Private Truly-Everlasting Robust-Prediction

Uri Stemmer [1]

Abstract

Private everlasting prediction (PEP), recently introduced by Naor et al. [2023], is a model for differentially private learning in which the learner never publicly releases a hypothesis. Instead, it provides black-box access to a "prediction oracle" that can predict the labels of an *endless stream* of unlabeled examples drawn from the underlying distribution. Importantly, PEP provides privacy both for the initial training set and for the endless stream of classification queries. We present two conceptual modifications to the definition of PEP, as well as new constructions exhibiting significant improvements over prior work. Specifically, we incorporate robustness against poisoning attacks into the definition of PEP; we present a relaxed privacy definition, suitable for PEP, that allows us to disconnect the privacy parameter δ from the number of total time steps T; and we present new constructions for axis-aligned rectangles and decision-stumps exhibiting improved sample complexity and runtime.

1. Introduction

The line of work on *private learning*, introduced by Kasiviswanathan et al. (2011), aims to understand the computational and statistical aspects of PAC learning while providing strong privacy protections for the training data. Recall that a (non-private) PAC learner is an algorithm that takes a training set containing classified random examples and returns a hypothesis that should be able to predict the labels of fresh examples from the same distribution. A *private* PAC learner must achieve the same goal while guaranteeing differential privacy w.r.t. the training data. Intuitively, this means that the produced hypothesis should not be significantly affected by any particular labeled example. Formally, the definition of differential privacy is as follows.

[1]Tel Aviv University and Google Research, Israel. Correspondence to: Uri Stemmer <u@uri.co.il>.

Proceedings of the 41^{st} International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

Definition 1.1 (Dwork et al. (2006))**.** *Let $\mathcal{A}$ be a randomized algorithm whose input is a dataset. Algorithm $\mathcal{A}$ is* (ε, δ)-differentially private (DP) *if for any two datasets D, D' that differ on one row (such datasets are called* neighboring*) and for any event F it holds that* $\Pr[\mathcal{A}(D) \in F] \leq e^{\varepsilon} \cdot \Pr[\mathcal{A}(D') \in F] + \delta$.

Unfortunately, it is now known that private learning can require *significantly more resources* than non-private learning, which can be very prohibitive. In fact, there are simple cases where private learning is known to be *impossible*, even though they are trivial without the privacy constraint. One such example is the class of all 1-dimensional threshold functions over the real line (Bun et al., 2015; Alon et al., 2022). In addition, there are cases where private learning is possible, but provably requires significantly more runtime than non-private learning (under cryptographic assumptions) (Bun & Zhandry, 2016).

The line of work on *private prediction*, introduced by Dwork & Feldman (2018), offers a different perspective on private learning that circumvents some these barriers. Specifically, in *private prediction* we consider a setting where the private learning algorithm does *not* output a hypothesis. Instead, it provides black-box access to a "prediction oracle" that can be used to predict the labels of fresh (unlabeled) examples from the underlying distribution. Works on private prediction[1] showed that, informally, for any concept class C there is a private prediction algorithm that can answer m "prediction queries" given a training set of size $\approx \mathrm{VC}(C) \cdot \sqrt{m}$. Note, however, that the number of "prediction queries" here is at most quadratic in the size of the initial training set. This hinders the acceptance of private prediction as a viable alternative to private learning, as with the latter we obtain a privatized hypothesis that could be used to predict the labels infinitely many points. There are also empirical results showing that when the number of required predictions is large, then private prediction can be inferior to classical private learning (van der Maaten & Hannun, 2020). To tackle this, Naor et al. (2023) introduced the notion of *private everlasting prediction (PEP)* that supports an *unbounded* number of prediction queries. To achieve this, Naor et al. (2023) allowed the content of the black-box to constantly

[1]See, e.g., (Dwork & Feldman, 2018; Bassily et al., 2018; Nandi & Bassily, 2020; Dagan & Feldman, 2020).

evolve as a function of the given queries, while guaranteeing privacy both for the initial training set and for the queries. Furthermore, they proved that such an "evolving" black-box is necessary in order to support an unbounded number of queries. Informally, they established the following theorem.

Theorem 1.2 (Naor et al. (2023), informal)**.** *For every concept class C there is a private everlasting predictor (PEP) using training set of size $\approx \frac{1}{\alpha\cdot\varepsilon^2}\cdot \mathrm{VC}^2(C)$, where α is the accuracy parameter and ε is the privacy parameter (ignoring the dependence on all other parameters). The algorithm is not computationally efficient.*

1.1. Our Contributions

Robustness to out-of-distribution queries. Theorem 1.2 shows that PEP could potentially be much more efficient than classical private learning. Specifically, it shows that the sample complexity of PEP is at most quadratic in the non-private sample complexity, while with classical private learning the gap could be arbitrarily large (if at all finite). However, a major limitation of PEP is that it only guarantees accuracy provided that *all* the classification queries are drawn from the *correct* underlying distribution. Specifically, the algorithm initially gets a labeled training set sampled according to some fixed (but unknown) distribution $\mathcal{D}$, and then answers an infinite sequence of prediction queries, *provided that all of these prediction queries are sampled from the same distribution $\mathcal{D}$*. This was required by previous constructions in order to be able to "refresh" the content of the black-box for supporting infinitely many queries without exhausting the privacy budget. This could be quite limiting, where after investing efforts in training the everlasting predictor, a few out-of-distribution queries might completely contaminate the prediction oracle, rendering it useless for future queries (even if these future queries would indeed be sampled from the correct distribution). We incorporate robustness against such poisoning attacks into the definition, and introduce a variant of PEP which we call *private everlasting robust prediction (PERP)*. Informally, the requirement is that the predictor should continue to provide utility, even if only a γ fraction of its queries are sampled from the correct distribution, and all other queries are *adversarial*. We do not require the algorithm to provide accurate predictions on adversarial queries, only that these adversarial queries would not break the validity of the predictor on "legitimate" queries. We observe that the construction of Naor et al. (2023) can be adjusted to satisfy our new definition of PERP. Informally,

Observation 1.3 (informal)**.** *For every concept class C there is a private everlasting* robust *predictor using training set of size $\approx \frac{1}{\alpha^2\cdot\gamma\cdot\varepsilon^2}\cdot \mathrm{VC}^2(C)$, where α is the accuracy parameter, γ is the robustness parameter, and ε is the privacy parameter (ignoring the dependence on all other parameters). The algorithm is not computationally efficient.*

Note that the "price of robustness" in this observation is roughly $\frac{1}{\alpha\gamma}$, as compared to the non-robust construction of Theorem 1.2. We leave open the possibility of a generic construction for PERP with sample complexity linear in $\frac{1}{\alpha}$, or with sample complexity that increases slower than $\frac{1}{\gamma}$. We show that these two objectives are indeed achievable in specific cases (to be surveyed later).

Disconnecting the privacy parameter δ from the time horizon. It is widely agreed that the definition of differential privacy only provides meaningful guarantees when the privacy parameter δ is much smaller than $\frac{1}{n}$, where n is the size of the dataset. The reason is that it is possible to satisfy the definition while leaking about δn records in the clear (in expectation), so we want $\delta n \ll 1$ to prevent such a leakage. In the context of PEP, this means that $1/\delta$ should be much larger than the total number of time steps (or queries), which we denote as T. The issue is that the sample complexity of all known constructions for PEP grows with $\mathrm{polylog}(\frac{1}{\delta})$, which means that the sample complexity actually grows with $\mathrm{polylog}(T)$.[2] This means that current constructions for private everlasting prediction are not really "everlasting", as a finite training set allows for supporting only a bounded number of queries (sub-exponential in the size of the training set). It is therefore natural to ask if PEP could be made "truly everlasting", having sample complexity independent of the time horizon. We answer this question in the affirmative, by presenting a relaxed privacy definition (suitable for PEP) that allows us to disconnect the privacy parameter δ from the time horizon T. More formally, we *redefine* the privacy parameter δ in a way that allows for a significantly improved dependence on the time horizon.

New constructions for PEP. Two additional shortcomings of the generic construction of Naor et al. (2023), as well as our robust extension to it, are that (1) it is computationally *inefficient*, and (2) it exhibits sample complexity *quadratic* in the VC dimension of the target class. We present a computationally *efficient* construction for the class of all axis-aligned rectangles that exhibits sample complexity *linear* in the dimension. Furthermore, our constructing achieves very strong robustness properties, where the sample complexity grows very slowly as a function of the robustness parameter γ. Specifically,

Theorem 1.4 (informal)**.** *There exists a computationally efficient PERP for axis aligned rectangles in d dimensions that uses sample complexity $\approx \frac{d}{\alpha\cdot\varepsilon^2}\cdot\log^2(\frac{1}{\gamma})$, where α is the accuracy parameter, γ is the robustness parameter, and ε is the privacy parameter (ignoring the dependence on all other parameters).*

Via a simple reduction, we show that our construction can

[2] This excludes cases where (offline) private PAC learning is easy, such as learning point functions.

be used to obtain a robust predictor also for the class of d-*dimensional decision-stumps*. (The VC dimension of this concept class is known to be $\Theta(\log d)$.) Specifically,

Theorem 1.5 (informal)**.** *There exists a computationally efficient PERP for d-dimensional decision-stumps that uses sample complexity $\approx \frac{\log d}{\alpha \cdot \varepsilon^2} \cdot \log^2(\frac{1}{\gamma})$, where α is the accuracy parameter, γ is the robustness parameter, and ε is the privacy parameter (ignoring the dependence on all other parameters).*

For both of these classes, classical private learning is known to be *impossible* (when defined over infinite domains, such as the reals). Thus, Theorems 1.4 and 1.5 provide further evidence that PEP could become a viable alternative to the classical model private learning: These theorems provide examples where classical private learning is impossible, while PEP is possible *efficiently* with sample complexity that almost matches the non-private sample complexity (while satisfying strong robustness properties).

