') - \mathcal{G}(T(\ell,h),\mu') \\
&= w(\ell,\mu')G(q(\ell,\mu')) - w(\ell_0,\mu')G(q(\ell_0,\mu')) \\
&\quad - w(\ell_1,\mu')G(q(\ell_1,\mu')).
\end{aligned}$$

Having these we can rewrite $\left|\mathrm{im}_{\ell,h,\mu}(\mathcal{S}) - \mathrm{im}_{\ell,h,\mu'}(\mathcal{S}')\right|$ as follows,

$$\begin{aligned}
&\left|\mathrm{im}_{\ell,h,\mu}(\mathcal{S}) - \mathrm{im}_{\ell,h,\mu'}(\mathcal{S}')\right| \\
&= \left|\mathcal{G}(T,\mu) - \mathcal{G}(T(\ell,h),\mu) - \mathcal{G}(T,\mu') + \mathcal{G}(T(\ell,h),\mu')\right| \\
&= \Big| w(\ell,\mu)G(q(\ell,\mu)) - w(\ell_0,\mu)G(q(\ell_0,\mu)) \\
&\quad - w(\ell_1,\mu)G(q(\ell_1,\mu)) - w(\ell,\mu')G(q(\ell,\mu')) \\
&\quad + w(\ell_0,\mu')G(q(\ell_0,\mu')) + w(\ell_1,\mu')G(q(\ell_1,\mu'))\Big| \\
&\leq \left|w(\ell,\mu)G(q(\ell,\mu)) - w(\ell,\mu')G(q(\ell,\mu'))\right| \\
&\quad + \left|w(\ell_0,\mu')G(q(\ell_0,\mu')) - w(\ell_0,\mu)G(q(\ell_0,\mu))\right| \\
&\quad + \left|w(\ell_1,\mu')G(q(\ell_1,\mu')) - w(\ell_1,\mu)G(q(\ell_1,\mu))\right| \\
&\leq 15/4\zeta \leq 4\zeta,
\end{aligned}$$

where the last inequalities follow by Lemma 4.2.3. □

Now we are ready to prove Theorem 4.2.2:

Proof of Theorem 4.2.2. Let us denote Algorithm 3 by M. Consider a fixed decision tree T. We prove that, given $\mathcal{S} \sim \mathcal{S}'$ and μ, μ', Algorithm 3 chooses the same leaf and split function with high probability.

Let $\mathcal{C} = leaves(T) \times F$ denote the set of possible split candidates. For each $(\ell, h) \in \mathcal{C}$, $\mathrm{im}_{\ell,h,\mu}(\mathcal{S})$ denotes the improvement gained in classification of dataset $\mathcal{S}$ by splitting T at leaf ℓ according to split function h. Similarly, we have $\mathrm{im}_{\ell,h,\mu'}(\mathcal{S}')$. Provided that $\mathtt{d}(\mu,\mu') \leq \zeta$, by Lemma 4.2.4, the robust sensitivity of quality score $\mathrm{im}_{\ell,h,\mu}$ is at most 4ζ. Similar to the proof of Theorem 2.6.4 it follows that

$$\frac{\Pr[M(\mathcal{S},\mu) = (\ell,h)]}{\Pr[M(\mathcal{S}',\mu') = (\ell,h)]} \leq \exp\left(8 \cdot \eta \cdot \zeta\right).$$

This means each selection procedure where `DP-TopDown` selects a leaf and a splitting function is $(8 \cdot \eta \cdot \zeta, 0, \zeta)$-differentially private. Using composition theorem for differentially private mechanisms, Theorem 2.5.3, yields privacy guarantee

$$\tilde{\varepsilon} = 8t \cdot \eta \cdot \zeta,$$

for the construction of the internal nodes. We use $\tilde{\varepsilon}$ for labeling the leaves using Laplace Mechanism. Since the leaves partition dataset, this preserves $\tilde{\varepsilon}$-differential privacy by parallel

composition of deferentially private mechanisms (Theorem 2.5.4). Overall, `TopDown-DT` is an $(16t \cdot \eta \cdot \zeta, 0, \zeta)$-differentially private weak learner. □

