

Equivalence of Coarse and Fine-Grained Models for Learning with Distribution Shift

Adam Klivans

University of Texas at Austin

KLIVANS@UTEXAS.EDU

Shyamal Patel

Columbia University

SHYAMALPATELB@GMAIL.COM

Konstantinos Stavropoulos

University of Texas at Austin

KSTAVROP@UTEXAS.EDU

Arsen Vasilyan

University of Texas at Austin

ARSENVASILYAN@GMAIL.COM

Editors: Steve Hanneke and Tor Lattimore

Abstract

Recent work on provably efficient algorithms for learning with distribution shift has focused on two models: PQ learning (Goldwasser et al., 2020) and TDS learning (Klivans et al., 2024a). Algorithms for TDS learning are allowed to reject a test set entirely if distribution shift is detected. In contrast, PQ learners may only reject points that are deemed out-of-distribution on an individual basis. Our main result is a surprising equivalence between these two models in the distribution-free setting. In particular, we give an efficient black-box reduction from PQ learning to TDS learning for any Boolean concept class. This equivalence implies the first hardness results for distribution-free TDS learning of basic concept classes such as halfspaces. The main technical contribution underlying our equivalence is a method for boosting, via branching programs, the weak distinguishing power of TDS learners that have rejected the target domain.

We also show that giving a learner access to *membership queries* sidesteps these hardness results and allows for efficient, distribution-free PQ learnability of halfspaces. Our algorithm iteratively recovers large-margin separators obtained by applying successive Forster transforms on the training data.

Keywords: Computational Learning Theory, Complexity Theory

1. Introduction

One of the most fundamental models for learning with distribution shift is *domain adaptation*, where a learner is given access to a labeled training set drawn from a source distribution \mathcal{D} , an unlabeled test set drawn from a potentially different target distribution \mathcal{D}' , and the goal is to output a classifier that performs well with respect to the test set. The relationship between \mathcal{D} and \mathcal{D}' may be arbitrary, and prior work established conditions under which a classifier trained on some source distribution will generalize to a different target distribution (Ben-David et al., 2006; Blitzer et al., 2007; Mansour et al., 2009; Ben-David et al., 2010; David et al., 2010). In particular, these works prove generalization bounds assuming that some notion of distance between the source and target domains is bounded (e.g., discrepancy distance). These distances, however, are not efficiently verifiable and do not give algorithmic guarantees.

Motivated by these concerns, Goldwasser et al. (2020) defined the model of *Pointwise-Qualified learning* (PQ learning), where a learner produces a classifier that may abstain on points from the

test domain that it believes are out-of-distribution relative to the source domain. For points that are in-distribution the classifier is guaranteed to achieve low error. They give an algorithm that succeeds even under potentially extreme distribution shift but requires access to an oracle for empirical risk minimization (ERM). It is well known, however, that performing ERM is NP-hard for very simple function classes. In fact, their reduction requires an ERM oracle even when the training distribution is nicely structured (e.g., a Gaussian).

In search of truly efficient algorithms, [Klivans et al. \(2024a\)](#) introduced the model of Testable Distribution Shift learning (TDS learning), where an algorithm may abstain at the *population level*, either outputting a classifier with low error on the test set or rejecting the test set entirely. The rejection criterion of TDS learning is therefore much coarser than in PQ learning, where rejection is performed individually on test points. [Klivans et al. \(2024a\)](#) gave the first efficient algorithms for learning well-studied concept classes in this model (such as halfspaces), whenever the training distribution is nicely structured (e.g., log-concave or uniform on the hypercube). Their algorithms make no assumptions on the test distribution and do not require oracles for ERM. Extensions and improvements on these initial TDS results have appeared in multiple recent works ([Klivans et al., 2024a,b](#); [Chandrasekaran et al., 2024](#); [Goel et al., 2024](#); [Chandrasekaran et al., 2025](#)).

1.1. Our Main Result

In this paper, we prove, surprisingly, that distribution-free TDS learning and PQ learning are equivalent. Our main result is a computationally efficient black-box reduction from PQ learning to TDS learning, which shows that a learning algorithm that rejects distributions at the population level can be converted into one that makes per-example decisions about whether test points are in or out of distribution. At a high level, our reduction uses the method of boosting by branching programs, viewing TDS learners that reject as weak distinguishers of samples drawn from source and target domains.

This equivalence has strong implications: it immediately gives the first set of hardness results for distribution-free TDS learning of halfspaces (even halfspaces with a margin) and other basic concept classes, as these are known to be computationally intractable in the PQ setting ([Kalai and Kanade, 2021](#)). It further shows that assumptions on the training distribution are *necessary* for efficient TDS learnability, even for simple concept classes.

1.2. PQ Learning with Queries

Given these negative results, a natural question is whether there is some other way to circumvent hardness. As mentioned above, prior work on TDS learning has shown that assumptions on the training distribution such as log-concavity yield positive results for TDS learning and in some cases PQ learning ([Klivans et al., 2024a,b](#); [Chandrasekaran et al., 2024](#); [Goel et al., 2024](#); [Chandrasekaran et al., 2025](#)).

Here we explore a different path: can we design efficient, distribution-free PQ learning algorithms by giving the learner *membership query access*. That is, the learner can query the labels of arbitrary points in the support of the training distribution before seeing the test set. We make no assumptions on either the training or the test distributions. This setup provides more power to the learner but still preserves the core difficulty of distribution shift.

Recent work by [Lange and Qiao \(2025\)](#) shows that membership queries do *not* help in the related model of testable learning, suggesting that query access may not be helpful in our setting. Perhaps

unexpectedly, we show that queries do help in this setting and present an efficient, distribution-free PQ learner for halfspaces. Our algorithm works in stages, first using Forster transform to convert the halfspace into one with large enough margin, and then using queries to recover the parameters of the transformed halfspace. These results demonstrate a clear separation between testable learning and TDS learning. While membership queries fail to help in testable learning, they can circumvent hardness in the presence of distribution shift.

1.3. Technical Overview

Reducing PQ Learning to TDS Learning. Before describing our reduction, we briefly review the TDS and PQ learning models (see [Definition 1](#) and [Definition 2](#)). In both models, the learner has access to labeled samples from a training distribution $\mathcal{D}^{\text{train}}$ and unlabeled samples from a test distribution $\mathcal{D}^{\text{test}}$.

- A TDS learner either accepts or rejects. Upon acceptance, it must output a hypothesis whose error on $\mathcal{D}^{\text{test}}$ is at most ε . In addition, the learner is required to accept with high probability in the case where $\mathcal{D}^{\text{train}} = \mathcal{D}^{\text{test}}$.
- A PQ learner outputs a hypothesis h together with a selector g , which determines whether a given point should be classified. The selector must reject at most an ε fraction of points drawn from $\mathcal{D}^{\text{train}}$. Furthermore, for a random $\mathbf{x} \sim \mathcal{D}^{\text{test}}$, the probability that $g(\mathbf{x})$ accepts while $h(\mathbf{x})$ misclassifies \mathbf{x} must be at most ε .

Suppose we are given a TDS learning algorithm \mathcal{A} . If we run \mathcal{A} on samples from $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$ and it accepts, then with high probability the hypothesis h it outputs has error at most ε on $\mathcal{D}^{\text{test}}$. In this case, we immediately obtain the guarantees of PQ learning by taking the selector $g \equiv 1$ (i.e., rejecting no test points) together with the hypothesis h .

If, on the other hand, \mathcal{A} rejects, then it distinguishes between $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$ using m samples. In this regime, we may use a hybrid argument to obtain a weak distinguisher WD that, given a single sample, can distinguish whether it was drawn from $\mathcal{D}^{\text{train}}$ or $\mathcal{D}^{\text{test}}$ with advantage $\Omega(\frac{1}{m})$. Our main objective is then to boost this weak distinguishing advantage, producing an algorithm that still operates on a single sample but distinguishes between $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$ with high probability.

Before describing how we do this, we start by explaining why a black box boosting algorithm will not be sufficient for our purposes. In particular, consider using an algorithm such as AdaBoost to accomplish our goal. Running the AdaBoost algorithm with our weak distinguishing algorithm WD results in new distributions $(\mathcal{D}^{\text{train}})'$ and $(\mathcal{D}^{\text{test}})'$. We can then run \mathcal{A} on samples from $(\mathcal{D}^{\text{train}})'$ and $(\mathcal{D}^{\text{test}})'$. We have the following cases:

- If \mathcal{A} rejects, we get a weak distinguisher between these two distributions and can continue with the boosting procedure.
- However, if \mathcal{A} accepts, we get a hypothesis with good accuracy under $(\mathcal{D}^{\text{test}})'$ rather than a distinguisher. Unfortunately, h could have large error under $\mathcal{D}^{\text{test}}$ as $(\mathcal{D}^{\text{test}})'$ could be significantly different. Moreover, since we don't have a weak distinguisher between $(\mathcal{D}^{\text{train}})'$ and $(\mathcal{D}^{\text{test}})'$, it is unclear how to continue to make progress towards distinguishing $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$.

To circumvent this issue, we will show that a modification of the martingale boosting algorithm of Long and Servedio (2005) will allow us to prove our result. In particular, using this scheme, we will partition the domain into disjoint regions $\mathcal{R}_1, \dots, \mathcal{R}_k$ such that for each region \mathcal{R}_i either (i) \mathcal{A} accepts on $\mathcal{D}^{\text{train}}|_{\mathcal{R}_i}$ and $\mathcal{D}^{\text{test}}|_{\mathcal{R}_i}$ where $\cdot|_{\mathcal{R}_i}$ denotes the distribution conditioned on $x \in \mathcal{R}_i$, (ii) points in \mathcal{R}_i are primarily from $\mathcal{D}^{\text{test}}$, i.e. $\Pr_{x \sim \mathcal{D}^{\text{train}}}[x \in \mathcal{R}_i] \leq \varepsilon \Pr_{z \sim \mathcal{D}^{\text{test}}}[z \in \mathcal{R}_i]$, or (iii) points in \mathcal{R}_i are primarily from $\mathcal{D}^{\text{train}}$, meaning $\Pr_{z \sim \mathcal{D}^{\text{test}}}[z \in \mathcal{R}_i] \leq \varepsilon \Pr_{x \sim \mathcal{D}^{\text{train}}}[x \in \mathcal{R}_i]$.

Notably, given such a decomposition, we can set our selector function to reject on regions satisfying (ii), i.e. those containing many more points from $\mathcal{D}^{\text{test}}$ than $\mathcal{D}^{\text{train}}$. In particular,

$$g(x) = \begin{cases} 0 & \exists \mathcal{R}_i : x \in \mathcal{R}_i \wedge \Pr_{z \sim \mathcal{D}^{\text{train}}}[z \in \mathcal{R}_i] \leq \varepsilon \Pr_{z \sim \mathcal{D}^{\text{test}}}[z \in \mathcal{R}_i] \\ 1 & \text{otherwise} \end{cases}$$

Notably, such a selector rejects a small fraction of $\mathcal{D}^{\text{train}}$ as

$$\Pr_{x \sim \mathcal{D}^{\text{train}}}[g(x) = 0] = \sum_{\substack{i \in [k]: \\ \mathcal{R}_i \text{ satisfies (ii)}}} \Pr_{x \sim \mathcal{D}^{\text{train}}}[x \in \mathcal{R}_i] \leq \sum_{\substack{i \in [k]: \\ \mathcal{R}_i \text{ satisfies (ii)}}} \varepsilon \Pr_{x \sim \mathcal{D}^{\text{test}}}[x \in \mathcal{R}_i] \leq \varepsilon$$

With such a selector, we can then set our hypothesis $h(x)$ as follows: If x is contained in a region \mathcal{R}_i satisfying (i) set $h(x) := h_i(x)$, where h_i is the hypothesis output by \mathcal{A} on $\mathcal{D}^{\text{train}}|_{\mathcal{R}_i}$ and $\mathcal{D}^{\text{test}}|_{\mathcal{R}_i}$. On the other hand, for all x in regions \mathcal{R}_i satisfying (iii) we set $h(x) = 1$.

By an analogous computation as done for the selector, we have that points from $\mathcal{D}^{\text{test}}$ lie in a region \mathcal{R}_i satisfying (iii) with probability at most ε . On the other hand, each h_i has error at most ε under $\mathcal{D}^{\text{test}}|_{\mathcal{R}_i}$. Since the selector rejects any point lying in a region satisfying (ii), it then follows that the probability that $g(x) = 1$ and h makes an error is then at most 2ε , yielding the desired PQ learner.

PQ Learning Halfspaces with Queries. To prove our result on PQ learning halfspaces, we start by noting that it essentially suffices to prove the statement for homogenous halfspaces, i.e. those of the form $\text{sign}(w \cdot x)$ with $w \in \mathbb{R}^n$. In this homogenous setting, we'll assume, without loss of generality, that $\mathcal{D}^{\text{train}}, \mathcal{D}^{\text{test}} \subseteq \mathbb{S}^{n-1}$ and that the vector $w \in \mathbb{R}^n$ defining the halfspace satisfies $\|w\| = 1$.