2. Preliminaries

Notation. Two random variables Y_0, Y_1 are (ε, δ)-indistinguishable, denoted as $Y_0 \approx_{(\varepsilon,\delta)} Y_1$, if for every $b \in \{0,1\}$ and every event F it holds that $\Pr[Y_b \in F] \leq e^{\varepsilon} \cdot \Pr[Y_{1-b} \in F] + \delta$.

We now provide the formal definitions for private everlasting prediction. Before considering the utility and privacy requirements, the following definition specifies the interface, or the syntax, required from a prediction algorithm.

Definition 2.1 (Naor et al. (2023))**.** *A* prediction oracle *is an algorithm $\mathcal{A}$ with the following properties:*

1. *At the beginning of the execution, algorithm $\mathcal{A}$ receives a dataset $S \in (X \times \{0,1\})^n$ containing n labeled examples and selects a hypothesis $h_0 : X \to \{0,1\}$.*
2. *Then, in each round $r \in \mathbb{N}$, algorithm $\mathcal{A}$ gets a query, which is an unlabeled element $x_r \in X$, outputs $h_{r-1}(x_r)$ and selects a hypothesis $h_r : X \to \{0,1\}$.*

The following definition specifies the utility requirement from a prediction algorithm. Informally, the requirement is that *all* the hypotheses selected throughout the execution should have low generalization error.

Definition 2.2 (Naor et al. (2023))**.** *Let $\mathcal{A}$ be a prediction oracle. We say that $\mathcal{A}$ is an* (α, β, n)-everlasting predictor *for a concept class C over a domain X if the following holds for every concept $c \in C$ and for every distribution $\mathcal{D}$ over X. If the training set S contains n i.i.d. samples from $\mathcal{D}$ that are labeled by c, and if all the queries $x_1, x_2, \ldots$ are drawn i.i.d. from $\mathcal{D}$, then* $\Pr[\exists r \geq 0 \text{ s.t. } \mathrm{error}_{\mathcal{D}}(c, h_r) > \alpha] \leq \beta$.

The following definition specifies the privacy requirement from a prediction algorithm. Informally, we require DP w.r.t. both the initial training set and the queries. This is formalized by requiring that any (adaptive) adversary $\mathcal{B}$ cannot learn much about any single training point or any single query, even if this adversary completely determines all other points/queries throughout the execution and gets to see all of $\mathcal{A}$'s predictions for the queries it determines.

Definition 2.3 (Naor et al. (2023))**.** *A prediction oracle $\mathcal{A}$ is a* (ε, δ)-private *if for every adversary $\mathcal{B}$ and every $T \in \mathbb{N}$, the random variables $\mathrm{View}^0_{\mathcal{B},T}$ and $\mathrm{View}^1_{\mathcal{B},T}$ (defined in Figure 1) are (ε, δ)-indistinguishable.*

Remark 2.4. *$\mathrm{View}^b_{\mathcal{B},T}$ denotes all the information that the adversary $\mathcal{B}$ sees throughout the interaction specified in Figure 1. This consists of $\mathcal{B}$'s internal randomness (if any) and the sequence of predictions it gets from $\mathcal{A}$ throughout the execution. Note that once $\mathcal{A}, \mathcal{B}, T, b$ are fixed, then $\mathrm{View}^b_{\mathcal{B},T}$ is a random element which is determined by the internal randomness of $\mathcal{A}$ and $\mathcal{B}$. We remark that, without loss of generality, one may assume that the adversary $\mathcal{B}$ is deterministic.*

Remark 2.5. *Definition 2.3 slightly differs from the one given by Naor et al. (2023), in that in Definition 2.3 we protect against inclusion/exclusion of one query rather than protecting against a change to one query. This allowed us to slightly simplify the analysis of our algorithms, and has no real effect otherwise.*

Finally, we can define *private everlasting prediction* to be algorithms that satisfy both Definition 2.2 (the utility requirement) and Definition 2.3 (the privacy requirement):

Definition 2.6 (Naor et al. (2023))**.** *Algorithm $\mathcal{A}$ is an $(\alpha, \beta, \varepsilon, \delta, n)$-Private Everlasting Predictor (PEP) if it is an (α, β, n)-everlasting predictor and (ε, δ)-private.*

3. Conceptual Modifications to PEP

In this section we present the conceptual modifications we suggest to the definition of private everlasting prediction.

3.1. Robustness to out-of-distribution queries

We introduce a modified utility requirement for PEP in which only a γ fraction of the queries are assumed to be sampled from the target distribution, while all other queries could be completely adversarial. This modifies only the utility requirement of PEP (Definition 2.2), and has no effect on the privacy requirement (Definition 2.3).

Definition 3.1 (Everlasting robust prediction)**.** *Let $\mathcal{A}$ be a prediction oracle (as in Definition 2.1). We say that $\mathcal{A}$ is an* $(\alpha, \beta, \gamma, n)$-everlasting robust predictor *for a concept class C over a domain X if the following holds for every concept $c \in C$, every distribution $\mathcal{D}$ over X, and every adversary $\mathcal{F}$. Suppose that the n points in the initial dataset S are sampled i.i.d. from $\mathcal{D}$ and labeled correctly by c. Furthermore,*

Parameters: $b \in \{0,1\}, T \in \mathbb{N}$.

Training Phase:

1. The adversary $\mathcal{B}$ chooses two labeled datasets $S^0, S^1 \in (X \times \{0,1\})^*$ which are either neighboring or equal. % If S^0, S^1 are neighboring, then one of them can be obtained from the other by adding/removing one labeled point.
2. If $S^0 = S^1$ then set $c_0 = 0$. Otherwise set $c_0 = 1$. % We refer to the round r in which $c_r = 1$ as the "challenge round", or the "neighboring round". If $c_0 = 1$ then the adversary chooses to place the challenge already in the initial dataset. Otherwise, the adversary will have the chance to pose a (single) challenge in the following prediction phase.
3. Algorithm $\mathcal{A}$ gets S^b.

Prediction phase:

4. For round $r = 1, 2, \ldots, T$:
   (a) The adversary $\mathcal{B}$ outputs a point $x_r \in X$ and a bit $c_r \in \{0,1\}$, under the restriction that $\sum_{j=0}^{r} c_j \leq 1$.
   (b) If ($c_r = 0$ or $b = 1$) then algorithm $\mathcal{A}$ gets x_r and outputs a prediction $\hat{y}_r$.
   (c) If ($c_r = 1$ and $b = 0$) then algorithm $\mathcal{A}$ gets $\perp$. % Here $\perp$ is a special symbol denoting "no query". Note that if $b = 1$ then $\mathcal{A}$ always gets x_r as input. On the other hand, if $b = 0$ then in the unique round r such that $c_r = 1$, algorithm $\mathcal{A}$ gets $\perp$ instead of x_r. The adversary's goal is to distinguish between these two cases using the predictions it sees throughout the execution.
   (d) If $c_r = 0$ then the adversary $\mathcal{B}$ gets $\hat{y}_r$. % The adversary does not get $\hat{y}_r$ if $c_r = 1$.

Let $\text{View}^b_{\mathcal{B},T}$ be $\mathcal{B}$'s entire view of the execution, i.e., its internal randomness and the sequence of predictions it received.

Figure 1. Definition of $\text{View}^0_{\mathcal{B},T}$ and $\text{View}^1_{\mathcal{B},T}$.

suppose that each of the query points x_i is generated as follows. With probability γ, the point x_i is sampled from $\mathcal{D}$. Otherwise (with probability $1-\gamma$), the adversary $\mathcal{F}$ chooses x_i arbitrarily, based on all previous inputs and outputs of $\mathcal{A}$. Then, $\Pr[\exists r \geq 0 \text{ s.t. } \text{error}_{\mathcal{D}}(c, h_r) > \alpha] \leq \beta$.

We now define private everlasting robust prediction as follows.

Definition 3.2 (Private Everlasting Robust Prediction)**.** *Algorithm $\mathcal{A}$ is an $(\alpha, \beta, \gamma, \varepsilon, \delta, n)$-Private Everlasting Robust Predictor (PERP) if it is an $(\alpha, \beta, \gamma, n)$-everlasting robust predictor (as in Definition 3.1) and it is (ε, δ)-private (as in Definition 2.3).*

Equipped with our new definition of PERP, we observe that the construction of Naor et al. (2023) extends from PEP to PERP, at the cost of inflating the sample complexity by roughly a $\frac{1}{\alpha\gamma}$ factor. This is captured by the following observation; see Appendix A for more details.

Observation 3.3. *For every concept class C and every $\alpha, \beta, \gamma, \varepsilon, \delta$ there is an $(\alpha, \beta, \gamma, \varepsilon, \delta, n)$-PERP for C where $n = \tilde{O}\left(\frac{\text{VC}^2(C)}{\alpha^2 \cdot \gamma \cdot \varepsilon^2} \cdot \text{polylog}\left(\frac{1}{\beta\delta}\right)\right)$. The algorithm is not computationally efficient.*

3.2. Truly everlasting private prediction

As we mentioned in the introduction, current constructions for PEP exhibit sample complexity that scale as $\text{polylog}(\frac{1}{\delta})$. In addition, the definition of differential privacy is only considered adequate in this case provided that $\delta \ll \frac{1}{T}$, which means that current constructions for PEP are not "truly everlasting" as they require a training set whose size scales with the bound on the number of queries T. We now formalize this discussion using the following definition.