Remark 4.2.1. Using advanced composition for differentially private mechanisms, Theorem 2.5.3, for every $\tilde{\delta} > 0$ yields privacy guarantee

$$\tilde{\varepsilon}_{\tilde{\delta}} = t(8 \cdot \eta \cdot \zeta)^2 + 8 \cdot \eta \cdot \zeta \sqrt{t \ln(1/\tilde{\delta})},$$

for the construction of the internal nodes. We use $\tilde{\varepsilon}_{\tilde{\delta}}$ for labeling the leaves using Laplace Mechanism. Since the leaves partition dataset, this preserves $\tilde{\varepsilon}_{\tilde{\delta}}$-differential privacy by parallel composition of deferentially private mechanisms (Theorem 2.5.4). Overall, `TopDown-DT` is an $(2\tilde{\varepsilon}_{\tilde{\delta}}, \tilde{\delta}, \zeta)$-differentially private weak learner.

As before, given a total privacy budget of ϵ, we divide it uniformly across rounds of boosting. Then, by Theorem 3.0.1, we solve $\epsilon = 16\tau \cdot t \cdot \eta \cdot \zeta$ for η to determine how much noise `DP-TopDown` must inject at each round. `LazyBB` furnishes the promise that $\zeta \leq 1/\kappa n$. Overall:

Corollary 4.2.5. *`LazyBB` runs for τ rounds using `DP-TopDown` at noise rate $\eta = \frac{\epsilon\kappa n}{16\tau t}$ is ϵ-DP.*

Chapter 5

Experiments and Conclusions

5.1 Experiments

Here we compare our smooth boosting algorithm (`LazyBB`) over both decision trees and 1-Rules to: differentially private logistic regression using objective perturbation (DP-LR) [6], Differentially Private Bagging (DP-Bag) [18], and Privacy-Preserving Gradient Boosting Decision Trees (DPBoost) [22]. In our implementation we used the IBM differential privacy library (available under MIT licence) [14] for standard DP mechanisms and accounting, and scikit-learn (available under BSD licence) for infrastructure [25]. These experiments show that smooth boosting of *1-Rules* can yield improved model accuracy and sparsity under identical privacy constraints.

We experiment[1] with three freely available real-world datasets. **Adult** (Available from UCI Machine Learning Repository) has 32,561 training examples, 16,282 test examples, and 162 features after dataset-oblivious one-hot coding — which incurs no privacy cost. The task is to predict if someone makes more than 50k US dollars per year from Census data. Our reported accuracies are holdout tests on the canonical test set associated with Adult. **Cod-RNA** (available from the LIBSVM website) has 59,535 training examples and 80 features after dataset-oblivious one-hot coding, and asks for detection of non-coding RNA. **Mushroom** (available from the LIBSVM website) has 8124 training examples and 117 features after one-hot coding, which asks to identify poisonous mushrooms. For Mushroom and CodRNA, we report cross-validated estimates of accuracy. All experiments were run on a 3.8 GHz 8-Core Intel Core i7 with 16GB of RAM consumer desktop computer.

5.1.1 Parameter Selection Without Assumptions

We select parameters for `LazyBB` and `DP-LR` entirely using grid-search and cross-validation (Table 5.1.1 for `LazyBB`) for each value of epsilon plotted i.e. $\epsilon \in (0.05, 0.1, \cdots, 0.5, 1, 3, 5)$. Over the small datasets we use for experiments, the Weak Learner assumption does not hold

[1]Implementation and datasets available at: `http://www.sfu.ca/~bsalamat/Thesis_Supplemental.zip`.

WKL	Parameter		
	τ	λ	κ
`OneRule`	$5, 9, 15, 19, 25, 29,$ $39, 49, 65, 75, 99$	$0.2, 0.25,$ $\cdots, 0.5$	$0.2, 0.25,$ $\cdots, 0.5$
`TopDown`	$5, 9, 15, 19, 25, 29,$ $35, 39, 45, 51$	$0.35, 0.4$	$0.25, 0.3$