The key idea is then to simply notice that if $w \cdot x$ has many points with margin at least γ , i.e. satisfying $|w \cdot x| \geq \gamma$, then any halfspace $\text{sign}(\hat{w} \cdot x)$, where \hat{w} has Euclidean distance at most $\gamma/3$ from w , agrees with w on these high margin points. In particular, any point $x \in \mathbb{S}^{n-1}$ with $|\hat{w} \cdot x| \geq 2\gamma/3$ has $\text{sign}(\hat{w} \cdot x) = \text{sign}(w \cdot x)$. Moreover, any point $x \in \mathbb{S}^{n-1}$ with $|w \cdot x| \geq \gamma$ must satisfy $|\hat{w} \cdot x| \geq 2\gamma/3$. Thus, if we could compute a high accuracy estimate \hat{w} of w , we could use the selector function $g(x) = \mathbb{I}\{|\hat{w} \cdot x| \geq 2\gamma/3\}$ and hypothesis $h(x) = \text{sign}(\hat{w} \cdot x)$ to correctly label all high margin points. On the other hand, computing such an estimate \hat{w} can be done easily with queries by noting that the center of mass under $\mathcal{N}(0, I_n)$ of the points $\{x \in \mathbb{R}^n : w \cdot x \geq 0\}$ points in the direction of w .

Of course, since the distribution $\mathcal{D}^{\text{train}}$ is arbitrary, it's possible that no points have large margin. As such, our algorithm first applies a Forster transform to transform the point set into one where there is a non-trivial fraction of points with a margin. We can then run the above high margin PQ learning algorithm to correctly label all points with a high margin. Afterwards, we can recurse on the unlabeled points until only an ε fraction of points from $\mathcal{D}^{\text{train}}$ are unlabeled.

Given a point $x \sim \mathcal{D}^{\text{test}}$, we apply the transformations and classifiers computed for $\mathcal{D}^{\text{train}}$ to x . If any of the high margin PQ learners that we trained accepts, we accept and label the point according to the learner. Otherwise, we reject. Notably, since our high margin PQ learner correctly labels all points accepted by its selector, our hypothesis will have low error over $\mathcal{D}^{\text{test}}$ whenever the selector accepts. On the other hand, since our transformations and classifiers label most of our training set, we expect only an ε fraction of points from $\mathcal{D}^{\text{train}}$ to be rejected by our selector, as desired.

1.4. Related Work

Learning via Refutation Recall that the first step of our proof is to obtain a weak distinguisher between the training and test distributions using a hybrid argument. Similar hybrid arguments have appeared in prior work on learning via refutation (Vadhan, 2017; Kothari and Livni, 2018). In particular, Kothari and Livni (2018) use such an argument to derive a weak PAC learner from a distinguishing algorithm that can tell whether (i) there exists a concept in a given class achieving nontrivial advantage on a labeled distribution, or (ii) the labels are completely random. This weak learner is then converted into a strong agnostic learner via a black-box boosting procedure.

In contrast, standard black-box boosting does not apply in our setting, since the TDS learner may accept and output a hypothesis rather than explicitly distinguish the two distributions. To address this, we build on the boosting framework of Long and Servedio (2005) and develop a new analysis that accommodates weak distinguishers that are allowed to abstain.

Domain Adaptation Domain adaptation has been extensively studied over the past two decades (Ben-David et al., 2006; Blitzer et al., 2007; Mansour et al., 2009; Ben-David et al., 2010; David et al., 2010; Redko et al., 2020). As in our setting, this line of work considers learners that observe labeled samples from a source distribution and unlabeled samples from a target distribution, and aim to achieve low error on the target distribution—without the option to reject.

A classical approach (Ben-David et al., 2006; Blitzer et al., 2007; Mansour et al., 2009) bounds the target error of an empirical risk minimizer trained on source data by the sum of a joint optimal error term and a measure of divergence between the source and target marginals, such as the discrepancy or d_A distance. These divergences can be estimated statistically from unlabeled data, yielding statistically efficient guarantees. However, computing discrepancy-based distances is computationally intractable in general, requiring exponential time even for simple concept classes such as halfspaces (Chandrasekaran et al., 2024).

PQ Learning PQ learning was introduced by Goldwasser et al. (2020), who showed that access to an agnostic ERM oracle suffices to solve the problem. Kalai and Kanade (2021) subsequently established that PQ learning is computationally equivalent to *reliable agnostic learning* Kalai et al. (2012). Reliable agnostic learning is strictly easier than standard agnostic learning; for example, parities over the Boolean hypercube admit efficient reliable agnostic learners. In contrast, reliably learning conjunctions—and hence PQ learning for this class—is at least as hard as PAC learning DNFs, a long-standing open problem that serves as a canonical barrier in computational learning theory.

Goel et al. (2024) later studied a distribution-specific variant of PQ learning and obtained the first efficient algorithms for several concept classes under Gaussian training distributions.

Testable Learning Testable learning with distribution shift (TDS learning) was introduced by Klivans et al. (2024a) and is closely related to the model of testable agnostic learning of Rubinfeld and Vasilyan (2023). Both frameworks require the learner to certify its own performance. In testable agnostic learning, there is no distribution shift, and the learner must either output a hypothesis with near-optimal error or reject upon detecting a violation of a target distributional assumption. TDS learning generalizes this framework to settings where the training and test distributions may differ.

A substantial line of work has developed computationally efficient algorithms for testable agnostic learning (Rubinfeld and Vasilyan, 2023; Gollakota et al., 2023, 2024b; Diakonikolas et al., 2024; Gollakota et al., 2024a; Diakonikolas et al., 2023a; Slot et al., 2024; Klivans et al., 2025) and for TDS learning (Klivans et al., 2024a,b; Chandrasekaran et al., 2024; Goel et al., 2024; Chandrasekaran et al., 2025), primarily in distribution-specific settings such as the Gaussian distribution or the uniform distribution over the Boolean hypercube.

In the distribution-free setting, testable agnostic learning coincides with standard agnostic learning. Prior to our work, the strongest known relationship was that TDS learning is no harder than PQ learning, and hence no harder than reliable agnostic learning. In particular, it was unknown whether TDS learning of halfspaces admits an efficient algorithm, even in the presence of margin assumptions. Our results resolve this question by showing that TDS learning is computationally equivalent to reliable agnostic learning: strictly easier than agnostic learning, yet harder than realizable PAC learning. As a consequence, TDS learning of halfspaces (even with margin) inherits hardness barriers such as PAC learning DNFs.

Finally, Lange and Qiao (2025) showed that membership queries do not provide additional power for testable agnostic learning. In contrast, we show that membership queries can be leveraged in the TDS setting by giving an efficient TDS learner for halfspaces that uses membership queries prior to observing any test examples.

2. Preliminaries

Notation We let \mathcal{X} be some domain. A concept class \mathcal{C} over \mathcal{X} is a set of functions $c : \mathcal{X} \rightarrow \{\pm 1\}$. For a distribution \mathcal{D} over \mathcal{X} , we denote with $\bar{\mathcal{D}}$ the corresponding labeled distribution, where the marginal distribution on the labels as well as their coupling with \mathcal{D} will be clear by the context. In particular, when we say that \mathcal{D} is labeled by some $f \in \mathcal{C}$, then the distribution $\bar{\mathcal{D}}$ is the distribution over pairs $(\mathbf{x}, f(\mathbf{x}))$, where $\mathbf{x} \sim \mathcal{D}$. Finally, we will use boldface to denote random variables.

2.1. TDS and PQ Learning

Definition 1 (TDS Learning (Klivans et al., 2024a)) *Let \mathcal{C} be a concept class over \mathcal{X} . The algorithm \mathcal{A} is a TDS learner for \mathcal{C} up to error ε probability of failure δ if, provided labeled sample access to a training distribution $\bar{\mathcal{D}}^{\text{train}}$ labeled by some unknown $f \in \mathcal{C}$, and unlabeled sample access to a test distribution $\mathcal{D}^{\text{test}}$, algorithm \mathcal{A} either outputs **REJECT** or outputs **ACCEPT** and returns a hypothesis $h : \mathcal{X} \rightarrow \{\pm 1\}$ such that, with probability at least $1 - \delta$, the following hold:*

- (a) (soundness) Upon acceptance, the test error is bounded as $\Pr_{\mathbf{x} \sim \mathcal{D}^{\text{test}}}[f(\mathbf{x}) \neq h(\mathbf{x})] \leq \varepsilon$.
- (b) (completeness) If $\mathcal{D}^{\text{train}} = \mathcal{D}^{\text{test}}$, then the algorithm accepts.

Definition 2 (PQ Learning (Goldwasser et al., 2020)) *Let \mathcal{C} be a concept class over \mathcal{X} . The algorithm \mathcal{A} is a PQ learner for \mathcal{C} up to error ε and probability of failure δ if, provided labeled sample*

access to a training distribution $\bar{\mathcal{D}}^{\text{train}}$ labeled by some unknown $f \in \mathcal{C}$, and unlabeled sample access to a test distribution $\mathcal{D}^{\text{test}}$, algorithm \mathcal{A} outputs, with probability at least $1 - \delta$, a hypothesis $h : \mathcal{X} \rightarrow \{\pm 1\}$ and a selector $g : \mathcal{X} \rightarrow \{0, 1\}$ such that:

- (a) (accuracy) The test error is bounded as $\Pr_{\mathbf{x} \sim \mathcal{D}^{\text{test}}}[f(\mathbf{x}) \neq h(\mathbf{x}) \text{ and } g(\mathbf{x}) = 1] \leq \varepsilon$.
- (b) (rejection rate) The probability of rejection is bounded as $\Pr_{\mathbf{x} \sim \mathcal{D}^{\text{train}}}[g(\mathbf{x}) = 0] \leq \varepsilon$.

3. A Reduction from PQ Learning to TDS Learning

3.1. Getting a Weak Distinguisher

Let \mathcal{A} be the TDS algorithm and suppose that it draws at most m samples from $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$. We begin by noting that if \mathcal{A} is likely to reject a set of samples from $\mathcal{D}^{\text{test}}$ then it is clearly able to distinguish between $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$ using m samples. To formalize these statements, we make the following definitions.

Definition 3 (“Rejecting” TDS Algorithm) Given distributions $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$, we say that a TDS learning algorithm \mathcal{A} is rejecting if \mathcal{A} outputs *REJECT* with probability at least 0.02.

Definition 4 (Distinguishing Advantage) We say that an algorithm \mathcal{A} distinguishes \mathcal{D} and \mathcal{D}' with m samples and advantage γ if

$$|\Pr_{\mathbf{a}_1, \dots, \mathbf{a}_m \sim \mathcal{D}}[\mathcal{A}(\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_m) = 1] - \Pr_{\mathbf{b}_1, \dots, \mathbf{b}_m \sim \mathcal{D}'}[\mathcal{A}(\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_m) = 1]| \geq \gamma$$

Notably, a rejecting TDS algorithm, \mathcal{A} , has $\Omega(1)$ distinguishing advantage using m samples. Our main goal in this section will be to show, by a hybrid argument, that this implies the existence of an algorithm that distinguishes between $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$ using only a single sample and with distinguishing advantage $\Omega(1/m)$.

Indeed, we show this precisely with the following lemma, proven in [Appendix C](#).

Lemma 5 Let $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$ be distributions, \mathcal{A} be an $(\varepsilon, 0.01)$ -TDS learner for a class \mathcal{C} that is rejecting algorithm and uses at most m samples from $\mathcal{D}^{\text{test}}$, and $\delta \in (0, 1/3)$, then with probability $1 - \delta$, $\text{GETWEAKDISTINGUISHER}(\mathcal{D}^{\text{train}}, \mathcal{D}^{\text{test}}, \mathcal{A})$ (cf. [Appendix C](#)) outputs an algorithm with distinguishing advantage $\Omega(1/m)$ that only uses a single sample.

3.2. Boosting the Weak Distinguisher to Obtain a PQ Learner

Our main result is to provide a PQ learning algorithm that has access to a TDS learning oracle and runs in time polynomial to the oracle’s run time. In particular, we prove the following result.

Theorem 6 (PQ learning via TDS learning) Let \mathcal{A} be an $(\varepsilon, 0.01)$ -TDS learner for some class \mathcal{C} that runs in time $T_{\mathcal{A}}(\varepsilon)$ and has total (training plus test) sample complexity $m_{\mathcal{A}}(\varepsilon)$. Then, [Algorithm 1](#) is a PQ learner for \mathcal{C} up to error and rejection rate $O(\varepsilon)$ and probability of failure δ . In particular, [Algorithm 1](#) calls \mathcal{A} $\text{poly}(m_{\mathcal{A}}, 1/\varepsilon, \log 1/\delta)$ times, uses $\text{poly}(m_{\mathcal{A}}, 1/\varepsilon, \log 1/\delta)$ additional samples from the training and test distributions, and runs in time $T_{\mathcal{A}} \cdot \text{poly}(m_{\mathcal{A}}, 1/\varepsilon, \log 1/\delta)$.

Before describing the algorithm and the idea of the proof, we make the following observations.