Definition 3.4 (Leakage attack)**.** *Let $\mathcal{A}$ be an (offline) algorithm that operates on a dataset $D \in [0,1]^T$. We say that $\mathcal{A}$ is* leaking *if when running it on a uniformly random dataset D, its outcome can be post-processed to identify a point y that with probability at least 1/2 satisfies $y \in D$.*

See Appendix A for an extension of this definition to online/interactive algorithms, such as prediction oracles.

Fact 3.5. *Let $\mathcal{A}$ be an algorithm that takes a dataset $D \in [0,1]^T$ and outputs every input point independently with probability δ. This algorithm satisfies $(0, \delta)$-DP. Furthermore, if $T \geq \frac{1}{\delta}$, then $\mathcal{A}$ is leaking.*

We put forward a simple twist to the privacy definition that allows us to exclude this undesired behaviour, without inflating the sample complexity of our algorithms by a $\text{polylog}(T)$ factor. Specifically, we require δ to *decrease over time*. Intuitively, instead of allowing every record i to be leaked with probability δ, we allow the ith record to be leaked with probability at most $\boldsymbol{\delta}(i)$, such that $\sum_i \boldsymbol{\delta}(i) \triangleq \delta^*$. In the context of PEP, we show that the sample complexity only needs to grow with $\frac{1}{\delta^*}$ and not with $\max_i\{\frac{1}{\boldsymbol{\delta}(i)}\}$. This is a significant improvement: While $\max_i\{\frac{1}{\boldsymbol{\delta}(i)}\}$ must be larger than T to prevent the leakage attack mentioned above, this is not the case with $\frac{1}{\delta^*}$ which can be taken to be a small constant. This still suffices in order to ensure that the algorithm is *not* leaking.

As a warmup, let us examine this in the context of the standard (offline) definition of differential privacy, i.e., in the context of Definition 1.1.

Definition 3.6. *Let $\varepsilon \geq 0$ be a fixed parameter and let $\boldsymbol{\delta} : \mathbb{N} \to [0,1]$ be a function. Let $\mathcal{A}$ be a randomized algorithm whose input is a dataset. Algorithm $\mathcal{A}$ is* $(\varepsilon, \boldsymbol{\delta})$-DP *if for any index i, any two datasets D, D' that differ on the ith entry, and any event F it holds that* $\Pr[\mathcal{A}(D) \in F] \leq e^{\varepsilon} \cdot \Pr[\mathcal{A}(D') \in F] + \boldsymbol{\delta}(i)$.

We now turn to the adaptive setting, as is required by the PEP model. Intuitively, the complication is that, unlike in the non-adaptive case, the index of the challenge round is a *random variable*, determined by the adversary during runtime. A direct approach for handling with this is to require that a similar definition holds for all conditioning on the index of the challenge round. This is captured in the following definition, where we use the terminology of Definition 2.3 (PEP's privacy definition), and use r^* to denote the challenge round, i.e., the round r in which $c_r{=}1$.

Definition 3.7. *Let $\varepsilon \geq 0$ be a fixed parameter and let $\boldsymbol{\delta} : \mathbb{N} \to [0,1]$ be a function. A prediction oracle $\mathcal{A}$ is a* $(\varepsilon, \boldsymbol{\delta})$-private *if for every adversary $\mathcal{B}$, every $T \in \mathbb{N}$, and every $i \in [T]$, conditioned on $r^* = i$ then the random variables $\text{View}^0_{\mathcal{B},T}$ and $\text{View}^1_{\mathcal{B},T}$ (defined in Figure 1) are $(\varepsilon, \boldsymbol{\delta}(i))$-indistinguishable.*

Note that this strictly generalizes Definition 2.3. Indeed, by taking $\boldsymbol{\delta}$ to be fixed, i.e., $\boldsymbol{\delta}(i) = \delta^*$ for all i, we get that $\text{View}^0_{\mathcal{B},T}$ and $\text{View}^1_{\mathcal{B},T}$ are (ε, δ^*)-indistinguishable (without the conditioning).[3] So Definition 3.7 is not weaker than Definition 2.3. The other direction is not true. Specifically, consider an algorithm $\mathcal{A}$ that with probability δ declares (upon its instantiation) that it is going to publish the first record in the clear. Otherwise the algorithm publishes nothing. This is typically something that would not be considered as a "privacy violation", as this catastrophic event happens only with probability δ. Indeed, this algorithm would satisfy Definition 2.3. However, with Definition 3.7, an adaptive attacker might decide to pose its challenge on the first input *if and only if* the algorithm has issued such a declaration. Otherwise the attacker never poses its challenge on the first record. In this case, in the conditional space where $r^* = 1$, we have that the attacker succeeds *with probability one*, and so this algorithm cannot satisfy Definition 3.7. This behavior makes Definition 3.7 somewhat harder to work with. We propose the following relaxed variant of Definition 3.7 that still rules out leaking algorithms.

Definition 3.8. *Let $\varepsilon \geq 0$ be a fixed parameter and let $\boldsymbol{\delta} : \mathbb{N} \to [0,1]$ be a function. A prediction oracle $\mathcal{A}$ is a* $(\varepsilon, \boldsymbol{\delta})$-private *if for every adversary $\mathcal{B}$, every $T \in \mathbb{N}$, every $i \in [T]$, and every event F it holds that* $\Pr[\textit{View}^0_{\mathcal{B},T} \in F \text{ and } r^* = i] \leq e^{\varepsilon} \cdot \Pr[\textit{View}^1_{\mathcal{B},T} \in F \text{ and } r^* = i] + \boldsymbol{\delta}(i)$, *and vice versa.*

This definition is strictly weaker (provides less privacy) than Definition 3.7. In particular, similarly to the calculation in Footnote 3, when taking $\boldsymbol{\delta} \equiv \delta^*$ it only implies $(\varepsilon, \sum_i \delta^*)$-DP rather than (ε, δ^*)-DP. Nevertheless, when $\sum_i \boldsymbol{\delta}(i) \ll 1$, this definition still prevents the algorithm from leaking, even if T is much larger than $(\sum_i \boldsymbol{\delta}(i))^{-1}$. This is captured by the following observation; see Appendix A for the proof.

Observation 3.9. *If $\mathcal{A}$ is $(\varepsilon, \boldsymbol{\delta})$-private (as in Definition 3.8) where $\sum_i \boldsymbol{\delta}(i) < \frac{1}{8}$, then $\mathcal{A}$ is not leaking.*

4. New Constructions for PEP

In this section we present our new PEP constructions. Our constructions outperform the generic construction of Naor et al. (2023) on three aspects: (1) our constructions are computationally efficient; (2) our constructions exhibit sample complexity *linear* in the VC dimension rather than quadratic; and (3) our constructions achieve significantly stronger robustness guarantees than our robust extension to the construction of Naor et al. (2023). We first introduce additional preliminaries that are needed for our constructions.

4.1. Additional preliminaries

The Reorder-Slice-Compute (RSC) paradigm. Consider a case in which we want to apply several privacy-preserving computations to our dataset, where each computation is applied to a *disjoint part* (slice) of the dataset. If the slices are selected in a data-independent way, then a straightforward analysis shows that our privacy guarantees do not deteriorate with the number of slices. The following algorithm (Algorithm RSC) extends this to the more complicated case where we need to select the slices adaptively in a way that depends on the outputs of prior steps.

Algorithm 1 RSC **(Cohen et al., 2023)**

Input: Dataset $D \in X^n$ and parameters $\tau, \varepsilon, \delta$.

For $i = 1, \ldots, \tau$ do:

1. Receive a number $m_i \in \mathbb{N}$, an ordering $\prec^{(i)}$ over X, and an (ε, δ)-DP protocol $\mathcal{A}_i$.
2. $\hat{m}_i \leftarrow m_i + \texttt{Geom}(1 - e^{-\varepsilon})$ % Here Geom denotes the geometric distribution.
3. $S_i \leftarrow$ the largest $\hat{m}_i$ elements in D under $\prec^{(i)}$
4. $D \leftarrow D \setminus S_i$
5. Instantiate $\mathcal{A}$ on S_i and provide external access to it (via its query-answer interface).

[3] To see this, note that for any event F we have $\Pr[\text{View}^0_{\mathcal{B},T} \in F] = \sum_i \Pr[r^* = i] \cdot \Pr[\text{View}^0_{\mathcal{B},T} \in F | r^* = i] \leq e^{\varepsilon} \left(\sum_i \Pr[r^* = i] \cdot \Pr[\text{View}^1_{\mathcal{B},T} \in F | r^* = i]\right) + \delta^* = e^{\varepsilon} \cdot \Pr[\text{View}^1_{\mathcal{B},T} \in F] + \delta^*$.

Theorem 4.1 (Cohen et al. (2023)). *For every $\hat{\delta} > 0$, Algorithm* `RSC` *is $(O(\varepsilon \log(1/\hat{\delta})), \hat{\delta} + 2\tau\delta)$-DP.*

Algorithm `Stopper`. The following algorithm is a special case of the classical `AboveThreshold` algorithm of Dwork et al. (2009), also known as the Sparse Vector Technique. It allows to continually monitor a bit stream, and to indicate when the number of ones in this stream (roughly) crosses some threshold.

Algorithm 2 `Stopper`

Input: Privacy parameters ε, δ, threshold t, and a dataset D containing input bits.