Table 5.1: Parameters grid

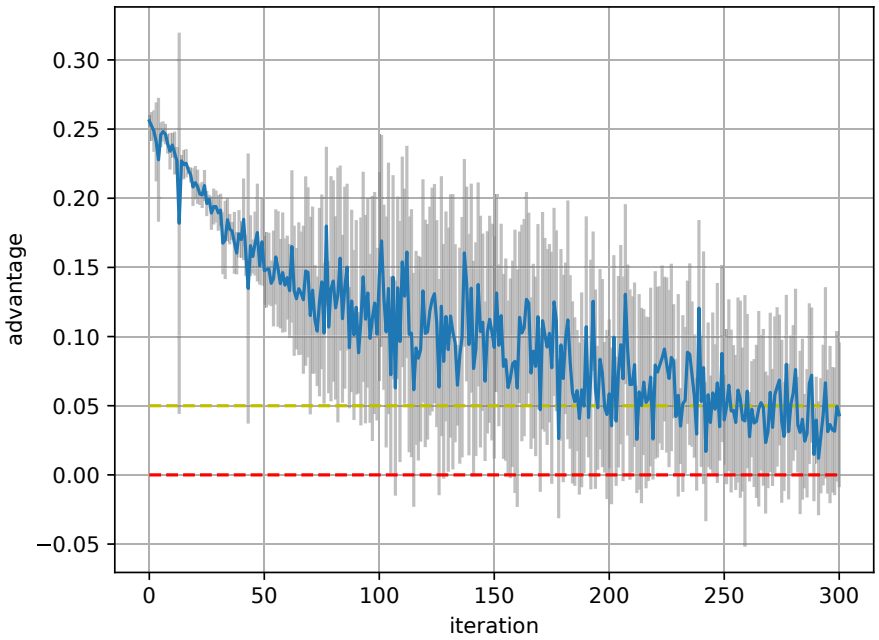

Figure 5.1: Advantage curve and margin histogram.

for "long enough" to realize the training error guarantee of Theorem 3.0.1. For example, fixing $\kappa = 1/2$ — seeking "good" margin on only half the training set — suppose we have a $(1/20)$-advantage Weak Learner. That is, at every round of boosting, each new hypothesis has accuracy at least 55% over the intermediate distribution. Under these conditions, Theorem 3.0.1 guarantees utility after approximately 4,000 rounds of boosting. Figure 5.1 plots advantage on the Adult dataset at each round of boosting with $\lambda = \gamma/4$ as required by Theorem 3.0.1, averaged over 10 runs of the boosting decision stumps with total privacy budget $\epsilon = 1$. The weak learner assumption fails after only 250 rounds of boosting.

And yet, even when run with much *faster* learning rate λ, we see good accuracy from `LazyBB` — the assumption holds for *small* τ, ensuring that `DP-1R` has advantage. So, Theorem 3.0.1 is much more pessimistic than is warranted. This is a know limitation of

the analysis for any *non*-adaptive boosting algorithm [30]. In the non-private setting, we set λ very slow and boost for "many" rounds, until decay in advantage triggers a stopping criterion. In the private setting (where non-adaptivity makes differential privacy easier to guarantee) running for "many" rounds is not feasible; noise added for privacy would saturate the model. These experiments motivate further theoretical investigation of boosting dynamics for non-adaptive algorithms, due to their utility in the privacy-preserving setting.