Remark 7 The output of TDSBOOST (*Algorithm 1*) consists of a hypothesis and a selector. Their evaluation requires access to $(\text{frozen}(i, t))_{i,t}$, $(\text{label}(i, t))_{i,t}$, and to the functions $(\text{next}_{i,t}^A)_{i,t}$. Because evaluating $\text{next}_{i,t}^A$ requires oracle access to \mathcal{A} , the representations of the hypothesis and selector must encode sufficient information to invoke this oracle, e.g., by including the code of \mathcal{A} . Moreover, the hypothesis and the selector are both randomized functions.

Remark 8 In the premise of *Theorem 6*, we assume that \mathcal{A} is a distribution-free TDS learning algorithm, i.e., it satisfies the guarantees of *Definition 1* for every training distribution $\mathcal{D}^{\text{train}}$. This assumption can be relaxed. Indeed, *Algorithm 1* invokes \mathcal{A} only on samples drawn from truncated versions of $\mathcal{D}^{\text{train}}$. Since truncation preserves any property \mathcal{I} that holds almost surely (e.g., boundedness or the existence of a margin with respect to a target concept f), the same argument yields a PQ learner under any distribution $\mathcal{D}^{\text{train}}$ satisfying \mathcal{I} , provided that \mathcal{A} is a TDS learner for distributions satisfying \mathcal{I} .

Algorithm Idea TDSBOOST (*Algorithm 1*) is an extension of the Martingale boosting algorithm of Long and Servedio (2005), which was proposed in the context of boosting a weak PAC learner to obtain a strong PAC learning algorithm with favorable noise tolerance properties compared to prior work. Here, we are building on these ideas in the context of learning with distribution shift, by using them to boost the distinguishing advantage provided by a rejecting TDS learner according to *Lemma 5*.

In particular, we consider the following contrived learning problem. Suppose that we wish to learn a function $g : \mathcal{X} \times \{0, 1\} \rightarrow \{0, 1\}$, where $g(x, b) = b$, under a distribution \mathcal{G} . To draw (x, b) from \mathcal{G} , with probability $\frac{1}{2}$, we draw $x \sim \mathcal{D}_1$ and set $b = 1$. Otherwise, we draw $x \sim \mathcal{D}_0$ and set $b = 0$. Due to *Lemma 5*, as long as the TDS learner \mathcal{A} rejects with probability at least $\Omega(1)$ when run on the distributions $\mathcal{D}_1, \mathcal{D}_0$, we have that *Algorithm 3* is a weak learner for the above learning problem, i.e., outputs a hypothesis $\text{WD}^{\mathcal{A}} : \mathcal{X} \rightarrow \{0, 1\}$ such that:

$$\Pr_{(x,b) \sim \mathcal{G}} \left[\text{WD}^{\mathcal{A}}(x) = g(x, b) \right] \geq \frac{1}{2} + \gamma,$$

where $\gamma = \Omega(1/m_{\mathcal{A}})$ and $m_{\mathcal{A}}$ is the total sample complexity of \mathcal{A} . For the remainder of this subsection, we will use the notation \mathcal{D}_1 to denote the training distribution $\mathcal{D}^{\text{train}}$, and \mathcal{D}_0 to denote the test distribution $\mathcal{D}^{\text{test}}$.¹

Note that since \mathcal{A} is a distribution-free TDS learner, at a first glance, one might think that *Algorithm 3* is a distribution-free weak learner for the problem of learning g . If that were indeed the case, then we could apply Martingale boosting in order to obtain a strong learner for g whose output \hat{g} only depends on x and not b .² This, in turn, would give a PQ learner, because \hat{g} would strongly distinguish between the training and test distributions and we would, therefore, have the following high-probability bounds on the error and rejection rates:

$$\begin{aligned} \Pr_{x \sim \mathcal{D}_0} [1 \neq f(x), \hat{g}(x) = 1] &\leq \Pr_{x \sim \mathcal{D}_0} [\hat{g}(x) \neq 0] \leq \varepsilon \\ &\quad \text{(Here we used the trivial hypothesis } \hat{h} \equiv 1) \\ \Pr_{x \sim \mathcal{D}_1} [\hat{g}(x) = 0] &= \Pr_{x \sim \mathcal{D}_1} [\hat{g}(x) \neq 1] \leq \varepsilon \end{aligned}$$

-
1. This notation is convenient because we can use \mathcal{D}_b for $b \in \{0, 1\}$ to refer to either the training distribution ($b = 1$, and also $g(x, b) = 1$) or the test distribution ($b = 0$, and also $g(x, b) = 0$).
 2. The same would be true if we applied any boosting algorithm that outputs a hypothesis \hat{g} with the following property: If the outputs of the weak learners only depend on x and not b , then \hat{g} is a junta on x as well.

Algorithm 1 TDSBOOST($\mathcal{A}, \bar{\mathcal{D}}_1, \mathcal{D}_0, \varepsilon, \delta$)

Input: TDS learning oracle \mathcal{A} for some class \mathcal{C} , sample access to (training) distribution $\bar{\mathcal{D}}_1$ realizable by \mathcal{C} , sample access to unlabeled (test) distribution \mathcal{D}_0 , and $\varepsilon, \delta \in (0, 1)$

Output: A hypothesis $\hat{h}^{\mathcal{A}} : \mathcal{X} \rightarrow \{\pm 1\}$ and a selector $\hat{g}^{\mathcal{A}} : \mathcal{X} \rightarrow \{0, 1\}$

- 1 Let $m_{\mathcal{A}}$ be the sample complexity of \mathcal{A} with parameters $(\varepsilon, 0.01)$
- 2 Let $C \geq 1$ be a sufficiently large constant and let $T = C m_{\mathcal{A}}^2 \log \frac{1}{\varepsilon}$, and $\delta' = \frac{\delta}{C T^{10}}, r = C \log \frac{T}{\varepsilon \delta}, p_{\min} = \frac{3\varepsilon}{4T(T+1)}, a_{\min} = 0.95, \gamma_1 = \frac{\varepsilon}{4T(T+1)}, \gamma_2 = \frac{1}{C m_{\mathcal{A}}}$
- 3 Set $V_{1,1}(\mathbf{x}) \equiv 1, V_{i,t}(\mathbf{x}) \equiv 0$ for $2 \leq t \leq T, 1 \leq i \leq t$; $\triangleright V_{i,t}(\mathbf{x}) = \mathbb{1}\{\mathbf{x}$ reaches node $(i, t)\}$.
- 4 Set $\text{next}_{i,t}^{\mathcal{A}}(\mathbf{x}) \equiv \perp$ for all $1 \leq i \leq t \leq T$; \triangleright Routing functions of the branching program
- 5 Set $\text{label}(i, T) \leftarrow \mathbb{1}\{i \geq T/2\}$ for all $1 \leq i \leq T$; \triangleright Assign node to \mathcal{D}_b (by giving label b)
- 6 Set $h_{i,t}(\mathbf{x}) \equiv 1$ for all $1 \leq i \leq t \leq T$; \triangleright Each node corresponds to some hypothesis
/* Constructing the Branching Program $P = (\text{next}^{\mathcal{A}}, \text{label})$ */
- 7 **for** $t = 1, 2, \dots, T$ **do**
- 8 **for** $i = 1, 2, \dots, t$ **do**
- 9 Compute (γ_1, δ') -estimates $\hat{p}_{i,t}^b$ of $p_{i,t}^b = \Pr_{\mathbf{x} \sim \mathcal{D}_b, P}[V_{i,t}(\mathbf{x}) = 1]$ for $b \in \{0, 1\}$
- 10 **if** either of these estimates is less than p_{\min} and $t < T$ **then**
- 11 Set $\text{label}(i, t) \leftarrow 1 - b$, with $\hat{p}_{i,t}^b < p_{\min}$; \triangleright If one distribution rarely visits a node,
assign the node to the other distribution
- 12 **end**
- 13 **else**
- 14 Set $\mathcal{G}_{i,t}^b$ to be the conditional distribution $\mathcal{D}_b \mid V_{i,t}(\mathbf{x}) = 1$ for $b \in \{0, 1\}$
- 15 Compute $(0.01, \delta')$ -estimate $\hat{a}_{i,t}$ of the probability that algorithm \mathcal{A} accepts when run on inputs $(\bar{\mathcal{G}}_{i,t}^1, \mathcal{G}_{i,t}^0, \varepsilon, \delta)$
- 16 **if** $\hat{a}_{i,t} \geq a_{\min}$ **then**
- 17 Set $\text{label}(i, t) \leftarrow 1$
- 18 Run algorithm \mathcal{A} on inputs $(\bar{\mathcal{G}}_{i,t}^1, \mathcal{G}_{i,t}^0, \varepsilon, \delta)$ independently r times and let $h_{i,t}$ be the majority function of the hypotheses produced by accepting runs
- 19 **end**
- 20 **else if** $t < T$ **then**
- 21 Set $\text{WD}_{i,t}^{\mathcal{A}} \leftarrow \text{GETWEAKDISTINGUISHER}(\bar{\mathcal{G}}_{i,t}^1, \mathcal{G}_{i,t}^0, \mathcal{A}, \delta')$
- 22 Set $\mathcal{G}_{i,t}$ to be the mixture $\frac{1}{2}\mathcal{G}_{i,t}^1 + \frac{1}{2}\mathcal{G}_{i,t}^0$
- 23 Compute (γ_2, δ') -estimate $\hat{q}_{i,t}$ of probability that $\text{WD}_{i,t}^{\mathcal{A}}(\mathbf{x}) = 1$ over $\mathbf{x} \sim \mathcal{G}_{i,t}$
- 24 Set $\hat{w}_{i,t}^{\mathcal{A}}$ to be the randomized function $\hat{w}_{i,t}^{\mathcal{A}}(\mathbf{x}) = \text{BALANCE}(\hat{q}_{i,t}, \text{WD}_{i,t}^{\mathcal{A}}(\mathbf{x}))$
- 25 Set $\text{next}_{i,t}^{\mathcal{A}}$ to be the randomized function $\text{next}_{i,t}^{\mathcal{A}}(\mathbf{x}) = (i + \hat{w}_{i,t}^{\mathcal{A}}(\mathbf{x}), t + 1)$
- 26 Update the function $V_{i+b,t+1}(\mathbf{x})$ to take the following values: $V_{i+b,t+1}(\mathbf{x}) + V_{i,t}(\mathbf{x}) \mathbb{1}\{\hat{w}_{i,t}^{\mathcal{A}}(\mathbf{x}) = b\}$ for all $b \in \{0, 1\}$
- 27 **end**
- 28 **end**
- 29 **end**
- 30 **end**
- 31 We set the hypothesis $\hat{h}^{\mathcal{A}}$ and the selector $\hat{g}^{\mathcal{A}}$ to be evaluated as follows:
We begin from $N = (1, 1) \in [T]^2$ and move to $\text{next}_N^{\mathcal{A}}(\mathbf{x})$ recursively until we reach $N \in [T]^2$ with $\text{next}_N^{\mathcal{A}}(\mathbf{x}) = \perp$. We set $\hat{h}^{\mathcal{A}}(\mathbf{x}) = h_N(\mathbf{x})$ and $\hat{g}^{\mathcal{A}}(\mathbf{x}) = \text{label}_N$

However, the above attempt fails, because [Algorithm 3](#) is not really distribution-free. In particular, [Algorithm 3](#) is guaranteed to work only when the distributions \mathcal{D}_1 and \mathcal{D}_0 are such that the algorithm \mathcal{A} rejects with non-trivial probability when run on samples from these distributions (see [Definition 3](#)). Fortunately, if \mathcal{A} rejects with very small probability, then we may use it to obtain a hypothesis with low test error. Moreover, we can incorporate this case in the Martingale boosting algorithm without introducing any technical complications, thereby obtaining the desired result.

Algorithm Description The algorithm TDSBOOST creates a branching program and uses it to partition the domain \mathcal{X} into a collection of regions for each of which it is either safe to abstain (selector takes the value 0) or to predict (selector takes the value 1).³

The branching program consists of nodes of the form $(i, t) \in [T]^2$ with $i \leq t$, routing functions for each node $\text{next}_{i,t}^A : \mathcal{X} \rightarrow \{(i, t+1), (i+1, t+1), \perp\}$, labels $\text{label}_{i,t} \in \{0, 1\}$ for leaf nodes (i.e., nodes with $\text{next}_{i,t}^A = \perp$), and auxiliary functions $h_{i,t} : \mathcal{X} \rightarrow \{\pm 1\}$ for leaf nodes.

A point $x \in \mathcal{X}$ starts from node $N = (1, 1)$ and is routed through the branching program by applying the function $\text{next}_N^A(x)$ recursively until it reaches a leaf node N with $\text{next}_N^A(x) = \perp$. The aforementioned partition contains one region for each leaf node N consisting of the points x that are routed to N . The label label_N of a leaf node corresponds to the suggested value of the selector, and the hypothesis h_N corresponds to the suggested classifier for the elements of the region corresponding to node N .

The algorithm creates the branching program using an iterative top-down approach. It considers for each node (i, t) the distributions $\mathcal{G}_{i,t}^1$ and $\mathcal{G}_{i,t}^0$ which correspond to \mathcal{D}_1 and \mathcal{D}_0 , respectively, conditioned on visiting node (i, t) through the branching program constructed so far.