1. **In update time:** Obtain $x \in \{0,1\}$ and add x to D.
2. **In query time:**
   (a) Let $\widetilde{\text{sum}}_i \leftarrow \text{Lap}\left(\frac{8}{\varepsilon}\log(\frac{2}{\delta})\right) + \sum_{x \in D} x$
   (b) If $\widetilde{\text{sum}}_i \geq t$ then output $\top$ and HALT
   (c) Else output $\perp$ and CONTINUE

Theorem 4.2. *Algorithm* `Stopper` *is (ε, δ)-DP w.r.t. the input bits (both in the initial dataset D and in update times).*

Algorithm `BetweenThresholds`. Algorithm `BetweenThresholds` allows for testing a sequence of low-sensitivity queries to learn whether their values are (roughly) above or below some predefined thresholds.

Algorithm 3 `BetweenThresholds` (Bun et al., 2017)

Input: Dataset $S \in X^*$, privacy parameters ε, δ, number of "medium" reports k, and thresholds $t_l < t_h$.

1. Let $c = 0$
2. In each round i, receive a sensitivity-1 query f_i and do the following:
   (a) Let $\hat{f}_i \leftarrow f_i(S) + \text{Lap}\left(\frac{4}{\varepsilon}\sqrt{k \log(\frac{2}{\delta})}\right)$
   (b) If $\hat{f}_i < t_l$ then output "low"
   (c) Else if $\hat{f}_i > t_h$ then output "high"
   (d) Else output "medium" and set $c \leftarrow c + 1$. If $c = k$ then HALT.

The properties of this algorithm are specified in the following theorem.

Theorem 4.3 (Bun et al. (2017); Cohen & Lyu (2023)). *Suppose that $k \geq 4\log(\frac{2}{\delta})$ and that $t_h - t_l \geq \frac{16}{\varepsilon}\sqrt{k \log(\frac{2}{\delta})}$. Algorithm* `BetweenThresholds` *is (ε, δ)-DP.*

Algorithm `ChallengeBT`. Note that algorithm `BetweenThresholds` halts exactly after the kth time that a "medium" answer is returned. For our application we will need a variant of this algorithm in which the halting time leaves some ambiguity as to the exact number of "medium" answers obtained so far, and to whether the last provided answer was a "medium" answer or not. We introduce such a simple variant by combining Algorithm `BetweenThresholds` with Algorithm `Stopper`. The construction is given in Algorithm `ChallengeBT`.

Algorithm 4 `ChallengeBT`

Input: Dataset $S \in X^*$, privacy parameters ε, δ, number of "medium" reports k where $k \geq 4\log(\frac{4}{\delta})$, thresholds t_l, t_h satisfying $t_h - t_l \geq \frac{32}{\varepsilon}\sqrt{k \log(\frac{4}{\delta})}$, a bound on the number of steps T, and an adaptively chosen stream of queries $f_i : X^* \to \mathbb{R}$ with sensitivity 1.

1. Instantiate `Stopper` with privacy parameters (ε, δ) and threshold $t = k$ on the empty dataset.
2. Instantiate a modified version of `BetweenThresholds` on S with parameters $(\varepsilon, \frac{\delta}{2}), k' = k + \frac{8}{\varepsilon}\log(\frac{2}{\delta})\log(\frac{T}{\delta}), t_l, t_h$. The modification is that the algorithm never halts.
3. Denote Flag $= 1$.
4. Support the following queries:
   (a) **Stopping query:** Set Flag $= 1$. Query `Stopper` to get an answer a, and output a. If `Stopper` halted during this query then HALT the execution.
   (b) **BT query:** Obtain a sensitivity-1 function f. If Flag $= 0$ then ignore this f and do nothing. Otherwise do the following:
      i. Set Flag $= 0$.
      ii. Feed f to `BetweenThresholds` and receive an answer a. Output a.
      iii. If $a =$"medium" then update `Stopper` with 1 and otherwise update it with 0.

The following theorem captures the privacy properties of algorithm `ChallengeBT`; see Appendix B for the proof.

Theorem 4.4. *`ChallengeBT` is (ε, δ)-DP w.r.t. the input dataset S.*

4.2. Axis-aligned rectangles

We are now ready to present our construction for axis-aligned rectangles in the Euclidean space $\mathbb{R}^d$. A concept in this class could be thought of as the product of d intervals, one on each axis. Formally,

Definition 4.5 (Axis-Aligned Rectangles). *Let $d \in \mathbb{N}$ denote the dimension. For every $w = (a_1, b_1, \ldots, a_d, b_d) \in \mathbb{R}^{2d}$ define the concept $\text{rec}_w : \mathbb{R}^d \to \{0,1\}$ as follows. For $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$ we have $\text{rec}_w(x) = 1$ iff for every $i \in [d]$ it holds that $x_i \in [a_i, b_i]$. Define the class of all axis-aligned rectangles over $\mathbb{R}^d$ as $\texttt{REC}_d = \{\text{rec}_w : w \in \mathbb{R}^{2d}\}$.*

4.2.1. A SIMPLIFIED OVERVIEW OF OUR CONSTRUCTION

The generic construction of Naor et al. (2023) is based on the celebrated "sample and aggregate" technique (Nissim et al., 2007). Specifically, it maintains $k \gg 1$ independent hypothesis, and answers every query using a noisy majority vote among these k hypotheses. After enough queries have been answered, then Naor et al. (2023) re-trains (at least) k new hypotheses, treating the previously answered queries as the new training set. The downside with this is that we need to pay (in sample complexity) for these k independent hypotheses. This is the bottleneck in the construction of Naor et al. (2023) which resulted in a sample complexity that is *quadratic* in the VC dimension. We show that this can be avoided for rectangles in high dimensions. Intuitively, our gain comes from *not* maintaining many complete hypotheses, but rather only a single "evolving" hypothesis.

To illustrate this, let us consider the case where $d = 1$. That is, for some $a \leq b$, the target function c^* is an *interval* of the form $c^*(x) = 1$ iff $a \leq x \leq b$. Let us also assume that the target distribution $\mathcal{D}$ is such that $\Pr_{x\sim\mathcal{D}}[c^*(x) = 1] \gg 0$, as otherwise the all 0 hypothesis would be a good solution.

Now suppose we get a sample S from the underlying distribution $\mathcal{D}$, labeled by c^*, and let $S_{\text{left}} \subseteq S$ and $S_{\text{right}} \subseteq S$ be two datasets containing the $m \approx \frac{1}{\alpha\varepsilon}$ smallest positive points in S and the m largest positive points in S (respectively). We can use $S_{\text{left}}, S_{\text{right}}$ to privately answer queries as follows: Given a query $x \in \mathbb{R}$, if x is smaller than (almost) all of the points in S_{left}, or larger than (almost) all of the points in S_{right}, then we label x as 0. Otherwise we label it as 1. By the Chernoff bound, this would have error less than α. Furthermore, this could be done privately using the "sparse vector technique" (or the `BetweenThresholds` algorithm). This has merit as, by the properties of `BetweenThresholds`, in the privacy analysis we would only need to account for queries x that are "deep" inside S_{left} or S_{right}, which would hopefully not happen too often. However, recall that we aim to operate in the adversarial case, where the vast majority of the queries can be adversarial. Thus, it is quite possible that *most* of the queries would indeed incur such a privacy loss. Nevertheless, we show that if x generated a privacy loss w.r.t. (say) S_{left}, then informally, we can use x to replace one of the points in S_{left}, as it is "deep" inside it. So essentially every time we incur a privacy loss w.r.t. one of the "boundary datasets" we maintain, we get a new point to add to them in exchange. We show that this balances out the privacy loss all together.

More precisely, it is actually *not* true that every query that generates a privacy loss w.r.t. some "boundary dataset" could be added to it. The reason is that we want our construction to be "everlasting", supporting an unbounded number of queries. In this regime the `BetweenThresholds` algorithm would sometimes err and falsely identify a query x as being "deep" inside one of our "boundary datasets". There would not be too many such cases, so that we can tolerate the privacy loss incurred by such errors, but we do not want to add unrelated/adversarial queries to the "boundary datasets" we maintain as otherwise they will get contaminated over time. In the full construction we incorporate measures to protect against this. Our formal construction is given in algorithm `RectanglesPERP`.