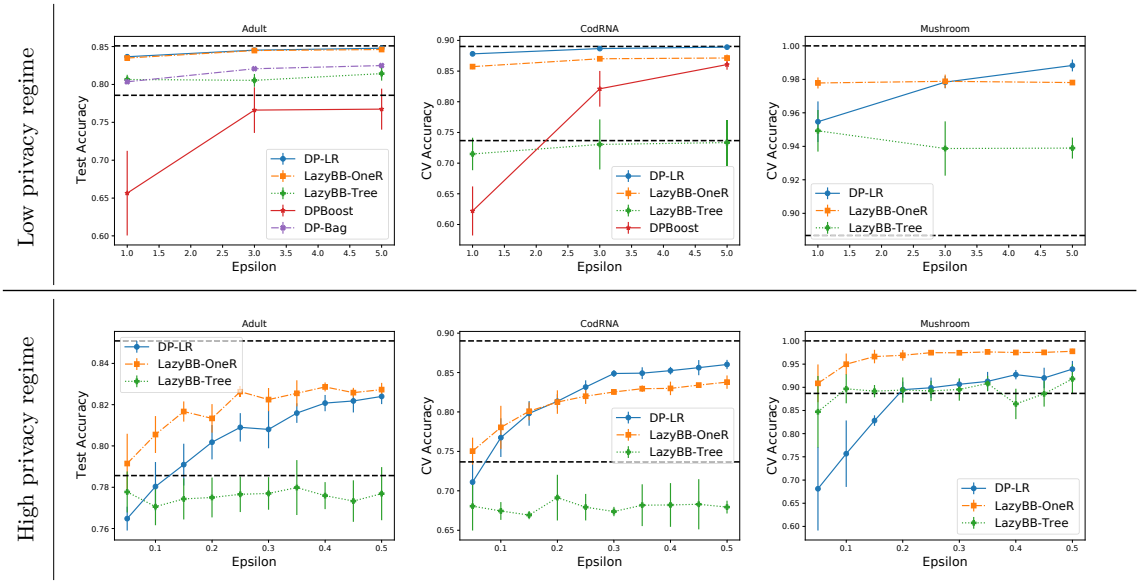

Figure 5.2: Learning Curves — Privacy vs. accuracy.

5.1.2 Results

In Figure 5.2 we plot the accuracy of each of the 5 methods above against privacy constraint ϵ, along with two non-private baselines to both quantify the "cost of privacy" and ensure that the private learners are non-trivial. The strong non-private baseline is the implementation of Gradient Boosted Trees in sklearn, the weak non-private baseline is a single 1-Rule. It is important to note that for DP-Bag and DPBoost, we only compare our results for datasets and regimes that the corresponding hyperparameters are reported in the related works. Surprisingly, we found that `LazyBB` over 1-Rules and differentially private logistic regression were the best performing models — despite being the *simplest* algorithms to state, reason about, and run.

5.1.3 Effect of Approximate Differential Privacy

Figures 5.3 and 5.4 compare the cross validation average accuracy on Adult dataset in the pure and approximate differential privacy regimes, for two different strategies of hyperparemeter selection: oblivious to ϵ (Figure 5.3), and ϵ-dependent (Figure 5.4). This emphasizes the importance of tuning hyperparameters for each choice of ϵ *separately*. For approximate

differential privacy, we consider the small constant value of $\delta = 10^{-5}$, the same as that used by DP-Bag [18].

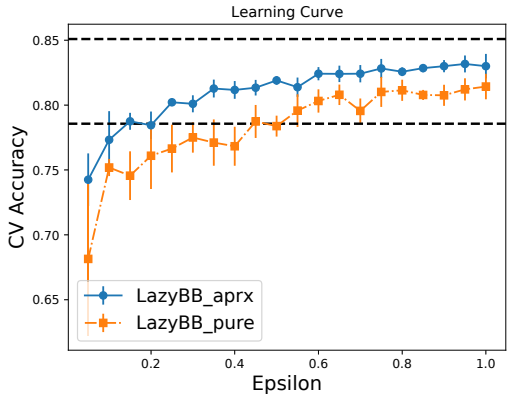

Figure 5.3: CV accuracy on Adult of (ϵ, δ)-DP `LazyBB` ($\kappa = 1/4,\ \lambda = 1/4,\ \tau = 99$) with `DP-1R`, $\delta \in \{0, 10^{-5}\}$, varying ϵ, vs. non-private baselines.

When we set hyperparameters identically for each ϵ, using approximate differential privacy can allow significantly increased accuracy at each ϵ. We found this to be the case especially for higher τ; we select $\tau = 99$ to illustrate. However, if we are allowed to separately optimize for each ϵ, the significance of this advantage disappears. Though average accuracy clearly improves, it is not outside one standard deviation of average accuracy for pure differential privacy. It seems that boosted 1-Rules are too simple to distinguish between pure and approximate differential privacy constraints on this small dataset.

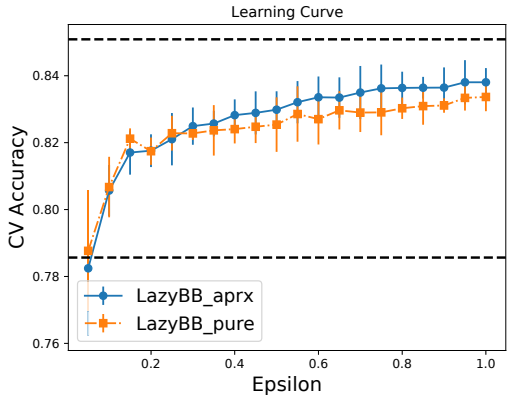

Figure 5.4: CV accuracy on Adult of (ϵ, δ)-DP `LazyBB` with `DP-1R`, $\delta \in \{0, 10^{-5}\}$, varying ϵ, vs. non-private baselines, with best model for each ϵ displayed.