If sampling from either of these distributions (through rejection sampling) is too costly, then at least one of $\mathcal{D}_1, \mathcal{D}_0$ rarely visits the node (i, t) and we make (i, t) a leaf node, with label opposite to the distribution that rarely visits it. Recall that the labels of the leaf nodes determine the values of the selector. This step is safe to do because it is always safe to abstain on regions of low probability under \mathcal{D}_1 , as well as give predictions on regions of low probability under \mathcal{D}_0 (see [Definition 2](#)).

The aforementioned conditioning defines a new TDS learning problem. If the given TDS learner \mathcal{A} is very likely to accept, then we use it to obtain a hypothesis $h_{i,t}$ with high accuracy on $\mathcal{G}_{i,t}^0$ and set its label to 1. Otherwise, the algorithm \mathcal{A} rejects with non-trivial probability and, by [Lemma 5](#), we may use [Algorithm 3](#) to obtain a weak distinguisher $\text{WD}_{i,t}^A$ for the distributions $\mathcal{G}_{i,t}^1$ and $\mathcal{G}_{i,t}^0$. Having $\text{WD}_{i,t}^A$ at hand, we are able to construct the routing function $\text{next}_{i,j}^A$ by using the rebalancing trick of [Long and Servedio \(2005\)](#), which is given by [Algorithm 2](#).

Algorithm 2 BALANCE(q, w)

Input: $q \in [0, 1], w \in \{0, 1\}$

Output: A random variable \hat{w} over $\{0, 1\}$

- 1 Let $b = \arg \min_{b' \in \{0,1\}} |b' - q|$ and ξ be Bernoulli random variable that is 1 with probability $\frac{1}{1+2|b-q|}$
 - 2 Set $\hat{w} \leftarrow w\xi + (1-b)(1-\xi)$
-

3. The branching program, and hence the corresponding partition of the domain \mathcal{X} , is randomized, as its routing functions $\text{next}_{i,t}^A$ are randomized.

Comparison to Martingale Boosting Similarly to standard Martingale boosting, the routing functions are chosen so that samples from the distribution \mathcal{D}_1 are biased to move towards nodes (i, t) where i is large, and samples from the distribution \mathcal{D}_0 are biased towards nodes with i small. This is accomplished by using the weak distinguishers provided by [Algorithm 3](#) to construct the routing functions. In particular, we inherit the following result from [Long and Servedio \(2005\)](#) which essentially shows that nodes of the form (i, T) with $i < T/2$ are rarely visited by samples of \mathcal{D}_1 and nodes (j, t) with $i \geq T/2$ are rarely visited by samples of \mathcal{D}_0 .

Lemma 9 (Implicit in Theorem 3 from Long and Servedio (2005)) *Let \mathcal{G} be the following distribution over $\mathcal{X} \times \{0, 1\}$. To draw (\mathbf{x}, \mathbf{b}) from \mathcal{G} , with probability $\frac{1}{2}$, we draw $z \sim \mathcal{D}_1$ and set $\mathbf{x} = z$ and $\mathbf{b} = 1$. Otherwise, we draw $z \sim \mathcal{D}_0$ and set $\mathbf{x} = z$ and $\mathbf{b} = 0$. Suppose that we obtain the randomized branching program P by running $\text{TDSBOOST}(\mathcal{A}, \bar{\mathcal{D}}_1, \mathcal{D}_0, \varepsilon, \delta)$ where \mathcal{A} is an (ε, δ) -TDS learner. Then we have:*

$$\Pr_{(\mathbf{x}, \mathbf{b}) \sim \mathcal{G}, P} \left[\mathbf{x} \text{ reaches node } (i, T) \text{ in } P \text{ and } \text{label}(i, T) \neq \mathbf{b} \right] \leq \frac{\varepsilon}{2}$$

The main difference between [Algorithm 1](#) and the Martingale boosting algorithm of [Long and Servedio \(2005\)](#) is that our algorithm creates some additional leaf nodes at levels $t < T$. However, this does not change the analysis of [Lemma 9](#), as the routing functions of the internal nodes are created based on valid weak distinguishers given by [Algorithm 3](#).

Sketch of the Proof As we have mentioned, [Algorithm 1](#) creates a branching program P that partitions the domain \mathcal{X} into regions $\{\mathcal{R}_N\}_{N \in \text{leaves}(P)}$ corresponding to the leaves $\text{leaves}(P)$ of P . For each such region \mathcal{R}_N , we either abstain if $\text{label}(N) = 0$ or predict according to hypothesis h_N if $\text{label}(N) = 1$. There are three types of leaf nodes:

1. The first type $\mathcal{T}_1 \subseteq \text{leaves}(P)$ includes nodes N at level $t < T$ that are rarely visited by \mathcal{D}_b for some $b \in \{0, 1\}$ and $\text{label}(N) = 1 - b$. If $b = 1$, then the training distribution assigns negligible mass to \mathcal{R}_N and hence abstaining on this region does not increase the rejection rate significantly. If $b = 0$, then the test distribution assigns negligible mass to \mathcal{R}_N and hence even if we make wrong predictions, the error rate is not increased significantly.
2. The second type $\mathcal{T}_2 \subseteq \text{leaves}(P)$ includes the nodes at level $t = T$. Due to [Lemma 9](#), these nodes have a similar property, i.e., $\Pr_{\mathbf{x} \sim \mathcal{D}_b, P} [\mathbf{x} \text{ reaches } \mathcal{T}_2 \text{ at node with label } 1 - b] \leq O(\varepsilon)$.
3. Finally, the third type $\mathcal{T}_3 \subseteq \text{leaves}(P)$ are those nodes N where the TDS learner accepts with probability at least 0.9, which enables us to learn a hypothesis with low test error conditioned on reaching node N , by amplifying the acceptance probability through the majority vote of several repetitions of the TDS learner (based on Proposition C.1 of [Klivans et al. \(2024a\)](#)).

Following this sketch, we provide a formal proof for [Theorem 6](#) in [Appendix D](#).

4. Learning Halfspaces with Distribution Shift with Queries

We note that a simple corollary of [Theorem 6](#) is that it is unlikely that there is a polynomial time algorithm in n and ε^{-1} to TDS learn halfspaces in the distribution-free setting. Indeed, combining [Theorem 6](#) with [Kalai and Kanade \(2021\)](#), which proves that PQ learning and reliable learning are

equivalent, implies that TDS learning of halfspaces to accuracy ε is at least as hard as learning an intersection of $\Omega(1/\varepsilon)$ halfspaces in the distribution-free PAC model. Notably, whether there exists a polynomial-time algorithm for learning even an intersection of even two halfspaces is a long standing open question. Moreover, under suitable cryptographic assumptions (Klivans and Sherstov, 2009; Tiegel, 2024), learning an intersection of polynomially many halfspaces requires sub-exponential time. Even for halfspaces with a margin, it is unlikely there is a polynomial time algorithm for TDS learning. In particular, if we consider learning conjunctions over $\{0, 1\}^n$, then every point has an inverse polynomial margin. By Theorem 6 and Remark 8, it then follows that we can reduce TDS learning of conjunctions over $\{0, 1\}^n$ to PQ learning of conjunctions. In turn, Kalai and Kanade (2021) showed PQ learning conjunctions is as hard as PAC learning DNFs, another long standing open problem in learning theory believed to not have polynomial time algorithms.

All that said, in this section, we will show that if we allow ourselves membership queries, then we can PQ and thus TDS learn halfspaces in polynomial time. In particular, we define a PQ learner with membership queries as follows:

Definition 10 (PQ Learning with Membership Queries) *Let \mathcal{C} be a concept class over \mathcal{X} . An PQ learning algorithm with membership queries, \mathcal{A} , for the class \mathcal{C} up to error ε and probability of failure δ operates as follows. First, the algorithm is given labeled sample access to a training distribution $\overline{\mathcal{D}}_{\text{train}}$ labeled by some unknown $f \in \mathcal{C}$ and can query arbitrary points $x \in \mathcal{X}$ to receive the value $f(x)$. Afterward, the algorithm is given unlabeled samples access to $\mathcal{D}^{\text{test}}$ and must (without making any further queries) output, with probability at least $1 - \delta$, a hypothesis $h : \mathcal{X} \rightarrow \{\pm 1\}$ and a selector $g : \mathcal{X} \rightarrow \{0, 1\}$ such that:*

- (a) (accuracy) *The test error is bounded as $\Pr_{\mathbf{x} \sim \mathcal{D}^{\text{test}}}[f(\mathbf{x}) \neq h(\mathbf{x}) \text{ and } g(\mathbf{x}) = 1] \leq \varepsilon$.*
- (b) (rejection rate) *The probability of rejection is bounded as $\Pr_{\mathbf{x} \sim \mathcal{D}^{\text{train}}}[g(\mathbf{x}) = 0] \leq \varepsilon$.*

In this setting, we will then prove

Theorem 11 *There exists a PQ learning algorithm for halfspaces over \mathbb{R}^n with membership queries that runs in time $\text{poly}(n, \frac{1}{\varepsilon}, \log(1/\delta))$.*

The proof of the theorem is shown in Appendix E. To prove the statement, we first show that we can essentially reduce to homogeneous halfspaces. Afterwards, we give a PQ learning algorithm for homogenous halfspaces with a margin. Finally, we combine our algorithm for PQ learning high-margin halfspaces with the Forster transform to prove Theorem 19.

Remark 12 *Our upper bound for learning halfspaces with membership queries holds even under the stronger reliable and probably useful (RPU) model Rivest and Sloan (1988); El-Yaniv and Wiener (2010, 2012). Unlike PQ learning, an RPU learner is only given (labeled) training examples, and outputs a selector g and a classifier h such that: whenever a point x is selected ($g(x) = 1$), the prediction is always correct, i.e., $h(x) = f^*(x)$.*

At first glance, this may appear to contradict prior exponential lower bounds for RPU learning with membership queries Hopkins et al. (2020). However, those results assume that queries are restricted to a bounded-radius ball, whereas our algorithm allows arbitrary queries in \mathbb{R}^n .

Discussion

We have shown that PQ and TDS learning are equivalent in the distribution-free setting, where learners are required to succeed for arbitrary training distributions. A natural open question is whether this equivalence extends to the distribution-specific setting, in which PQ or TDS learners are only required to succeed when the training distribution equals a fixed reference distribution \mathcal{D}^* , while no assumptions are made on the test distribution.

Existing work in distribution-specific learning establishes positive results for both models, but does not always yield matching guarantees. For example, there exists an efficient TDS learner for AC^0 circuits when \mathcal{D}^* is the uniform distribution on the Boolean hypercube, whereas no efficient PQ learner is currently known for this setting. Nevertheless, no formal separation between the two models has been established.

Acknowledgments

Adam Klivans, Konstantinos Stavropoulos, and Arsen Vasilyan were supported in part by the NSF AI Institute for Foundations of Machine Learning (IFML). Shyamal Patel was supported by NSF awards CCF-2106429, CCF-2107187, CCF-2218677, ONR award ONR-13533312, and an NSF Graduate Student Fellowship. Konstantinos Stavropoulos was also supported by the 2025 Apple Scholars in AI/ML PhD fellowship.

References

- Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19, 2006.
- Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79:151–175, 2010.
- John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman. Learning bounds for domain adaptation. *Advances in neural information processing systems*, 20, 2007.
- Gautam Chandrasekaran, Adam R Klivans, Vasilis Kontonis, Konstantinos Stavropoulos, and Arsen Vasilyan. Efficient discrepancy testing for learning with distribution shift. *arXiv preprint arXiv:2406.09373*, 2024.
- Gautam Chandrasekaran, Adam Klivans, Lin Lin Lee, and Konstantinos Stavropoulos. Learning neural networks with distribution shift: Efficiently certifiable guarantees. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=ed7zI291RF>.
- Shai Ben David, Tyler Lu, Teresa Luu, and Dávid Pál. Impossibility theorems for domain adaptation. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 129–136. JMLR Workshop and Conference Proceedings, 2010.
- Ilias Diakonikolas, Daniel Kane, Vasilis Kontonis, Sihan Liu, and Nikos Zarifis. Efficient testable learning of halfspaces with adversarial label noise. *Advances in Neural Information Processing Systems*, 36, 2023a.