4.2.2. PRIVACY ANALYSIS OF RECTANGLESPERP

Theorem 4.6. *`RectanglesPERP` is $(\varepsilon, \boldsymbol{\delta})$-private, as in Definition 3.8, where $\sum_i \boldsymbol{\delta}(i) \leq \delta^*$.*

Proof. Using the notations of `RectanglesPERP`, we define the function $\boldsymbol{\delta} : \mathbb{N} \to [0, 1]$ where $\boldsymbol{\delta}(i) = \delta_{p(i)}$ and where $p(i)$ denote the phase to which i belongs. When the time i or the phase p is clear from the context, we write δ_p instead of $\delta_{p(i)} = \boldsymbol{\delta}(i)$ to simplify the notations. Fix $T \in \mathbb{N}$, fix an adversary $\mathcal{B}$ for interacting with `RectanglesPERP` as in Figure 1, and consider the random variables $\text{View}^0_{\mathcal{B},T}$ and $\text{View}^1_{\mathcal{B},T}$. Now fix an index $i \in [T]$ and let $p = p(i)$. We need to show that for every event F it holds that $\Pr[\text{View}^0_{\mathcal{B},T} \in F \text{ and } c_i = 1] \approx_{(\varepsilon,\delta_p)} \Pr[\text{View}^1_{\mathcal{B},T} \in F \text{ and } c_i = 1]$. To simplify notations, for $b \in \{0, 1\}$ let us define the random variable $\overline{\text{View}}^b_{\mathcal{B},T}$ which is equal to $\text{View}^b_{\mathcal{B},T}$ if $c_i = 1$, and equal to $\perp$ otherwise. This is well-defined as c_i is included in the view of the adversary. Using this notation, our goal is to show that $\overline{\text{View}}^0_{\mathcal{B},T} \approx_{(\varepsilon,\delta_p)} \overline{\text{View}}^1_{\mathcal{B},T}$. To show this, we leverage the simulation proof-technique of Cohen et al. (2023). Specifically, we will show that $\overline{\text{View}}^b_{\mathcal{B},T}$ can be perfectly simulated by post-processing the outcome of an (ε, δ_p)-DP computation on the bit b. To this end, we describe two interactive mechanisms: a simulator `Sim` and a helper `Helper`. The goal of the simulator is to simulate $\mathcal{B}$'s view in the game specified in Figure 1. This simulator does not know the value of the bit b. Still, it executes $\mathcal{B}$ and attempts to conduct as much of the interaction as it can without knowing b. Only when it is absolutely necessary for the simulation to remain faithful, the simulator will pose queries to `Helper` who responds with the result of a DP computation on b. We will show the following two statements, which imply the theorem by closure to post-processing: (1) `Helper` is (ε, δ_p)-DP w.r.t. the bit b; and (2) `Sim` perfectly simulates $\overline{\text{View}}^b_{\mathcal{B},T}$.

The simulator `Sim` begins by instantiating the adversary $\mathcal{B}$ (who is expecting to interact with `RectanglesPERP` through the game specified in Figure 1). We analyze the execution in cases, according to whether $i = 0$ or not. At any case, if $\mathcal{B}$ poses its challenge in any round other than i then

Algorithm 5 `RectanglesPERP`

Initial input: Labeled dataset $S \in (\mathbb{R}^d \times \{0,1\})^n$ and parameters $\varepsilon, \delta^*, \alpha, \beta, \gamma$.

Parameters: For $p \in \mathbb{N}$ let $\alpha_p = \frac{\alpha}{2^p}$, and $\beta_p = \frac{\beta}{2^p}$, and $\delta_p = \tilde{\Theta}\left(\frac{\delta^* \gamma \alpha \varepsilon^2 \beta}{d \cdot 8^p}\right)$, and $m_p = \frac{1}{\varepsilon^2}\,\mathrm{polylog}\left(\frac{d}{\alpha_p \cdot \beta_p \cdot \gamma \cdot \varepsilon \cdot \delta_p}\right)$, $k_p = 2m_p$, $t_p = \Theta\left(\frac{d \cdot m_p}{\gamma \cdot \alpha_p}\right)$, $\Delta_p = \frac{\sqrt{k_p}}{\varepsilon}\,\mathrm{polylog}(\frac{d \cdot t_p}{\beta_p \cdot \delta_p})$.

1. Let $p = 1$. % p denotes the current "phase".
2. Instantiate algorithm `RSC` on S.
3. For $j = 1, 2, \ldots, d$: Use `RSC` to run a copy of `ChallengeBT` on the $\approx m_p$ positive examples in S with largest jth coordinate, and another copy on the $\approx m_p$ positive examples in S with smallest jth coordinate. Each execution with k_p as the number of "medium" answers, with $\left(\frac{\varepsilon}{\log(1/\delta_p)}, \frac{\delta_p}{d}\right)$ as the privacy parameters, and with thresholds $t_l = \Delta_p$ and $t_h = 2\Delta_p$. Denote these executions as $\texttt{Right}_j$ and $\texttt{Left}_j$, respectively.
4. Let $D = \emptyset$ and for every $j \in [d]$ let $D_j^{\text{left}} = D_j^{\text{right}} = \emptyset$.
5. For time $i = 1, 2, 3, \ldots$ do the following:
   (a) For all $j \in [d]$, pose a "stopping query" to $\texttt{Left}_j$ and to $\texttt{Right}_j$.
   (b) If any of $\texttt{Left}_j$ or $\texttt{Right}_j$ has halted during Step 5a then re-execute it on (a copy of) the corresponding dataset D_j^{left} or D_j^{right}, respectively, and then empty this dataset.
   (c) Obtain an unlabeled query point $x_i \in X$
   (d) Set $\hat{y}_i \leftarrow 0$.
   (e) If $x_i = \perp$ the GOTO Step 5j.
   (f) For $j = 1, 2, \ldots, d$ do:
      i. Query $\texttt{Left}_j$ for the number of points in its dataset whose jth coordinate are bigger than $x_i[j]$, and obtain an answer a. If a ="high" then GOTO Step 5h. If a="medium" then add x_i to D_j^{left} and GOTO Step 5i.
      ii. Query $\texttt{Right}_j$ analogously.
   (g) Set $\hat{y}_i \leftarrow 1$
   (h) Add $(x_i, \hat{y}_i)$ to D
   (i) Output the predicted label $\hat{y}_i$
   (j) If $i = \sum_{q \in [p]} t_q$ then: % End of current phase.
      i. Let $p \leftarrow p + 1$.
      ii. Stop all copies of $\texttt{Left}_j$ and $\texttt{Right}_j$, and instantiate algorithm `RSC` on D.
      iii. Use `RSC` to obtain new executions $\texttt{Right}_j$ and $\texttt{Left}_j$ for $j \in [d]$ (as in Step 3).
      iv. Let $D = \emptyset$ and for every $j \in [d]$ let $D_j^{\text{left}} = D_j^{\text{right}} = \emptyset$.

the simulator outputs $\perp$, as in the definition of $\overline{\text{View}}^b_{\mathcal{B},T}$.

Easy case: $i = 0$ and $c_0 = 1$. That is, the adversary $\mathcal{B}$ specifies two *neighboring* datasets $S^0 \neq S^1$ in the beginning of the execution. In this case, `Sim` asks `Helper` to execute Step 3 of `RectanglesPERP`. That is, to execute `RSC` on S^b and to provide `Sim` with the querying interface to the resulting instantiations of `ChallengeBT`. This preserves (ε, δ_p)-DP by the privacy guarantees of `RSC`. The continuation of the execution can be done without any further access to the bit b. That is, `Sim` continues to query the instantiations of `ChallengeBT` produces by `Helper`, and whenever a new instantiation is required it can run it itself since there are no more differences between the execution with $b = 0$ or with $b = 1$.

More involved case: $i \geq 1$ and $c_i = 1$. In this case, `Sim` can completely simulate the execution of `RectanglesPERP` until the ith round. In particular, at the beginning of the ith round, the simulator is running (by itself) the existing copies of `ChallengeBT`, referred to as $\texttt{Left}_j$ and $\texttt{Right}_j$ in the code of the algorithm. During the ith round, the simulator obtains from the adversary the query point x_i, but as it does not know the bit b, it does not know who among $\{x_i, \perp\}$ needs to be fed to algorithm `RectanglesPERP`. Either way, the simulator can execute Steps 5a till 5d, as they do not depend on the current query. Next, the simulator runs Step 5f *assuming that the query is* x_i. We proceed according to the following two sub-cases:

Sub-case 1: A "medium" answer is not encountered during the simulation of Step 5f. Recall that a BT-query to algorithm `ChallengeBT` only changes its inner state if the answer is "medium". Hence, in this case, the execution of Step 5f with x_i did not affect the inner states of any of the executions of `ChallengeBT`, and the simulator can continue using them. However, the executions with x_i and with $\perp$ are still not synchronized, as x_i would be added to the dataset D in Step 5h whereas $\perp$ would not. This means that from this moment until a new phase begins (in Step 5j), there are two possible options for the dataset D: either with x_i or without it (depending on b). Thus, the next time that this dataset needs to be used by algorithm `RectanglesPERP`, which happens in Step 5(j)iii, the simulator asks `Helper` to conduct this step for it, similarly to the easy case. This preserves (ε, δ_p)-DP. From that moment on, the simulator does not need to access the `Helper` anymore.

Sub-case 2: A "medium" answer is encountered, say when querying $\texttt{Left}_j$. In this case, the executions with x_i and with $\perp$ differ on the following points:

- The executions differ in the inner state of $\texttt{Left}_j$. Specifically, let m denote the number of "medium" answers obtained from the current copy of $\texttt{Left}_j$ before time i. After time i the internal state of $\texttt{Left}_j$ differs be-

tween the two executions in that if $b = 0$ then the copy of `Stopper` inside $\texttt{Left}_j$ has a dataset with m ones, while if $b = 1$ then it has $m + 1$ ones. The execution of `BetweenThresholds` inside $\texttt{Left}_j$ is not affected. Thus, to simulate this, the simulator `Sim` asks `Helper` to instantiate a copy of algorithm `Stopper` on a dataset containing $m + b$ ones. This satisfies (ε, δ_p)-DP and allows `Sim` to continue the simulation of $\texttt{Left}_j$ exactly.

- The executions also differ in that x_i would be added to the dataset D_j^{left} during Step 5f while $\perp$ would not. In other words, from this moment until D_j^{left} is used (in Step 5b) or erased (in Step 5(j)iv), there are two options for this dataset: either with x_i or without it. Hence, if `Sim` needs to execute `ChallengeBT` on this dataset (in Step 5b) then instead of doing it itself it ask `Helper` to do it. This also satisfies (ε, δ_p)-DP. From that moment on, `Sim` does not need to access the `Helper` anymore.