5.1.4 Sparsity, regularization, and interpretability

Algorithms used for high-stakes decisions should be both well-audited and privacy-preserving. However, often there is a trade-off between privacy and interpretability [13]. Generally, noise injected to protect privacy harms interpretability. But our algorithms maintain accuracy under strong privacy constrains while admitting a high level of sparsity — which facilitates interpretability. Table 5.2 lists measurements across different levels of privacy. For an example of boosted one-rules at $\epsilon = 0.4$ DP, see Table 5.3.

`DP-LR` — another simple algorithm with excellent performance — uses L_2 regularization to achieve sparsity and better generalization. While L_2 regularization keeps all the assigned weights relatively small, it generally assigns a non-negligible weight to *every* features. Hence, the resulting model becomes less interpretable as the dimension of data grows. On other hand, `LazyBB` with 1-Rules controls sparsity indirectly by the number of rounds of boosting. Just as with non-private non-adaptive boosting algorithms, we can see this as a greedy approximation to L_1 regularization of a linear model [29]. Moreover, the final model can be interpreted as a simple integral weighted voting of features.

ε	features count mean	features count std	% features
0.40	6.4	0.800	3.95%
0.50	12.8	0.400	7.90%
1.00	30.6	1.200	18.88%
3.00	72.8	2.481	44.93%
5.00	49.8	2.785	30.74%

Table 5.2: Statistics of number of features used by `LazyBB` with `DP-1R` across different levels of privacy on adult dataset. See the Appendix for the complete table.

votes	(feature, value)
3	marital-status : Married-civ-spouse
-2	capital-gain = 0
1	occupation : Exec-managerial
1	occupation : Prof-specialty
1	13 <= education-num <= 14.5
-1	age <= 17

Table 5.3: A 0.4-DP model obtained by training `LazyBB` with `DP-1R` on adult dataset with 0.82 accuracy.

5.1.5 Pessimistic Generalization Theory

Empirically, `LazyBB` generalizes well. As with AdaBoost, we could try to explain this with large margins and Rademacher complexity, which applies to any voting classifier. So, we estimated the Rademacher complexity of 1-Rules over each dataset to predict test error. The bounds are far more pessimistic than the experiments. Intuitively, if `LazyBB` showed larger margins on the training data than on unseen data, this would constitute a *membership inference attack* — which is ruled out by differential privacy. This motivates theoretical investigation of new techniques to guarantee generalization of differentially private models trained on small samples.

The following table compares the best guaranteed lower bound derived by estimated Rademacher complexity and the test accuracy. The test accuracy of Adult dataset is obtained by evaluating the model on the test set, which was not touched during training. For Cod-RNA

and Mushroom dataset there is no canonical test set available, so we report cross-validation accuracy.

Dataset	Rademacher Estimate of Test Accuracy	(CV) test accuracy
Adult	0.37	0.83
Cod-Rna	0.09	0.86
Mushroom	0.49	0.98

Table 5.4: Comparison between Rademacher estimates of generalization performance and experimental generalization performance for boosted 1-Rules, at $\epsilon = 1$.

Appendix A

Sparsity statistics of the experiments

In Section 5.1 of the main body, we discussed sparsity and interpretability of `LazyBB` with 1-Rules. Here we share the complete table of sparsity measurements for all the experiments. For each level of privacy, we use the hyper-parameter selected by cross-validation and repeated the experiment 5 times to obtain confidence bounds.

ε	features count mean	features count std	% features
0.05	4.6	0.489	2.83%
0.10	4.8	0.400	2.96%
0.15	3.8	0.400	2.34%
0.20	3.6	1.019	2.22%
0.25	7.0	0.632	4.32%
0.30	13.8	1.166	8.51%
0.35	7.6	0.489	4.69%
0.40	6.4	0.800	3.95%
0.45	19.2	2.785	11.85%
0.50	12.8	0.400	7.90%
1.00	30.6	1.200	18.88%
3.00	72.8	2.481	44.93%
5.00	49.8	2.785	30.74%

Table A.1: Sparsity measurements for Adult dataset.