- Ilias Diakonikolas, Christos Tzamos, and Daniel M Kane. A strongly polynomial algorithm for approximate forster transforms and its application to halfspace learning. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 1741–1754, 2023b.
- Ilias Diakonikolas, Daniel Kane, Sihan Liu, and Nikos Zarifis. Testable learning of general halfspaces with adversarial label noise. In *The Thirty Seventh Annual Conference on Learning Theory*, pages 1308–1335. PMLR, 2024.
- Ran El-Yaniv and Yair Wiener. On the foundations of noise-free selective classification. *J. Mach. Learn. Res.*, 11:1605–1641, August 2010. ISSN 1532-4435.
- Ran El-Yaniv and Yair Wiener. Active learning via perfect selective classification. *J. Mach. Learn. Res.*, 13(null):255–279, February 2012. ISSN 1532-4435.
- Surbhi Goel, Abhishek Shetty, Konstantinos Stavropoulos, and Arsen Vasilyan. Tolerant algorithms for learning with arbitrary covariate shift. *Advances in Neural Information Processing Systems*, 2024.
- Shafi Goldwasser, Adam Tauman Kalai, Yael Kalai, and Omar Montasser. Beyond perturbations: Learning guarantees with arbitrary adversarial test examples. *Advances in Neural Information Processing Systems*, 33:15859–15870, 2020.
- Aravind Gollakota, Adam R Klivans, and Pravesh K Kothari. A moment-matching approach to testable learning and a new characterization of rademacher complexity. *Proceedings of the fifty-fifth annual ACM Symposium on Theory of Computing*, 2023.
- Aravind Gollakota, Adam Klivans, Konstantinos Stavropoulos, and Arsen Vasilyan. Tester-learners for halfspaces: Universal algorithms. *Advances in Neural Information Processing Systems*, 36, 2024a.
- Aravind Gollakota, Adam R Klivans, Konstantinos Stavropoulos, and Arsen Vasilyan. An efficient tester-learner for halfspaces. *The Twelfth International Conference on Learning Representations*, 2024b.
- Max Hopkins, Daniel Kane, and Shachar Lovett. The power of comparisons for actively learning linear classifiers. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.
- Adam Tauman Kalai and Varun Kanade. Efficient learning with arbitrary covariate shift. In *Algorithmic Learning Theory*, pages 850–864. PMLR, 2021.
- Adam Tauman Kalai, Varun Kanade, and Yishay Mansour. Reliable agnostic learning. *Journal of Computer and System Sciences*, 78(5):1481–1495, 2012.
- Michael J Kearns and Umesh Vazirani. *An introduction to computational learning theory*. MIT press, 1994.
- Adam Klivans, Konstantinos Stavropoulos, and Arsen Vasilyan. Testable learning with distribution shift. In *The Thirty Seventh Annual Conference on Learning Theory*, pages 2887–2943. PMLR, 2024a.

- Adam Klivans, Konstantinos Stavropoulos, Kevin Tian, and Arsen Vasilyan. The power of iterative filtering for supervised learning with (heavy) contamination. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL <https://openreview.net/forum?id=E7knuYAvpt>.
- Adam R Klivans and Alexander A Sherstov. Cryptographic hardness for learning intersections of halfspaces. *Journal of Computer and System Sciences*, 75(1):2–12, 2009.
- Adam R Klivans, Konstantinos Stavropoulos, and Arsen Vasilyan. Learning intersections of halfspaces with distribution shift: Improved algorithms and sq lower bounds. *The Thirty Seventh Annual Conference on Learning Theory*, 2024b.
- Pravesh K Kothari and Roi Livni. Improper learning by refuting. In *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*, pages 55–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018.
- Jane Lange and Mingda Qiao. Limitations of membership queries in testable learning. *arXiv preprint arXiv:2512.02279*, 2025.
- Philip M Long and Rocco A Servedio. Martingale boosting. In *International Conference on Computational Learning Theory*, pages 79–94. Springer, 2005.
- Yishay Mansour, Mehryar Mohri, and Afshin Rostamizadeh. Domain adaptation: Learning bounds and algorithms. In *Proceedings of The 22nd Annual Conference on Learning Theory (COLT 2009)*, Montréal, Canada, 2009. URL <http://www.cs.nyu.edu/~mohri/postscript/nadap.pdf>.
- Ievgen Redko, Emilie Morvant, Amaury Habrard, Marc Sebban, and Younès Bennani. A survey on domain adaptation theory: learning bounds and theoretical guarantees. *arXiv preprint arXiv:2004.11829*, 2020.
- Ronald L. Rivest and Robert Sloan. Learning complicated concepts reliably and usefully (extended abstract). In *Proceedings of the Seventh AAAI National Conference on Artificial Intelligence*, AAAI’88, page 635–640. AAAI Press, 1988.
- Ronitt Rubinfeld and Arsen Vasilyan. Testing distributional assumptions of learning algorithms. *Proceedings of the fifty-fifth annual ACM Symposium on Theory of Computing*, 2023.
- Lucas Slot, Stefan Tiegel, and Manuel Wiedmer. Testably learning polynomial threshold functions. *Advances in Neural Information Processing Systems*, 37:3781–3831, 2024.
- Stefan Tiegel. Improved hardness results for learning intersections of halfspaces. In *The Thirty Seventh Annual Conference on Learning Theory*, pages 4764–4786. PMLR, 2024.
- Salil Vadhan. On learning vs. refutation. In *Conference on Learning Theory*, pages 1835–1848. PMLR, 2017.

Appendix A. Useful Facts from Probability Theory

Lemma 13 (Reverse Markov Inequality) *Suppose that x is a random variable satisfying $x \leq B$ and $\mathbf{E}[x] \geq 0$, then*

$$\Pr \left[x \geq \frac{1}{2} \mathbf{E}[x] \right] \geq \frac{1}{2B} \mathbf{E}[x]$$

Proof Note that

$$\mathbf{E}[x] \leq \Pr \left[x \geq \frac{1}{2} \mathbf{E}[x] \right] \cdot B + \frac{1}{2} \mathbf{E}[x].$$

Rearranging yields the result. ■

We will also need the VC inequality.

Theorem 14 (VC Inequality) *Let \mathcal{C} denote a concept class over \mathcal{X} , \mathcal{D} be a distribution over \mathcal{X} , and $\varepsilon \in (0, 1)$. If d is the VC dimension of \mathcal{C} and $\mathbf{S} \sim \mathcal{D}$ is a set of size m , then*

$$\Pr_{\mathbf{S}} \left[\sup_{c \in \mathcal{C}} \left| \Pr_{x \sim \mathcal{D}}[c(x) = 1] - \frac{1}{|\mathbf{S}|} \sum_{s \in \mathbf{S}} \mathbb{I}\{c(s) = 1\} \right| \geq \varepsilon \right] \leq 8 \cdot \left(\frac{em}{d} \right)^d \cdot e^{-m\varepsilon^2/32}$$

Corollary 15 *Let \mathcal{C} denote a concept class over \mathcal{X} , \mathcal{D} be a distribution over \mathcal{X} , and $\varepsilon, \delta \in (0, 1/3)$. If d is the VC dimension of \mathcal{C} and $\mathbf{S} \sim \mathcal{D}$ is a set of size $\Omega\left(\frac{d \log(1/\varepsilon\delta)}{\varepsilon^2}\right)$, then with probability $1 - \delta$,*

$$\left| \Pr_{x \sim \mathcal{D}}[c(x) = 1] - \frac{1}{m} \sum_{s \in \mathbf{S}} \mathbb{I}\{c(s) = 1\} \right| \leq \varepsilon$$

for all $c \in \mathcal{C}$.

Appendix B. Halfspaces and Forster Transform

Recall that we call a halfspace homogeneous if is of the form $\text{sign}(w \cdot x)$, i.e. it goes through the origin. A crucial notion will be

Definition 16 (Margin) *Given a halfspace $f(x) = \text{sign}(w \cdot x)$ and a point x , we say that the margin of x with respect to f is*

$$\frac{|w \cdot x|}{\|w\|_2 \|x\|_2}$$

Another notion we'll need is

Definition 17 *We say that a set of points $S \subseteq \mathbb{S}^{n-1}$ is in ε -approximate radial isotropic position if*

$$\frac{1 - \varepsilon}{n} I_n \preceq \frac{1}{|S|} \sum_{x \in S} x x^T \preceq \frac{1 + \varepsilon}{n} I_n$$

Notably, such a condition is useful as it implies an anti-concentration bound for any halfspace. In particular,

Lemma 18 *Suppose that $S \subseteq \mathbb{S}^{n-1}$ is a set of points in $1/2$ -approximate radial isotropic position and $w \in \mathbb{S}^{n-1}$, then*

$$\Pr_{\mathbf{x} \sim S} \left[|w \cdot \mathbf{x}| \geq \frac{1}{2\sqrt{n}} \right] \geq \frac{1}{4n}$$

Proof Note that

$$\frac{1}{2n} \leq w^T \mathbf{E}_{\mathbf{x} \sim S} [\mathbf{x}\mathbf{x}^T] w = \mathbf{E}_{\mathbf{x} \sim S} [(w \cdot \mathbf{x})^2]$$

Since $(w \cdot \mathbf{x})^2 \leq \|w\|_2^2 \|\mathbf{x}\|_2^2 = 1$ by Cauchy Schwartz, the result follows from the Reverse Markov inequality applied to $(w \cdot \mathbf{x})^2$. \blacksquare

Crucially, there exists efficient algorithms to place points into ε -radial isotropic position.

Theorem 19 (Diakonikolas et al. (2023b)) *There exists a randomized algorithm that given a set $S \subseteq \mathbb{R}^n$ of size m and an $\varepsilon \in (0, 1)$, runs in time strongly polynomial in mn/ε and has the following high probability guarantee: either the algorithm finds a matrix A such that $\{Ax/\|Ax\| : x \in S\}$ is in ε -approximate radial isotropic position, or it finds a proper subspace $W \subsetneq \mathbb{R}^n$ such that $|S \cap W| > (\dim(W)/n) \cdot |S|$.*

Corollary 20 *Given a set of points $S \subseteq \mathbb{R}^n$ and parameter $\delta \in (0, 1)$, there exists an algorithm $\text{FORSTER}(S, \delta)$ that outputs a subspace V and matrix $A : V \rightarrow V$ such that $\{AP_V x/\|AP_V x\| : x \in V\}$ is in ε -approximate radial isotropic position in $\mathbb{R}^{\dim(V)}$, where $P_V : \mathbb{R}^n \rightarrow V$ denotes the projection matrix onto V . Moreover, the algorithm runs in time strongly polynomial in $mn \log(1/\delta)$ and $|S \cap V| > (\dim(V)/n) \cdot |S|$.*

Proof We now start by noting that we can boost the failure probability of the algorithm in **Theorem 19** to δ/n by repeating it $\log(n/\delta)$ times and checking whether the points are in radial isotropic condition or the subspace W contains enough points. As such, we'll assume throughout that it has failure probability δ/n .

Our procedure is then the following: Let $V = \mathbb{R}^n$ and $S' = S$. Loop at most n times: Run the algorithm from **Theorem 19** on $S' \subseteq V$ with $\varepsilon = 1/2$ to either get a matrix A or subspace $W \subsetneq V$. If we get a matrix, output A, V . Otherwise, update $V \leftarrow W$ and $S' \leftarrow P_W(S' \cap W)$.

We claim that if the first n calls to the algorithm in **Theorem 19** all succeed, then the above algorithm satisfies the corollary. Indeed, the process will eventually return a matrix since the dimension of V decreases in each iteration. Thus, we indeed have that $\{AP_V x/\|AP_V x\| : x \in S \cap V\}$ is in $1/2$ -radial isotropic position. Moreover, if $\mathbb{R}^n := W_0 \supseteq W_1 \supseteq W_2 \cdots \supseteq W_\ell$ are the subspaces in each iteration of the loop, then

$$|W_\ell \cap S| > \frac{\dim(W_\ell)}{\dim(W_{\ell-1})} |W_{\ell-1} \cap S| \geq |S| \prod_{i=1}^{\ell} \frac{\dim(W_i)}{\dim(W_{i-1})} = \frac{\dim(W_\ell)}{n} \cdot |S|.$$

Finally, the statement on the runtime follows from the polynomial runtime of **Theorem 19**. \blacksquare

Remark 21 *Note that applying a Forster transform preserves the property of a point being labeled by a homogeneous halfspace. In particular, let $S \subseteq \mathbb{R}^n$ and suppose that we are given labeled pairs (x, y) , where $x \in S$ and $y = \text{sign}(w \cdot x)$ for some $w \in \mathbb{R}^n$. We then observe that for $x \in V$ we have that the labeled point set $\left(\frac{AP_V x}{\|AP_V x\|}, y \right)$, where $y = \text{sign}(w \cdot x)$, are labeled by a halfspace in the transformed space, namely $\text{sign} \left((A^{-1})^T P_V w \cdot \frac{A_i P_V x}{\|A_i P_V x\|} \right)$.*

Appendix C. Missing Proofs from Section 3.1

Throughout this section, we will denote by $\mathcal{A}(\bar{\mathbf{S}}_{\text{train}}, \mathbf{X}_{\text{test}})$ the result of running \mathcal{A} on the appropriate number of labeled samples $\bar{\mathbf{S}}_{\text{train}}$ drawn from $\bar{\mathcal{D}}_{\text{train}}$ and a test set \mathbf{X}_{test} of m samples drawn from $\mathcal{D}^{\text{test}}$. We now start by proving the following lemma.