Overall, we showed that `Sim` can perfectly simulate the view of the adversary $\mathcal{B}$ while asking `Helper` to execute at most two protocols, each satisfying (ε, δ_p)-DP. This shows that `RectanglesPERP` is $(\varepsilon, \boldsymbol{\delta})$-private (follows from parallel composition). The fact that $\sum_i \boldsymbol{\delta}(i) = \sum_p t_p \cdot \delta_p$ converges to at most δ^* follows from our choices for δ_p and t_p. □

This concludes the privacy analysis of `RectanglesPERP`. See Appendix B for the utility analysis, as well as for the construction for decision-stumps.

Acknowledgements

The author would like to thank Amos Beimel for helpful discussions about Definitions 3.7 and 3.8. This work was partially supported by the Israel Science Foundation (grant 1871/19) and by the Blavatnik Family foundation.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Alon, N., Bun, M., Livni, R., Malliaris, M., and Moran, S. Private and online learnability are equivalent. *J. ACM*, 69 (4):28:1–28:34, 2022.

Bassily, R., Thakurta, A. G., and Thakkar, O. D. Model-agnostic private learning. In *NeurIPS*, 2018.

Bun, M. and Zhandry, M. Order-revealing encryption and the hardness of private learning. In *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part I*, pp. 176–206, 2016.

Bun, M., Nissim, K., Stemmer, U., and Vadhan, S. P. Differentially private release and learning of threshold functions. In *FOCS*, pp. 634–649, 2015.

Bun, M., Steinke, T., and Ullman, J. Make up your mind: The price of online queries in differential privacy. In *Proceedings of the twenty-eighth annual ACM-SIAM symposium on discrete algorithms*, pp. 1306–1325. SIAM, 2017.

Cohen, E. and Lyu, X. The target-charging technique for privacy accounting across interactive computations. In *NeurIPS*, 2023.

Cohen, E., Lyu, X., Nelson, J., Sarlós, T., and Stemmer, U. Optimal differentially private learning of thresholds and quasi-concave optimization. In *STOC*, pp. 472–482. ACM, 2023.

Dagan, Y. and Feldman, V. PAC learning with stable and private predictions. In *COLT*, 2020.

Dwork, C. and Feldman, V. Privacy-preserving prediction. In *COLT*, 2018.

Dwork, C., McSherry, F., Nissim, K., and Smith, A. Calibrating noise to sensitivity in private data analysis. In *TCC*, pp. 265–284, 2006.

Dwork, C., Naor, M., Reingold, O., Rothblum, G. N., and Vadhan, S. P. On the complexity of differentially private data release: efficient algorithms and hardness results. In *STOC*, pp. 381–390, 2009.

Kasiviswanathan, S. P., Lee, H. K., Nissim, K., Raskhodnikova, S., and Smith, A. What can we learn privately? *SIAM J. Comput.*, 40(3):793–826, 2011.

Nandi, A. and Bassily, R. Privately answering classification queries in the agnostic PAC model. In *ALT*, 2020.

Naor, M., Nissim, K., Stemmer, U., and Yan, C. Private everlasting prediction. In *NeurIPS*, 2023.

Nissim, K., Raskhodnikova, S., and Smith, A. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pp. 75–84, 2007.

van der Maaten, L. and Hannun, A. Y. The trade-offs of private prediction. *CoRR*, abs/2007.05089, 2020.

A. Missing Details from Section 3

A.1. Details regarding Observation 3.3

Observation 3.3 follows from essentially the same analysis that of Naor et al. (2023). Here we only flesh out the differences, highlighting the reason for the $\frac{1}{\alpha\gamma}$ blowup to the sample complexity. Informally, the generic construction of Naor et al. (2023) is based on the following subroutine:

1. Take a training set S containing n labeled samples.
2. Partition S into $T \approx \frac{\alpha n}{\mathrm{VC}(C)}$ chunks of size $\approx \frac{\mathrm{VC}(C)}{\alpha}$ each.
3. For each chunk $i \in [T]$, identify a hypothesis $h_i \in C$ that agrees with the ith chunk.
4. For $\approx \frac{\varepsilon^2 T^2}{\alpha}$ steps: take an unlabeled query x and label it using a noisy majority vote among the T hypotheses.
   % The reason that we can support $\approx \frac{\varepsilon^2 T^2}{\alpha}$ queries is as follows. Assuming that each of the T hypotheses has generalization error less than α, then w.h.p., except for $O(\alpha)$-fraction of the queries, in all other queries there will be a strong consensus among the T hypotheses to the extent that answering these queries would come "for free" in terms of the privacy analysis. So we only need to "pay" for about $\varepsilon^2 T^2$ queries, which is standard using composition theorems.
5. At the end of this process, we have $\approx \frac{\varepsilon^2 T^2}{\alpha}$ new labeled examples, which is more than n provided that $n \gtrsim \frac{\mathrm{VC}^2(C)}{\alpha \cdot \varepsilon^2}$. This allows us to repeat the same subroutine with these new labeled examples as our new training set.
   % This is actually not accurate, as this new training set contains errors while the original training set did not. In the full construction additional steps are taken to ensure that the error does not accumulate too much from one phase to the next, making their construction inefficient.

Based on this, Naor et al. (2023) presented a generic construction for PEP exhibiting sample complexity $n = \tilde{O}\left(\frac{\mathrm{VC}^2(C)}{\alpha \cdot \varepsilon^2} \cdot \mathrm{polylog}\left(\frac{1}{\beta\delta}\right)\right)$. Now suppose that we would like to withstand $(1-\gamma)$-fraction of adversarial queries. First, this prevents us from optimizing the number of supported queries in Step 4 above, and we could only answer $\approx \varepsilon^2 T^2$ queries. The reason is that with adversarial queries we can no longer guarantee that the vast majority of the queries would be in consensus among the hypotheses we maintain. Second, only a γ-fraction of these queries would be "true samples", and we would need to ensure that the number of such new "true samples" is more than what we started with. This translate to a requirement that $n \gtrsim \frac{\mathrm{VC}^2(C)}{\alpha^2 \cdot \gamma \cdot \varepsilon^2}$.

A.2. An extension of Definition 3.4

Definition A.1 (Leakage attack against interactive algorithms)**.** *Let $\mathcal{A}$ be an (interactive) prediction oracle, and consider an adversary $\mathcal{B}$ that interacts with $\mathcal{A}$ for T time steps, and adaptively decides on $k \leq T$ steps at which the algorithm gets a random uniform sample from $[0,1]$ without the adversary learning these points directly. We refer to these points as "challenge points". The adversary arbitrarily chooses all other inputs to the algorithm and sees all of its outputs (the predictions it returns). We say that $\mathcal{A}$ is* leaking *if there is such an adversary $\mathcal{B}$ that, with probability at least 1/2, at the end of the execution outputs a point y that is identical to one of the challenge points.*

A.3. Proof of Observation 3.9

Proof of Observation 3.9. Assume towards contradiction that $\mathcal{A}$ is leaking, and let $\mathcal{B}$ be an appropriate adversary (as in Definition A.1). Then, there must exist an index i such that with probability at least $4\boldsymbol{\delta}(i)$ it holds that (1) this coordinate is chosen to be a challenge point, and (2) the adversary recovers this point. Otherwise, by a union bound, the probability of $\mathcal{B}$ succeeding in its leakage attack would be at most $\sum_i 4\boldsymbol{\delta}(i) < \frac{1}{2}$. Let i be such an index, and consider a modified adversary $\mathbb{B}$ that interacts with $\mathcal{A}$ (as in Figure 1) while simulating $\mathcal{B}$. More specifically, the adversary $\mathbb{B}$ first samples T uniformly random points $x_1, \ldots, x_T \in [0,1]$ and runs $\mathcal{B}$ internally. Whenever $\mathcal{B}$ specifies an input then $\mathbb{B}$ passes this input to $\mathcal{A}$. In rounds j where $\mathcal{B}$ asks to give $\mathcal{A}$ a random point, then $\mathbb{B}$ passes x_j to $\mathcal{A}$. Recall that (as in Figure 1) the adversary $\mathbb{B}$ needs to choose one round as its challenge round; this is chosen to be round i. In that round the adversary $\mathbb{B}$ does not get to see $\mathcal{A}$'s prediction but it still needs to pass the prediction to $\mathcal{B}$ in order to continue the simulation. It passes a random bit y_i to $\mathcal{B}$ as if it is the prediction obtained from $\mathcal{A}$ (in all other rounds, $\mathbb{B}$ passes the predictions obtained from $\mathcal{A}$ to $\mathcal{B}$). Now, if the bit b in the experiment specified by Figure 1 is 1, and if y_i is equal to the prediction returned by $\mathcal{A}$ for the ith bit (which happens with probability $1/2$), then this experiment perfectly simulates the interaction between $\mathcal{B}$ and $\mathcal{A}$. Hence, whenever $b = 1$ then at the end of the execution $\mathcal{B}$ guesses x_i with probability at least $2\boldsymbol{\delta}(i)$. On the other hand, if the bit b in the

experiment specified by Figure 1 is 0, then $\mathcal{B}$ gets no information about x_i and hence guesses it with probability exactly 0. This contradicts the definition of $(\varepsilon, \boldsymbol{\delta})$-privacy. □