ε	features count mean	features count std	% features
0.05	6.0	0.632	7.50%
0.10	12.4	1.744	15.50%
0.15	19.6	1.625	24.50%
0.20	16.8	1.327	21.00%
0.25	11.2	0.748	14.00%
0.30	10.2	0.748	12.75%
0.35	26.4	1.356	33.00%
0.40	19.8	2.482	24.75%
0.45	34.4	1.497	43.00%
0.50	25.4	2.653	31.75%
1.00	54.2	3.187	67.75%
3.00	44.2	2.227	55.25%
5.00	32.0	2.098	40.00%

Table A.2: Sparsity measurements for Cod-RNA dataset.

ε	features count mean	features count std	% features
0.05	4.6	0.490	3.93%
0.10	7.2	1.166	6.15%
0.15	5.8	0.748	4.95%
0.20	8.6	1.497	7.35%
0.25	6.2	0.748	5.29%
0.30	5.6	0.490	4.78%
0.35	9.0	0.894	7.69%
0.40	9.8	1.166	8.37%
0.45	9.4	1.356	8.03%
0.50	11.8	1.720	10.08%
1.00	14.4	1.625	12.03%
3.00	28.8	2.926	24.61%
5.00	11.8	0.748	10.08%

Table A.3: Sparsity measurements for Mushroom dataset.

Appendix B

Hyperparameters

These are the hyperparemeters selected by cross-validation of boosted 1-Rules over each of our datasets. The privacy vs. accuracy curves use these settings for each value of ϵ.

ε	density	learning rate	no. estimators
0.05	0.50	0.50	5
0.10	0.45	0.50	5
0.15	0.50	0.40	5
0.20	0.50	0.30	5
0.25	0.35	0.50	9
0.30	0.40	0.40	19
0.35	0.30	0.45	9
0.40	0.35	0.50	9
0.45	0.40	0.45	25
0.50	0.35	0.50	15
1.00	0.35	0.45	39
3.00	0.35	0.45	99
5.00	0.35	0.45	75

Table B.1: Hyperparameters selected by cross-validation for Adult dataset.

ε	density	learning rate	no. estimators
0.05	0.50	0.50	9
0.10	0.50	0.35	19
0.15	0.50	0.50	29
0.20	0.40	0.50	25
0.25	0.50	0.45	25
0.30	0.50	0.45	25
0.35	0.45	0.35	49
0.40	0.45	0.45	39
0.45	0.50	0.40	65
0.50	0.45	0.50	49
1.00	0.40	0.50	99
3.00	0.30	0.40	99
5.00	0.35	0.40	99

Table B.2: Hyperparameters selected by cross-validation for Cod-Rna dataset.

ε	density	learning rate	no. estimators
0.05	0.45	0.50	5
0.10	0.50	0.40	9
0.15	0.50	0.45	9
0.20	0.50	0.40	15
0.25	0.30	0.40	9
0.30	0.35	0.50	9
0.35	0.40	0.35	15
0.40	0.45	0.40	19
0.45	0.35	0.20	19
0.50	0.45	0.25	25
1.00	0.25	0.30	29
3.00	0.20	0.20	75
5.00	0.20	0.50	29

Table B.3: Hyperparameters selected by cross-validation for Mushroom dataset.

Bibliography

[1] Boaz Barak, Moritz Hardt, and Satyen Kale. The uniform hardcore lemma via approximate bregman projections. In Claire Mathieu, editor, *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 1193–1200. SIAM, 2009.

[2] Avrim Blum, Cynthia Dwork, Frank McSherry, and Kobbi Nissim. Practical privacy: the sulq framework. In Chen Li, editor, *Proceedings of the Twenty-fourth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 13-15, 2005, Baltimore, Maryland, USA*, pages 128–138. ACM, 2005.

[3] Leo Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth, 1984.

[4] Mark Bun, Marco Leandro Carmosino, and Jessica Sorrell. Efficient, noise-tolerant, and private learning via boosting. In Jacob D. Abernethy and Shivani Agarwal, editors, *Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria]*, volume 125 of *Proceedings of Machine Learning Research*, pages 1031–1077. PMLR, 2020.

[5] Mark Bun and Thomas Steinke. Concentrated differential privacy: Simplifications, extensions, and lower bounds. In Martin Hirt and Adam D. Smith, editors, *Theory of Cryptography - 14th International Conference, TCC 2016-B, Beijing, China, October 31 - November 3, 2016, Proceedings, Part I*, volume 9985 of *Lecture Notes in Computer Science*, pages 635–658, 2016.