Lemma 22 *Consider distributions $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$ and let \mathcal{A} be a rejecting algorithm drawing m samples from $\mathcal{D}^{\text{test}}$ and $\delta \in (0, 1/3)$, then with probability $1 - \delta/3$, $\text{GETWEAKDISTINGUISHER}(\mathcal{D}^{\text{train}}, \mathcal{D}^{\text{test}}, \mathcal{A}, \delta)$ will find a set \bar{T} such that*

$$\Pr_{\mathbf{X} \sim (\mathcal{D}^{\text{train}})^{\otimes m}} [\mathcal{A}(\bar{T}, \mathbf{X}) \text{ accepts}] - \Pr_{\mathbf{X} \sim (\mathcal{D}^{\text{test}})^{\otimes m}} [\mathcal{A}(\bar{T}, \mathbf{X}) \text{ accepts}] \geq 0.003$$

Proof We'll assume that

$$\left| \hat{p}_1 - \Pr_{\mathbf{X} \sim (\mathcal{D}^{\text{train}})^{\otimes m}} [\mathcal{A}(\bar{\mathbf{S}}_{\text{train}}, \mathbf{X}) \text{ accepts}] + \Pr_{\mathbf{X}_{\text{test}} \sim (\mathcal{D}^{\text{test}})^{\otimes m}} [\mathcal{A}(\bar{\mathbf{S}}_{\text{train}}, \mathbf{X}) \text{ accepts}] \right| \leq 0.001$$

in every iteration of the loop on Line 4, which holds with probability at least $1 - \delta/6$ by a Hoeffding bound. Notably, when this event occurs, we will output a set T satisfying the statement of the lemma so long as $T \neq \emptyset$. Thus, it remains to show that the probability of $T \neq \emptyset$ is at most $\delta/6$.

Since \mathcal{A} is rejecting and rejects samples from $\mathcal{D}^{\text{train}}$ with probability at most 0.01, we have that

$$\Pr_{\bar{\mathbf{S}}_{\text{train}}, \mathbf{X} \sim (\mathcal{D}^{\text{train}})^{\otimes m}} [\mathcal{A}(\bar{\mathbf{S}}_{\text{train}}, \mathbf{X}) \text{ accepts}] - \Pr_{\bar{\mathbf{S}}_{\text{train}}, \mathbf{X} \sim (\mathcal{D}^{\text{test}})^{\otimes m}} [\mathcal{A}(\bar{\mathbf{S}}_{\text{train}}, \mathbf{X}) \text{ accepts}] \geq 0.01$$

So we conclude that

$$\mathbb{E}_{\bar{\mathbf{S}}_{\text{train}}} \left[\Pr_{\mathbf{X} \sim (\mathcal{D}^{\text{train}})^{\otimes m}} [\mathcal{A}(\bar{\mathbf{S}}_{\text{train}}, \mathbf{X}) \text{ accepts}] - \Pr_{\mathbf{X} \sim (\mathcal{D}^{\text{test}})^{\otimes m}} [\mathcal{A}(\bar{\mathbf{S}}_{\text{train}}, \mathbf{X}) \text{ accepts}] \right] \geq 0.01$$

Since the quantity in the expectation is at most 1, it follows by reverse Markov that

$$\Pr_{\bar{\mathbf{S}}_{\text{train}}} \left[\Pr_{\mathbf{X} \sim (\mathcal{D}^{\text{train}})^{\otimes m}} [\mathcal{A}(\bar{\mathbf{S}}_{\text{train}}, \mathbf{X}) \text{ accepts}] - \Pr_{\mathbf{X} \sim (\mathcal{D}^{\text{test}})^{\otimes m}} [\mathcal{A}(\bar{\mathbf{S}}_{\text{train}}, \mathbf{X}) \text{ accepts}] \geq 0.005 \right] \geq 0.005$$

We now note that if we draw a set $\bar{\mathbf{S}}_{\text{train}}$ satisfying the event above, then $\bar{T} \neq \emptyset$. However, since the algorithm repeats $C' \log(1/\delta)$ times, this happens with probability at most $\delta/6$, completing the proof. ■

Proof of Lemma 5. We will assume throughout that

$$\left| \hat{p}_2 - \Pr_{\mathbf{x} \sim \mathcal{D}^{\text{train}}} [\mathcal{A}'_{a,b}(\mathbf{x}) \text{ accepts}] + \Pr_{\mathbf{x} \sim \mathcal{D}^{\text{test}}} [\mathcal{A}'_{a,b}(\mathbf{x}) \text{ accepts}] \right| \leq \frac{1}{10^5 m}$$

in every iteration of the loops on Line 11, which holds by a Hoeffding bound with probability at least $1 - \delta/3$. Moreover, we will assume that the set \bar{T} satisfies

$$\Pr_{\mathbf{X} \sim (\mathcal{D}^{\text{train}})^{\otimes m}} [\mathcal{A}(\bar{T}, \mathbf{X}) \text{ accepts}] - \Pr_{\mathbf{X} \sim (\mathcal{D}^{\text{test}})^{\otimes m}} [\mathcal{A}(\bar{T}, \mathbf{X}) \text{ accepts}] \geq 0.003$$

which again happens with probability $1 - \delta/3$ by Lemma 22. Assuming both these events hold, it remains to prove that we output FAIL with probability at most $\delta/3$.

Algorithm 3 GETWEAKDISTINGUISHER($\mathcal{D}^{\text{train}}, \mathcal{D}^{\text{test}}, \mathcal{A}, \delta$)

Input: Sample access to $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$, a rejecting TDS learning algorithm \mathcal{A} that uses at most m samples from $\mathcal{D}^{\text{test}}$, and failure probability $\delta \in (0, 1)$

Output: An algorithm \mathcal{A}' that distinguishes $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$ with advantage $\Omega(1/m)$, or FAIL

3 Let $C \geq C' \geq 1$ be sufficiently large constants and set $\bar{T} \leftarrow \emptyset$, $r \leftarrow C \log(1/\delta)$, and $\ell \leftarrow Cm^2 \log(m/\delta)$

/* Find a training set \bar{T} witnessing a noticeable acceptance gap
*/

4 **for** $t = 1, 2, \dots, C' \log(1/\delta)$ **do**

5 Sample a set $\bar{\mathbf{S}}_{\text{train}}$ from $\bar{\mathcal{D}}_{\text{train}}$

6 Sample sets $\mathbf{X}_{\text{test}}^{(1)}, \dots, \mathbf{X}_{\text{test}}^{(r)}$ and $\mathbf{X}_{\text{train}}^{(1)}, \dots, \mathbf{X}_{\text{train}}^{(r)}$, each of size m , from $\mathcal{D}^{\text{test}}$ and $\mathcal{D}^{\text{train}}$ respectively

7 Compute

$$\hat{p}_1 := \frac{1}{r} \sum_{i=1}^r \left(\mathbb{1}\{\mathcal{A}(\bar{\mathbf{S}}_{\text{train}}, \mathbf{X}_{\text{train}}^{(i)}) \text{ accepts}\} - \mathbb{1}\{\mathcal{A}(\bar{\mathbf{S}}_{\text{train}}, \mathbf{X}_{\text{test}}^{(i)}) \text{ accepts}\} \right);$$

if $\hat{p}_1 \geq 0.004$ **then**

8 | Set $\bar{T} \leftarrow \bar{\mathbf{S}}_{\text{train}}$ and **break**

9 **end**

10 **end**

/* Search for a single-sample weak distinguisher

*/

11 **for** $i = 1, 2, \dots, m$ **do**

12 **for** $t = 1, 2, \dots, C' m \log(1/\delta)$ **do**

13 | Sample $\mathbf{a}_1, \dots, \mathbf{a}_{m-i} \sim \mathcal{D}^{\text{train}}$

14 | Sample $\mathbf{b}_1, \dots, \mathbf{b}_{i-1} \sim \mathcal{D}^{\text{test}}$

15 | Define the (randomized) algorithm $\mathcal{A}'_{\mathbf{a}, \mathbf{b}}$: on input $x \in \mathbb{R}^n$, output

$$\mathcal{A}(\bar{T}, \{\mathbf{b}_1, \dots, \mathbf{b}_{i-1}, x, \mathbf{a}_1, \dots, \mathbf{a}_{m-i}\});$$

16 | Sample $\mathbf{c}_1, \dots, \mathbf{c}_\ell \sim \mathcal{D}^{\text{train}}$ and $\mathbf{d}_1, \dots, \mathbf{d}_\ell \sim \mathcal{D}^{\text{test}}$

17 | Compute

$$\hat{p}_2 := \frac{1}{\ell} \sum_{j=1}^{\ell} \left(\mathbb{1}\{\mathcal{A}'_{\mathbf{a}, \mathbf{b}}(\mathbf{c}_j) \text{ accepts}\} - \mathbb{1}\{\mathcal{A}'_{\mathbf{a}, \mathbf{b}}(\mathbf{d}_j) \text{ accepts}\} \right);$$

if $\hat{p}_2 \geq \frac{1}{5000m}$ **then**

18 | **return** $\mathcal{A}'_{\mathbf{a}, \mathbf{b}}$

19 **end**

20 **end**

21 **end**

22 **return** FAIL

For brevity, we will assume that \mathcal{A} outputs 1 when it accepts and 0 when it rejects. To start, we argue that there exists an $i^* \in [m]$ such that

$$\Pr[\mathcal{A}(\bar{T}, \{\mathbf{b}_1, \dots, \mathbf{b}_{i^*-1}, \mathbf{a}_{i^*}, \dots, \mathbf{a}_m\}) = 1] - \Pr[\mathcal{A}(1, \{\mathbf{b}_1, \dots, \mathbf{b}_{i^*}, \mathbf{a}_{i^*+1}, \dots, \mathbf{a}_m\}) = 1] \geq \frac{1}{500m} \quad (1)$$

where $\mathbf{a}_1, \dots, \mathbf{a}_m \sim \mathcal{D}^{\text{train}}$ and $\mathbf{b}_1, \dots, \mathbf{b}_m \sim \mathcal{D}^{\text{test}}$. To see this, we apply a hybrid argument.

$$\begin{aligned} 0.003 &\leq \Pr[\mathcal{A}(\bar{T}, \{\mathbf{a}_1, \dots, \mathbf{a}_m\}) = 1] - \Pr[\mathcal{A}(\bar{T}, \{\mathbf{b}_1, \dots, \mathbf{b}_m\}) = 1] \\ &= \sum_{i=1}^m \Pr[\mathcal{A}(\bar{T}, \{\mathbf{b}_1, \dots, \mathbf{b}_{i-1}, \mathbf{a}_i, \dots, \mathbf{a}_m\}) = 1] - \Pr[\mathcal{A}(\bar{T}, \{\mathbf{b}_1, \dots, \mathbf{b}_i, \mathbf{a}_{i+1}, \dots, \mathbf{a}_m\}) = 1]. \end{aligned}$$

An averaging argument then implies that an i^* as in [Equation \(1\)](#) exists.

We now consider the iteration of the loop on [Line 11](#) where $i = i^*$. Note that for random variables $\mathbf{c}, \mathbf{a}_1, \dots, \mathbf{a}_{i-1} \sim \mathcal{D}^{\text{train}}$ and $\mathbf{d}, \mathbf{b}_1, \dots, \mathbf{b}_{m-i} \sim \mathcal{D}^{\text{test}}$, we have that

$$\begin{aligned} &\mathbf{E}_{\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}}[\mathcal{A}'_{\mathbf{a}, \mathbf{b}}(\mathbf{c}) - \mathcal{A}'_{\mathbf{a}, \mathbf{b}}(\mathbf{d})] \\ &= \Pr[\mathcal{A}(\bar{T}, \{\mathbf{b}_1, \dots, \mathbf{b}_{i^*-1}, \mathbf{c}, \mathbf{a}_1, \dots, \mathbf{a}_{m-i}\}) = 1] - \Pr[\mathcal{A}(\bar{T}, \{\mathbf{b}_1, \dots, \mathbf{b}_{i-1}, \mathbf{d}, \mathbf{a}_1, \dots, \mathbf{a}_{m-i}\}) = 1] \\ &\geq \frac{1}{500m} \end{aligned}$$

On the other hand, we have

$$\mathbf{E}_{\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}}[\mathcal{A}'_{\mathbf{a}, \mathbf{b}}(\mathbf{c}) - \mathcal{A}'_{\mathbf{a}, \mathbf{b}}(\mathbf{d})] = \mathbf{E}_{\mathbf{a}, \mathbf{b}}[\mathbf{E}_{\mathbf{c}, \mathbf{d}}[\mathcal{A}'_{\mathbf{a}, \mathbf{b}}(\mathbf{c}) - \mathcal{A}'_{\mathbf{a}, \mathbf{b}}(\mathbf{d})]]$$

Since $\left| \mathbf{E}_{\mathbf{c}, \mathbf{d}}[\mathcal{A}'_{\mathbf{a}, \mathbf{b}}(\mathbf{c}) - \mathcal{A}'_{\mathbf{a}, \mathbf{b}}(\mathbf{d})] \right|$ is at most 1, reverse Markov then yields that with probability $1/1000m$ over \mathbf{a}, \mathbf{b} , we have that

$$\Pr_{\mathbf{x} \sim \mathcal{D}^{\text{train}}}[\mathcal{A}'_{\mathbf{a}, \mathbf{b}}(\mathbf{x}) = 1] - \Pr_{\mathbf{x} \sim \mathcal{D}^{\text{test}}}[\mathcal{A}'_{\mathbf{a}, \mathbf{b}}(\mathbf{x}) = 1] \geq \frac{1}{1000m}.$$

Note that if we sampled such values \mathbf{a}, \mathbf{b} then we would output $\mathcal{A}'_{\mathbf{a}, \mathbf{b}}$ as

$$\hat{p}_2 \geq \frac{1}{1000m} - \frac{1}{10^5 m} \geq \frac{1}{2000m}.$$

by our assumption on the accuracy of \hat{p}_2 . Thus the probability of failure is bounded by

$$\left(1 - \frac{1}{1000m}\right)^{C'm \log(1/\delta)} \leq \delta/3$$

as desired. ■

Appendix D. Missing Proofs from [Section 3.2](#)

Proof of [Theorem 6](#). We assume that we are in the event that happens with probability at least $1 - \delta$ such that the following conditions hold:

- Our estimates $\hat{p}_{i,t}^b, \hat{a}_{i,t}, \hat{q}_{i,t}$ are within $\gamma_1, 0.01, \gamma_2$ of their expected values respectively.
- Any time we call [Algorithm 3](#), it outputs a valid weak distinguisher.
- The majority boosted TDS learner never fails when called, i.e., it outputs **ACCEPT** and gives a hypothesis $h_{i,t}$ with error at most 4ε on $\mathcal{G}_{i,t}^0$.