B. Missing Details from Section 4

B.1. Proof of Theorem 4.4

Proof of Theorem 4.4. First note that if in Step 2 we were to execute `BetweenThresholds` without modifications, then the theorem would follow the privacy guarantees of `BetweenThresholds`, as `ChallengeBT` simply post-processes its outcomes. As we explain next, this modification introduces a statistical distance of at most $\frac{\delta}{2}$, and thus the theorem still holds. To see this, observe that algorithm `Stopper` tracks the number of "medium" answers throughout the execution. Furthermore, note that the value k' with which we instantiate algorithm `BetweenThresholds` is noticeably bigger than k (the threshold given to `Stopper`), so that with probability at least $1 - \frac{\delta}{2}$ algorithm `Stopper` halts *before* the k' time in which a "medium" answer is observed. When that happens, the modification we introduced to `BetweenThresholds` has no effect on the execution.[4] This completes the proof. □

B.2. Utility analysis of `RectanglesPERP`

Fix a target distribution $\mathcal{D}$ over $\mathbb{R}^d$ and fix a target rectangle c. Recall that c is defined as the intersection of d intervals, one on every axis, and denote these intervals as $\left\{[c_j^{\text{left}}, c_j^{\text{right}}]\right\}_{j\in[d]}$. That is, for every $x \in \mathbb{R}^d$ we have $c(x) = 1$ if and only if $\forall j \in [d]$ we have $c_j^{\text{left}} \leq x[j] \leq c_j^{\text{right}}$. We write $\texttt{RECTANGLE}(c) = \{x \in \mathbb{R}^d : c(x) = 1\} \subseteq \mathbb{R}^d$ to denote the positive region of c. We assume for simplicity that $\Pr_{x\sim\mathcal{D}}[c(x) = 1] > \alpha$ (note that if this is not the case then the all-zero hypothesis is a good solution).

Recall that throughout the execution of `RectanglesPERP` we maintain the "boundary datasets" D_j^{left} and D_j^{right} for $j \in [d]$. Ideally we would want to argue that from one phase to the next, these "boundary datasets" are *nested*. That is, we would want to argue that the projection of the current D_j^{left} on the jth axis is contained within the projection of the previous D_j^{left}. However, as we mentioned in Section 4.2.1, this is not quite true. Instead, we will show that these "boundary datasets" are *almost* nested in the sense that from one phase to the next they might expand beyond the previous "boundary datasets", but this expansion will cover only a negligible mass of the underlying distribution, which will be acceptable. These "expansion regions" are formalized by the following definition.

Definition B.1. *For $p \in \mathbb{N}$ and $j \in [d]$ we define $\texttt{Strip}_{p,j}^{\text{left}}$ and $\texttt{Strip}_{p,j}^{\text{right}}$ according to Algorithm 6.*

Algorithm 6 Defining sub-regions of $\texttt{RECTANGLE}(c)$

1. Denote $R = \texttt{RECTANGLE}(c)$. For $p \in \mathbb{N}$ we denote $\alpha_p = \alpha/2^p$ and $\beta_p = \beta/2^p$, where α and β are the utility parameters.
2. For $p = 1, 2, 3, \ldots$
   (a) For axis $j = 1, 2, 3, \ldots, d$:
   - Define $\texttt{Strip}_{p,j}^{\text{left}}$ to be the rectangular strip along the inside left side of R which encloses exactly weight α_p/d under $\mathcal{D}$.
   - Set $R \leftarrow R \setminus \texttt{Strip}_{p,j}^{\text{left}}$
   - Define $\texttt{Strip}_{p,j}^{\text{right}}$ to be the rectangular strip along the inside right side of R which encloses exactly weight α_p/d under $\mathcal{D}$.
   - Set $R \leftarrow R \setminus \texttt{Strip}_{p,j}^{\text{right}}$

Remark B.2. *In Algorithm 6 we assume that the strips we define has weight* exactly α_p/d. *This might not be possible (e.g., if $\mathcal{D}$ has atoms), but this technicality can be avoided using standard techniques. For example, by replacing every sample $x \in \mathbb{R}$ with a pair (x, z) where z is sampled uniformly from $[0, 1]$. We ignore this issue for simplicity.*

[4] Formally, let E denote the even that all of the noises sampled by `Stopper` throughout the execution are smaller than $k' - k$. We have that $\Pr[E] \geq 1 - \delta/2$. Furthermore, conditioned on E, the executions with and without the modification are identical. This shows that this modification introduces a statistical distance of at most $\delta/2$.

The following notations will be convenient.

Definition B.3. *Let $A \subseteq \mathbb{R}^d$ and let $j \in [d]$. For a point $x \in \mathbb{R}$ we write $x \in_j A$ if the projection of A on the jth axis contains x. For a vector $x \in \mathbb{R}^d$ we write $x \in_j A$ if $x[j] \in_j R$. For a set of points (or vectors) S we write $S \subseteq_j A$ if for every $x \in S$ it holds that $x \in_j A$.*

Definition B.4. *Given an (interactive) algorithm $\mathcal{A}$ whose input is a dataset, we write $\mathrm{data}(\mathcal{A})$ to denote the dataset on which $\mathcal{A}$ was executed.*

The following event states that all the noises sampled throughout the execution are bounded.

Definition B.5. *For $p \in \mathbb{N}$, let $E(p)$ denote the following event w.r.t. the execution of `RectanglesPERP`:*

1. *All the geometric RV's sampled during phase p (via `RCS`) are at most $\frac{1}{\varepsilon}\log(\frac{1}{\delta_p})\log(\frac{4d}{\beta_p})$.*
2. *All the Laplace RV's sampled during phase p (via `ChallengeBT`) are at most Δ_p in absolute value, where $\Delta_p = \frac{4\log(1/\delta_p)}{\varepsilon}\sqrt{k_p \log(\frac{2d}{\delta_p})}\log\left(\frac{8dt_p}{\beta_p}\right)$.*

Lemma B.6. *For every $p \in \mathbb{N}$ we have $\Pr[E(p)] \geq 1 - \beta_p$.*

Proof. In the beginning of phase p we sample $2d$ geometric RV's (via the `RCS` paradigm) with parameter $\left(1 - e^{-\varepsilon/\log(1/\delta_p)}\right)$. By the properties of the geometric distribution, with probability at least $1 - \frac{\beta_p}{2}$, all of these samples are at most $\frac{1}{\varepsilon}\log(\frac{1}{\delta_p})\log(\frac{4d}{\beta_p})$.

Throughout phase p there are at most t_p time steps. In every time step we have (at most) 2 draws from the Laplace distribution in every copy of `ChallengeBT`, and there are 2d such copies. By the properties of the Laplace distribution, with probability at least $1 - \frac{\beta_p}{2}$, all of these samples are at most $\frac{4\log(1/\delta_p)}{\varepsilon}\sqrt{k_p \log(\frac{2d}{\delta_p})}\log\left(\frac{8dt_p}{\beta_p}\right)$ in absolute value. □

The following lemma is the key to the utility analysis of `RectanglesPERP`. In particular, it shows that the "boundary datasets" we maintain throughout the execution do not expand "too much".

Lemma B.7. *If the size of the initial labeled dataset satisfies $n = \Omega\left(\frac{d}{\alpha\cdot\varepsilon^2}\,\mathrm{polylog}(\frac{d}{\alpha\cdot\beta\cdot\gamma\cdot\varepsilon\cdot\delta^*})\right)$, then for every $p \in \mathbb{N}$, the following holds with probability at least $\left(1 - 2\sum_{q\in[p]}\beta_q\right)$.*

1. *Event $E(p')$ holds for all $p' \leq p$.*
2. *For every $p' \leq p$, at any moment during phase p' and for every $j \in [d]$ it holds that*

$$\mathrm{data}\left(\mathtt{Left}_j\right) \in_j \bigcup_{q\in[p']} \mathtt{Strip}^{\mathrm{left}}_{q,j}, \quad \textit{and} \quad \mathrm{data}\left(\mathtt{Right}_j\right) \in_j \bigcup_{q\in[p']} \mathtt{Strip}^{\mathrm{right}}_{q,j}.$$

3. *At the end of phase p, for every $j \in [d]$ and $s \in \{\mathrm{left}, \mathrm{right}\}$ the dataset D contains at least $2m_{p+1}$ points from $\mathtt{Strip}^s_{p+1,j}$ with the label 1, and all positively labeled points in D belong to $\mathtt{RECTANGLE}(C)$.*

Proof. The proof of is by induction on p.[5] To this end, we assume that Items 1-3 hold for $p-1$, and show that in this case they also hold for p, except with probability at most $2\beta_p$. First, by Lemma B.6, Item 1 holds with probability at least $1 - \beta_p$. We proceed with the analysis assuming that this is the case.

By assumption, at the end of phase $p-1$, the dataset D contains at least $2m_p$ points from every $\mathtt{Strip}^s_{p,j}$. As we assume that the noises throughout phase p are bounded (i.e., that Item 1 of the lemma holds), and asserting that $m_p \geq \frac{1}{\varepsilon}\log(\frac{1}{\delta_p})\log(\frac{4d}{\beta_p})$, we get that Item 2 of the lemma holds in the beginning of phase p (in Step 5(j)iii, when the copies of `ChallengeBT` are executed by the RSC paradigm). Throughout the phase, if we ever re-execute any of the copies of `ChallengeBT` in Step 5b, say the copy $\mathtt{Left}_j$, then this happens because $\mathtt{Left}_j$ has produced $\approx k_p$ "medium" answers. By our assumption

[5] The base case of this induction (for $p = 1$) follows from similar arguments to those given for the induction step, and is omitted for simplicity.

that the noise is bounded, and by asserting that $k_p \geq 2\Delta_p$, we get that there were at least $k_p/2$ "medium" answers. (Here Δ_p bounds the Laplace noise throughout the phase, see Definition B.5.) Note that every query x that generated such a "medium" answer is added to D_j^{left}. Again by our assumption that the noise is bounded, and by asserting that $m_p \geq 4\Delta_p$, every such query x satisfies $x \in_j \text{data}(\texttt{Left}_j)$. Hence, if we re-execute `ChallengeBT` in Step 5b, then the projection of its new dataset on the jth axis is contained within the projection of its previous dataset. Furthermore, by asserting that $k_p \geq 2m_p$ we get that the new dataset contains at least m_p points. This implies (by induction) that Item 2 of the lemma continues to hold throughout all of the phase.