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

[7] Cynthia Dwork. Differential privacy. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *Automata, Languages and Programming, 33rd International Colloquium, ICALP 2006, Venice, Italy, July 10-14, 2006, Proceedings, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 1–12. Springer, 2006.

[8] Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 486–503. Springer, 2006.

[9] Cynthia Dwork and Jing Lei. Differential privacy and robust statistics. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 371–380. ACM, 2009.

[10] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.

[11] Cynthia Dwork, Guy N. Rothblum, and Salil P. Vadhan. Boosting and differential privacy. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 51–60. IEEE Computer Society, 2010.

[12] Arik Friedman and Assaf Schuster. Data mining with differential privacy. In Bharat Rao, Balaji Krishnapuram, Andrew Tomkins, and Qiang Yang, editors, *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, July 25-28, 2010*, pages 493–502. ACM, 2010.

[13] Frederik Harder, Matthias Bauer, and Mijung Park. Interpretable and differentially private predictions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34:4083–4090, Apr. 2020.

[14] Naoise Holohan, Stefano Braghin, Pól Mac Aonghusa, and Killian Levacher. Diffprivlib: The IBM differential privacy library. *CoRR*, abs/1907.02444, 2019.

[15] Robert C. Holte. Very simple classification rules perform well on most commonly used datasets. *Mach. Learn.*, 11:63–91, 1993.

[16] Laurent Hyafil and Ronald L. Rivest. Constructing optimal binary decision trees is np-complete. *Inf. Process. Lett.*, 5(1):15–17, 1976.

[17] Wayne Iba and Pat Langley. Induction of one-level decision trees. In Derek H. Sleeman and Peter Edwards, editors, *Proceedings of the Ninth International Workshop on Machine Learning (ML 1992), Aberdeen, Scotland, UK, July 1-3, 1992*, pages 233–240. Morgan Kaufmann, 1992.

[18] James Jordon, Jinsung Yoon, and Mihaela van der Schaar. Differentially private bagging: Improved utility and cheaper privacy than subsample-and-aggregate. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 4325–4334, 2019.

[19] Shiva Prasad Kasiviswanathan, Homin K. Lee, Kobbi Nissim, Sofya Raskhodnikova, and Adam D. Smith. What can we learn privately? *SIAM J. Comput.*, 40(3):793–826, 2011.

[20] Michael J. Kearns and Yishay Mansour. On the boosting ability of top-down decision tree learning algorithms. In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 459–468. ACM, 1996.

[21] David A Levin and Yuval Peres. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.

[22] Qinbin Li, Zhaomin Wu, Zeyi Wen, and Bingsheng He. Privacy-preserving gradient boosting decision trees. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 784–791. AAAI Press, 2020.

[23] Frank McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. *Commun. ACM*, 53(9):89–97, 2010.

[24] Frank McSherry and Kunal Talwar. Mechanism design via differential privacy. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pages 94–103. IEEE Computer Society, 2007.

[25] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[26] J. Ross Quinlan. Induction of decision trees. *Mach. Learn.*, 1(1):81–106, 1986.

[27] J. Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[28] Lior Rokach and Oded Maimon. *Data Mining with Decision Trees*. WORLD SCIENTIFIC, 2nd edition, 2014.

[29] Saharon Rosset, Ji Zhu, and Trevor Hastie. Boosting as a regularized path to a maximum margin classifier. *J. Mach. Learn. Res.*, 5:941–973, 2004.

[30] Robert E. Schapire and Yoav Freund. *Boosting: Foundations and Algorithms*. The MIT Press, 2012.

[31] Kaiwen Wang, Travis Dick, and Maria-Florina Balcan. Scalable and provably accurate algorithms for differentially private distributed decision tree learning. *CoRR*, abs/2012.10602, 2020.

[32] Alexandra Wood, Micah Altman, Aaron Bembenek, Mark Bun, Marco Gaboardi, James Honaker, Kobbi Nissim, David R. OBrien, Thomas Steinke, and Salil Vadhan. Differential privacy: A primer for a non-technical audience. *Vanderbilt Journal of Entertainment & Technology Law*, 21(1):209–275, 2018.