For the first property, we may use standard Hoeffding bounds to show that our estimates — which we obtain through averaging — are concentrated around their expectations. To show that the last property holds with high probability, we combine Proposition C.1 of [Klivans et al. \(2024a\)](#) with the observation that majority boosting amplifies the probability of acceptance as long as the probability of acceptance is larger than $1/2$.

We will first bound the rejection rate on the training distribution. We have the following:

$$\begin{aligned} \Pr_{\mathbf{x} \sim \mathcal{D}_1, P}[\hat{g}(\mathbf{x}) = 0] &\leq \Pr_{\mathbf{x} \sim \mathcal{D}_1, P}[\exists N \in \text{leaves}(P) : \mathbf{x} \text{ reaches } N \wedge \text{label}(N) = 0] \\ &\leq (p_{\min} + \gamma_1)T^2 + \varepsilon && \text{(first and second type leaves)} \\ &\leq O(\varepsilon) \end{aligned}$$

To obtain the above bound we used the property that each leaf $N \in \mathcal{T}_1$ with $\text{label}(i, t) = 0$ satisfies $\Pr_{\mathbf{x} \sim \mathcal{D}_1, P}[\mathbf{x} \text{ reaches } N] \leq p_{\min} + \gamma_1$, as well as [Lemma 9](#) for leaves of the form (i, T) .

For the error rate we obtain similar bounds for the contributions of the first and second types of leaf nodes, but we also need to account for the leaves of third type. Fortunately, any such node N corresponds to hypothesis with low error on the test distribution conditioned on reaching node N . We have the following:

$$\begin{aligned} \Pr_{\mathbf{x} \sim \mathcal{D}_0, P}[\hat{h}(\mathbf{x}) \neq f(\mathbf{x}) \wedge \hat{g}(\mathbf{x}) = 1] &\leq \Pr_{\mathbf{x} \sim \mathcal{D}_0, P}[\exists N \in \text{leaves}(P) : \mathbf{x} \text{ reaches } N \wedge \text{label}(N) = 1 \wedge h_N(\mathbf{x}) \neq f(\mathbf{x})] \\ &\leq (p_{\min} + \gamma_1)T^2 + \varepsilon + 4\varepsilon && \text{(first, second and third type leaves)} \\ &\leq O(\varepsilon) \end{aligned}$$

Finally, the bound on the run time follows from the fact that the boosting procedure runs in time $\text{poly}(1/\gamma, \frac{1}{\varepsilon}, \log(1/\delta), m_{\mathcal{A}}) \cdot T(\mathcal{C}, \varepsilon)$, where $\gamma = \Omega(1/m_{\mathcal{A}})$ is the advantage of the weak learner from [Lemma 5](#). ■

Appendix E. Deferred Proofs from [Section 4](#)

E.1. A Reduction to Homogeneous Halfspaces

In this section, we show that PQ learning halfspaces with a margin reduces to a restricted class of query algorithms for homogeneous halfspaces. In particular, we define

Definition 23 (Respectful Algorithm) *Suppose that we are given a query algorithm \mathcal{A} to PQ learn a halfspace $f(x) := \text{sign}(w \cdot x)$ in \mathbb{R}^n . We say that \mathcal{A} is respectful if whenever $\mathcal{D}^{\text{train}}$ is supported on $\{(x, 1) : x \in \mathbb{R}^{n-1}\}$, we have that \mathcal{A} almost surely never queries a point of the form $(x, 0)$ for some $x \in \mathbb{R}^{n-1}$.*

We then have that

Lemma 24 *Suppose that there exists a respectful PQ learning algorithm for homogeneous halfspaces over \mathbb{R}^n with membership queries that runs in time $\text{poly}(n, \frac{1}{\epsilon}, \log(1/\delta))$, then [Theorem 11](#) holds.⁴*

Proof Let \mathcal{A} denote our respectful algorithm for homogeneous halfspaces and suppose that we are tasked with learning an unknown halfspace $f(x) = \text{sign}(w \cdot x - \theta)$ with distributions $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$. Let Φ be the map that sends a point $x \in \mathbb{R}^n$ to $(1, x) \in \mathbb{R}^{n+1}$ and denote by $\Phi(\mathcal{D}^{\text{train}})$ and $\Phi(\mathcal{D}^{\text{test}})$ denote the distributions resulting from drawing from $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$ respectively and then applying Φ . Note that after applying this transformation, the target halfspace can be written as $\text{sign}((w, -\theta) \cdot \Phi(x))$ and thus correspond to a homogeneous halfspace after our transformation.

We will now run \mathcal{A} on $\Phi(\mathcal{D}^{\text{train}})$ and $\Phi(\mathcal{D}^{\text{test}})$. To do so, we must argue that we can simulate membership queries in this new space. Indeed, suppose that \mathcal{A} queries a point (x, c) for some $c \in \mathbb{R}$ and $x \in \mathbb{R}^n$. If $c = 0$, we simply output FAIL. Otherwise, we can output $f(x/c) \cdot \text{sign}(c)$. Since

$$f(x/c) \cdot \text{sign}(c) = \text{sign}(w \cdot (x/c) - \theta) \cdot \text{sign}(c) = \text{sign}(w \cdot x - c\theta) = \text{sign}((w, -\theta) \cdot (x, c)),$$

this correctly answers the membership query.

We now note that we fail with measure zero as \mathcal{A} is respectful, thus this process almost surely outputs a hypothesis h and selector g . Using $h \circ \Phi$ and $g \circ \Phi$, then yields the hypothesis and selector for PQ learning f . ■

E.2. PQ Learning Homogeneous Halfspaces with a Margin

We now turn to showing how to PQ learn a homogenous halfspaces, i.e. those of the form $f(x) = \text{sign}(w \cdot x)$, with a margin (cf. [Definition 17](#)). As is standard in this setting, we will assume without loss of generality that $\mathcal{D}^{\text{train}}$ and $\mathcal{D}^{\text{test}}$ are supported on the unit sphere \mathbb{S}^{n-1} and $\|w\|_2 = 1$.

Algorithm 4 LEARN-HIGH-MARGIN-HALFSPACE(f, γ, δ)

Input: Query access to f , a margin parameter $\gamma \in (0, 1)$, and failure parameter $\delta \in (0, 1/3)$

Output: A hypothesis $h : \mathcal{X} \rightarrow \{\pm 1\}$ and a selector $g : \mathcal{X} \rightarrow \{0, 1\}$

- 1 Let $\ell \leftarrow \frac{2000n}{\gamma^2} \log(\frac{n}{\delta})$
 - 2 Sample $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(\ell)} \sim \mathcal{N}(0, \frac{\pi}{2} I_n)$
 - 3 Compute $\hat{w} \leftarrow \frac{1}{\ell} \sum_{i=1}^{\ell} f(\mathbf{x}^{(i)}) \mathbf{x}^{(i)}$
 - 4 Set $h(x) \leftarrow \text{sign}(\hat{w} \cdot x)$ and $g(x) \leftarrow \mathbb{I}(|\hat{w} \cdot x| \geq \frac{2\gamma}{3})$
 - 5 **return** (h, g)
-

Our main lemma in this section will then be

4. We quickly remark that while we will work in the real model of computation for simplicity, we note that if there is a respectful homogenous halfspace PQ learning algorithm that only queries points with polynomial bit complexity then the algorithm obtained for arbitrary halfspaces via the reduction will also only query points of polynomial bit complexity.

Lemma 25 *Let f denote an unknown halfspace, then with probability at least $1 - \delta$, we have that $\text{LEARN-HIGH-MARGIN-HALFSPACE}(f, \gamma, \delta)$ outputs a hypothesis h such that for all for all $x \in \mathbb{S}^{n-1}$ (i) if x has margin at least γ with respect to f , then $g(x) = 1$ and (ii) if $g(x) = 1$ then $h(x) = f(x)$ and (iii) $\|\widehat{w}\| \geq \frac{2}{3}$*

Proof We will show that the lemma follows from the following claim

Claim 1

$$\Pr[\|\widehat{w} - w\|_2 \geq \gamma/3] \leq \delta$$

Proof By rotational symmetric, we can let $w = e_1$ without loss of generality. We then note that

$$\mathbf{E}[f(\mathbf{x}^{(i)}) \cdot \mathbf{x}^{(i)}] = e_1$$

Indeed, by symmetry the expectation of of the j th coordinate is 0 for all $j \neq 1$. We can then compute the first expectation of the first coordinate as

$$\mathbf{E}_{\mathbf{z} \sim \mathcal{N}(0, \frac{\pi}{2})}[|\mathbf{z}|] = \sqrt{\frac{2}{\pi}} \cdot \sqrt{\frac{\pi}{2}} = 1$$

We now note that for all $j \in [n]$, $(f(\mathbf{x}^{(i)}) \cdot \mathbf{x}^{(i)} - e_i)_j$ is sub-gaussian with variance proxy at most 10. Thus it follows that for any j , we have that

$$\Pr[|\widehat{w}_j - (e_i)_j| \geq \gamma/3\sqrt{n}] \leq \exp(-\ell\gamma^2/2000n) \leq \frac{\delta}{n}$$

Thus, applying a union bound then yields that for all $j \in [n]$

$$|\widehat{w}_j - (e_i)_j| \leq \gamma/3\sqrt{n}$$

with probability $1 - \delta$. Notably, this implies that

$$\|\widehat{w} - e_i\|_2 \leq \gamma/3$$

with probability at least $1 - \delta$ as desired. ■

We will now show that conditioned on $\|\widehat{w} - w\|_2 \leq \gamma/3$, properties (i), (ii), and (iii) hold. Indeed, the triangle inequality immediately implies $\widehat{w} \geq 1 - \gamma/3 \geq \frac{2}{3}$. Next, for property (i), note that

$$|w \cdot x - \widehat{w} \cdot x| \leq \|w - \widehat{w}\|_2 \leq \gamma/3$$

Thus it follows that if $|w \cdot x| \geq \gamma$ then $|\widehat{w} \cdot x| \geq 2\gamma/3$ and thus $g(x) = 1$. Similarly, we can observe that if $|\widehat{w} \cdot x| \geq 2\gamma/3$, we have that $\text{sign}(w \cdot x) = \text{sign}(\widehat{w} \cdot x)$, yielding property (ii). ■

Algorithm 5 LEARN-HALFSPACE($\mathbf{S}_{\text{train}}, \varepsilon, \delta$)

Input: A set examples $\mathbf{S}_{\text{train}}$ drawn i.i.d. from $\mathcal{D}^{\text{train}}$, a rejection parameter ε , and a failure probability δ

Output: A hypothesis $h : \mathcal{X} \rightarrow \{\pm 1\}$ and a selector $g : \mathcal{X} \rightarrow \{0, 1\}$

```

1  $R_0 \leftarrow \mathbf{S}_{\text{train}}, i \leftarrow 0$ 
2 Set  $g_0(x) \equiv 0$  and  $h_0(x) \equiv 1$ 
3 while  $\frac{|R_i|}{|\mathbf{S}_{\text{train}}|} \geq \frac{1}{2}\varepsilon$  and  $i < 4n \log \frac{2}{\varepsilon}$  do
4    $(A_i, V_i) \leftarrow \text{FORSTER}\left(R_i, \frac{\delta}{24n \log(2/\varepsilon)}\right)$ 
5   if  $R_i \cap V_i = \emptyset$  then
6     break
7   end
8    $(\ell_i, s_i) \leftarrow \text{LEARN-HIGH-MARGIN-HALFSPACE}\left(f(P_{V_i}^T A_i^{-1} x), \frac{1}{2\sqrt{n}}, \frac{\delta}{24n \log(2/\varepsilon)}\right)$ 
9    $R_{i+1} \leftarrow R_i \setminus \left\{x \in R_i \cap V_i : s_i\left(\frac{A_i P_{V_i} x}{\|A_i P_{V_i} x\|}\right) = 1\right\}$ 
10  Define
      
$$g_{i+1}(x) \leftarrow g_i(x) \vee \left(\mathbb{1}\{x \in V_i\} \wedge s_i\left(\frac{A_i P_{V_i} x}{\|A_i P_{V_i} x\|}\right)\right) \quad \text{for all } x$$

11  Define
      
$$h_{i+1}(x) \leftarrow \begin{cases} h_i(x) & \text{if } g_i(x) = 1, \\ \ell_i\left(\frac{A_i P_{V_i} x}{\|A_i P_{V_i} x\|}\right) & \text{otherwise} \end{cases}$$

12   $i \leftarrow i + 1$ 
13 end
14 return  $(h_i, g_i)$ 

```

E.3. Proof of Theorem 11

In this section, we give an algorithm to prove Theorem 11. To do so, our algorithm LEARN-HALFSPACE will apply a Forster transform to ensure that a non-trivial fraction of points have a large margin. We can then run LEARN-HIGH-MARGIN-HALFSPACE to classify these points. Afterwards, we can repeat these steps for the points in $\mathbf{S}_{\text{train}}$ that were not labeled.