We now proceed with the analysis of Item 3. We have already established that, conditioned on Items 1,2,3 holding for $p-1$, then with probability at least $1-\beta_p$ we have that Items 1,2 hold also for p. Furthermore, this probability is over the sampling of the noises throughout phase p. The query points which are sampled throughout phase p from the target distribution $\mathcal{D}$ are independent of these events. Recall that there are t_p time steps throughout phase p, and that roughly γ portion of them are sampled from $\mathcal{D}$. Recall that each $\texttt{Strip}^s_{p+1,j}$ has weight α_{p+1}/d under $\mathcal{D}$. Thus, by the Chernoff bound (and a union bound over j, s), if $t_p \geq \frac{8d}{\gamma\alpha_p} \ln\left(\frac{2d}{\beta_p}\right)$, then with probability at least $1-\beta_p$, throughout phase p we receive at least $\frac{\gamma\cdot\alpha_p\cdot t_p}{2d}$ points from every $\texttt{Strip}^s_{p+1,j}$. By the assumption that the noises are bounded (Item 1), each of these points is added to the dataset D with the label 1. Thus, provided that $t_p \geq \frac{4d}{\gamma\cdot\alpha_p}\cdot m_{p+1}$, then the dataset D at the end of phase p contains at least $2\cdot m_{p+1}$ points from every $\texttt{Strip}^s_{p+1,j}$. Note that this dataset might contain additional points:

- D might contain additional points with label 0, but these do not affect the continuation of the execution.
- D might contain additional points with label 1. However, again by the assumption that the noises are bounded, we have that our labeling error is *one sided*, meaning that if a point is included in D with the label 1 then its true label *must* also be 1.

To summarize, the dataset D contains at least $2\cdot m_{p+1}$ (distinct) points from every $\texttt{Strip}^s_{p+1,j}$ with the label 1, and all positively labeled points it contains belong to $\texttt{RECTANGLE}(C)$. This completes the proof. □

The utility guarantees of `RectanglesPERP` now follow from Lemma B.7.

Theorem B.8. *For every α, β, γ Algorithm `RectanglesPERP` is an $(\alpha, \beta, \gamma, n)$-everlasting robust predictor where $n = \Theta\left(\frac{d}{\alpha\cdot\varepsilon^2}\,\text{polylog}(\frac{d}{\alpha\cdot\beta\cdot\gamma\cdot\varepsilon\cdot\delta^*})\right)$.*

Proof. At time i of the execution, consider the hypothesis defined by Steps 5c–5i of `RectanglesPERP`. (This hypothesis takes an unlabeled point x_i and returns a label $\hat{y}_i$.) We now claim that whenever Items 1 and 2 defined by Lemma B.7 hold (which happens with probability at least $1-\beta$), then all of these hypotheses has error at most α w.r.t. the target distribution $\mathcal{D}$ and the target concept c. To see this, observe that whenever Items 1 and 2 hold, then algorithm `RectanglesPERP` never errs on point outside of

$$\bigcup_{\substack{p\in\mathbb{N}, j\in[d]\\ s\in\{\text{left},\text{right}\}}} \texttt{Strip}^s_{p,j},$$

which has weight at most α under $\mathcal{D}$. □

Remark B.9. *For constant d, the analysis outlined above holds even if the target concept c is chosen adaptively based on the initial sample S (and then S is relabeled according to the chosen concept c). The only modification is that in the base case of Lemma B.7, we cannot use the Chernoff bound to show that each $\texttt{Strip}^s_{1,j}$ contains at least $2\cdot m_1$ points. Instead we would rely on standard uniform convergence arguments for VC classes, showing that w.h.p. over sampling the unlabeled points in S, the probability mass of* every *possible rectangle is close to its empirical weight in S up to a difference of $O(\alpha)$. This argument does not extend directly to super-constant values for d, because in the analysis we argued about "strips" with weight $\approx \alpha/d$, rather than α, but this does not change much when $d = O(1)$.*

Remark B.10. *In all our constructions (including Observation 1.3), the robustness parameter γ could actually be allowed to* decrease *throughout the execution. That is, as the execution progresses, the algorithm could potentially withstand growing rates of adversarial queries. For simplicity, in the analysis of Algorithm `RectanglesPERP` (and in Definition 3.1) we treat γ as remaining fixed throughout the execution.*

B.3. Decision-stumps

A decision stump could be thought of as a halfspace which is aligned with one of the principal axes. Thus, decision-stumps in d dimensions could be privately predicted using our construction for rectangles from Section 4.2. But this would be extremely wasteful as the VC dimension of decision-stumps is known to be $\Theta(\log d)$ instead of $\Theta(d)$. We briefly describe a simple PERP for this class which is computationally-efficient and exhibits sample complexity linear in $\log d$. Formally,

Definition B.11 (Decision-stumps)**.** *Let $d \in \mathbb{N}$ denote the dimension. For every $j \in [d]$, $\sigma \in \{\pm 1\}$, and $t \in \mathbb{R}$, define the concept* $\mathrm{desision}_{j,\sigma,t} : \mathbb{R}^d \to \{0,1\}$ *as follows. For $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$ we have* $\mathrm{desision}_{j,\sigma,t}(x) = \mathrm{sign}(\sigma \cdot (x_j - t))$*. Define the class of all decision-stumps over $\mathbb{R}^d$ as* $\mathrm{DECISION}_d = \{\mathrm{desision}_{j,s,t} : j \in [d], \sigma \in \{\pm 1\}, t \in \mathbb{R}\}$.

Algorithm 7 `DecisionPERP`

Initial input: Labeled dataset $S \in (\mathbb{R}^d \times \{0,1\})^n$ and parameters $\varepsilon, \delta, \alpha, \beta, \gamma$.

1. Use the exponential mechanism with privacy parameter $\frac{\varepsilon}{4}$ to select a pair $(j, \sigma) \in [d] \times \{\pm 1\}$ for which there is a decision stump with low empirical error on S.
2. Let p denote the number of positively labeled points in S, and let $\hat{p} \leftarrow p + \mathrm{Lap}(\frac{4}{\varepsilon})$.
3. Let D be a dataset containing the projection of the points in S onto the jth axis. Sort D in an ascending order if $\sigma = 1$, and in descending order otherwise. Relabel the first $\hat{p}$ examples in D as 1 and the other examples as 0.
4. Execute algorithm `RectanglesPERP` on D (as a multiset) for predicting thresholds in 1 dimension. Use parameters $\frac{\varepsilon}{4}, \frac{\delta}{2}, \frac{\alpha}{2}, \frac{\beta}{2}, \gamma$. During the execution, pass only the j coordinate of the given queries to `RectanglesPERP`.

Theorem B.12. *Algorithm* `DecisionPERP` *is an $(\alpha, \beta, \gamma, \varepsilon, \delta, n)$-PERP for the class of d-dimensional decision-stumps, where $n = \Theta\left(\frac{1}{\alpha\varepsilon}\log(d) + \frac{1}{\alpha\cdot\varepsilon^2}\,\mathrm{polylog}(\frac{1}{\alpha\cdot\beta\cdot\gamma\cdot\delta})\right)$.*

Proof. The privacy analysis is straightforward; follows from composition and from the fact that once $j, \sigma, \hat{p}$ are set, then changing one element of S affects at most two elements in D. As for the utility analysis, let $\mathcal{D}$ denote the target distribution and let (j^*, σ^*, t^*) denote the target concept. By the properties of the exponential mechanism, with probability at least $1 - \frac{\beta}{6}$, the pair (j, σ) selected in Step 1 is such that there is a number t with which (j, σ, t) misclassifies at most $O(\frac{1}{\varepsilon}\log\frac{d}{\beta})$ points in S. Now let $(j, \sigma, \hat{t})$ be a concept that agrees with the relabeled dataset D. By the properties of the Laplace distribution, with probability at least $1 - \frac{\beta}{6}$, these two concepts $(j, \sigma, t), (j, \sigma, \hat{t})$ disagree on at most $O(\frac{1}{\varepsilon}\log\frac{d}{\beta})$ points in S. By the triangle inequality, we hence get that $(j, \sigma, \hat{t})$ disagrees with the target concept (j^*, σ^*, t^*) on at most $O(\frac{1}{\varepsilon}\log\frac{d}{\beta})$ points in S. Asserting that $|S| = \Omega\left(\frac{1}{\alpha\varepsilon}\log\frac{d}{\beta}\right)$, by standard generalization arguments, with probability at least $1 - \frac{\beta}{6}$, the concept $(j, \sigma, \hat{t})$ has error at most $\frac{\alpha}{2}$ w.r.t. the target distribution and the target concept. Finally, by the properties of `RectanglesPERP`, with probability at least $(1 - \frac{\beta}{2})$ it guarantees $\frac{\alpha}{2}$-accuracy w.r.t. $(j, \sigma, \hat{t})$, even if $(1 - \gamma)$ fraction of the queries are adversarial. The theorem now follows by the triangle inequality. □