To analyze LEARN-HALFSPACE, we start by showing that we typically terminate when we have labeled all but an $\varepsilon/2$ fraction of the data set.

Lemma 26 *With probability $1 - \delta/3$, when LEARN-HALFSPACE($\mathbf{S}_{\text{train}}, \varepsilon, \delta$) terminates we have that $\frac{|R_i|}{|\mathbf{S}_{\text{train}}|} \leq \frac{\varepsilon}{2}$.*

Proof We'll assume that no call to FORSTER or LEARN-HIGH-MARGIN-HALFSPACE fails, which by a union bound happens with probability at least $1 - \delta/3$. Notably, in this case by Corollary 20, we will never have $V_i \cap R_i = \emptyset$ and thus never execute Line 5.

Now applying [Lemma 18](#), it then follows that

$$\Pr_{\mathbf{x} \sim R_i \cap V_i} \left[\left| \frac{\hat{w}}{\|\hat{w}\|} \cdot \frac{A_i P_{V_i} \mathbf{x}}{\|A_i P_{V_i} \mathbf{x}\|} \right| \geq \frac{1}{2\sqrt{n}} \right] \geq \frac{1}{4 \dim(V_i)}$$

Since $\|\hat{w}\| \geq \frac{2}{3}$ by the promise of [Lemma 25](#), it follows that

$$\Pr_{\mathbf{x} \sim R_i \cap V_i} \left[\left| \hat{w} \cdot \frac{A_i P_{V_i} \mathbf{x}}{\|A_i P_{V_i} \mathbf{x}\|} \right| \geq \frac{1}{3\sqrt{n}} \right] \geq \frac{1}{4 \dim(V_i)}$$

Note that the points in $R_i \cap V_i$ satisfying the above event also satisfy $\mathbb{I}(x \in V_i) \wedge s_i \left(\frac{A_i P_{V_i} x}{\|A_i P_{V_i} x\|} \right)$. So, it follows that

$$\frac{|R_{i+1}|}{|R_i|} \leq 1 - \frac{1}{4 \dim(V_i)} \cdot \frac{|R_i \cap V_i|}{|R_i|} \leq 1 - \frac{1}{4 \dim(V_i)} \cdot \frac{\dim(V_i) |R_i|}{n |R_i|} = 1 - \frac{1}{4n}$$

The result now follows by noting that

$$\frac{|R_i|}{|\mathbf{S}_{\text{train}}|} \leq \left(1 - \frac{1}{4n} \right)^{4n \log(2/\varepsilon)} \leq \frac{\varepsilon}{2}$$

■

Lemma 27 Suppose that $\mathbf{S}_{\text{train}}$ is a set of size $\Omega \left(\frac{Tn^2 \log(nT) \log(1/\varepsilon\delta)}{\varepsilon^2} \right)$ samples from $\mathcal{D}^{\text{train}}$, then with probability at least $1 - \delta/2$ the hypothesis h outputted by `LEARN-HALFSPACE`($\mathbf{S}_{\text{train}}, \varepsilon, \delta$) satisfies

$$\Pr_{\mathbf{x} \sim \mathcal{D}^{\text{train}}} [g(\mathbf{x}) = 0] \leq \varepsilon$$

Proof Let T denote the number of iterations of the while loop. We will argue that g_1, \dots, g_T come from a class of VC dimension $O(Tn^2 \log(nT))$. Towards this, we first show that the function $\mathbb{I}(x \in V_i) \wedge s_i \left(\frac{A_i x}{\|A_i x\|} \right)$ has VC dimension at most $O(n^2)$. Indeed, the indicator of a subspace has VC dimension at most $O(n)$. Using the definition of s_i as appearing in `LEARN-HIGH-MARGIN-HALFSPACE`, we have that

$$s_i \left(\frac{A_i P_{V_i} x}{\|A_i P_{V_i} x\|} \right) = \mathbb{I} \left(\hat{w} \cdot \frac{A_i P_{V_i} x}{\|A_i P_{V_i} x\|} \geq \frac{1}{3\sqrt{n}} \right)$$

Notably, this can be rewritten as a conjunction of a degree 2 PTF and a halfspace, namely

$$\text{sign} \left((\hat{w} \cdot A_i P_{V_i} x)^2 - \frac{1}{9n} \cdot \|A_i P_{V_i} x\|^2 \right) \wedge \text{sign}(\hat{w} \cdot A_i P_{V_i} x).$$

As such, it follows that $\mathbb{I}(x \in V_i) \wedge s_i \left(\frac{A_i P_{V_i} x}{\|A_i P_{V_i} x\|} \right)$ is the conjunction of a subspace, halfspace, and degree 2 PTF. Using standard results, it thus follows that it has VC dimension $O(n^2)$ as desired. Since each g_i is a decision list of length at most T over features of the form $\mathbb{I}(x \in V_i) \wedge s_i \left(\frac{A_i x}{\|A_i x\|} \right)$, it follows from standard techniques that the functions g_i for $i < T$ have VC dimension at most

$O(Tn^2 \log(nT))$ (see e.g. Theorem 3.6 of Kearns and Vazirani (1994)). By Corollary 15, it then follows that with probability $1 - \delta/6$

$$\Pr_{x \sim \mathcal{D}^{\text{train}}}[g_T(x) = 0] = \frac{|R_T|}{|\mathbf{S}_{\text{train}}|} + \frac{\varepsilon}{2}.$$

Thus, if $\Pr_{x \sim \mathcal{D}^{\text{train}}}[g(x) = 0] > \varepsilon$ either $\frac{|R_T|}{|\mathbf{S}_{\text{train}}|} > \frac{\varepsilon}{2}$, which happens with probability at most $\delta/3$ by Lemma 26, or the VC inequality fails. So, by a union bound, the lemma statement holds with probability $1 - \delta/2$, as desired. ■

With this we can complete the proof of Theorem 11.

Proof of Theorem 11. We start by noting that the LEARN-HALFSPACE successfully PQ learns a homogeneous halfspace assuming $|\mathbf{S}_{\text{train}}|$ is sufficiently large so as to satisfy the hypothesis of Lemma 27. Indeed, with probability $1 - \delta/2$, the rejection probability of g under $\mathcal{D}^{\text{train}}$ is at most ε by Lemma 27. Moreover, so long as no call to LEARN-HIGH-MARGIN-HALFSPACE fails, which happens with probability at least $1 - \delta/6$ by a union bound, we have that our algorithm has zero error on points with $g_i(x) = 1$ by Lemma 25. In particular, LEARN-HIGH-MARGIN-HALFSPACE outputs a hypothesis ℓ_j such that $\ell_j(x) = \text{sign}(P_{V_j} w \cdot A_j^{-1} x)$ for all $x \in \mathbb{S}^{n-1} \cap V_j$ such that $s_j(x) = 1$. As such we conclude that $\ell_j \left(\frac{A_j P_{V_j} x}{\|A_j P_{V_j} x\|} \right) = \text{sign}(w \cdot x)$ for all $x \in V_j$ satisfying $s_j \left(\frac{A_j P_{V_j} x}{\|A_j P_{V_j} x\|} \right) = 1$. Thus, it follows that with probability $1 - \delta/6$, we have that the hypothesis we output, h_i , makes no errors on points where $g_i(x) = 1$, as desired.

To complete the proof of the theorem, it now remains to show that LEARN-HALFSPACE is respectful. Towards this, note that all queries are made in our calls to LEARN-HIGH-MARGIN-HALFSPACE. Notably, we query $f(P_{V_i}^T A_i^{-1} x)$ on points drawn from $\mathcal{N}(0, \frac{\pi}{2} I_{V_i})$. Since $R_i \cap V_i \neq \emptyset$, it follows that there must exist a point $z \in V_i$ such that $z_n \neq 0$. Thus, querying f on a point of the form $(x, 0)$ is a measure zero event and LEARN-HALFSPACE is respectful as desired. Using Lemma 24 then yields the result. ■

Appendix F. The Agnostic Setting

In our main theorem (Theorem 6), we have shown that PQ learning can be reduced to TDS learning. However, the premise of our result requires that the training and test distributions are both realizable by the same function $f \in \mathcal{C}$. We will now focus on the agnostic setting where neither the training nor the test labels need be realizable.

Definition 28 (Agnostic Setting (Ben-David et al., 2006; Blitzer et al., 2007)) *Let \mathcal{C} be a concept class over \mathcal{X} . In the agnostic setting, the learner has labeled sample access to a training distribution $\bar{\mathcal{D}}^{\text{train}}$, and unlabeled sample access to the \mathcal{X} -marginal of a test distribution $\bar{\mathcal{D}}^{\text{test}}$, and the goal is to produce a hypothesis whose on the test distribution competes with the following benchmark:*

$$\lambda = \lambda(\bar{\mathcal{D}}^{\text{train}}, \bar{\mathcal{D}}^{\text{test}}; \mathcal{C}) := \min_{f \in \mathcal{C}} \left(\Pr_{(x,y) \sim \bar{\mathcal{D}}^{\text{train}}}[f(x) \neq y] + \Pr_{(x,y) \sim \bar{\mathcal{D}}^{\text{test}}}[f(x) \neq y] \right)$$

The parameter λ quantifies the relationship between the training and test labels and, in the absence of test labels, a dependence of the output's test error on λ is unavoidable (Klivans et al., 2024a). We may now define the agnostic versions of TDS and PQ learning.

Definition 29 (Agnostic TDS Learning (Klivans et al., 2024a)) *Let \mathcal{C} be a concept class over \mathcal{X} . The algorithm \mathcal{A} is a TDS learner for \mathcal{C} up to error $O(\lambda) + \varepsilon$ and probability of failure δ in the agnostic setting if it either outputs **REJECT** or outputs **ACCEPT** and returns a hypothesis $h : \mathcal{X} \rightarrow \{\pm 1\}$ such that, with probability at least $1 - \delta$, the following hold:*

- (a) (soundness) *Upon acceptance, the test error is $\Pr_{(\mathbf{x}, \mathbf{y}) \sim \bar{\mathcal{D}}^{\text{test}}}[\mathbf{y} \neq h(\mathbf{x})] \leq A \cdot \lambda + \varepsilon$, where $A \geq 1$ is some constant.*
- (b) (completeness) *If $\mathcal{D}^{\text{train}} = \mathcal{D}^{\text{test}}$, then the algorithm accepts.*

Definition 30 (Agnostic PQ Learning (Goldwasser et al., 2020)) *Let \mathcal{C} be a concept class over \mathcal{X} . The algorithm \mathcal{A} is a PQ learner for \mathcal{C} up to error $O(\lambda) + \varepsilon$, rejection rate η and probability of failure δ in the agnostic setting if it outputs, with probability at least $1 - \delta$, a hypothesis $h : \mathcal{X} \rightarrow \{\pm 1\}$ and a selector $g : \mathcal{X} \rightarrow \{0, 1\}$ such that:*

- (a) (accuracy) *The test error is bounded as $\Pr_{\mathbf{x} \sim \mathcal{D}^{\text{test}}}[f(\mathbf{x}) \neq h(\mathbf{x}) \text{ and } g(\mathbf{x}) = 1] \leq A \cdot \lambda + \varepsilon$, where $A \geq 1$ is some constant.*
- (b) (rejection rate) *The probability of rejection is bounded as $\Pr_{\mathbf{x} \sim \mathcal{D}^{\text{train}}}[g(\mathbf{x}) = 0] \leq \eta$.*

The following theorem states that efficient agnostic TDS learning up to error $O(\lambda)$ implies efficient agnostic PQ learning with error rate $O(\lambda/\eta)$ and rejection rate $O(\eta)$, where η can be chosen freely. Note that no PQ learner can attain error better than $\Omega(\lambda/\eta)$ and rejection rate at most η even if given an unbounded number of samples and runtime. (This follows from a straightforward modification of Lemma D.1 in Goldwasser et al. (2020)).

Theorem 31 (PQ learning via TDS learning: The Agnostic Case) *Let \mathcal{A} be an agnostic TDS learner for some class \mathcal{C} that achieves error $O(\lambda) + \varepsilon$, runs in time $T_{\mathcal{A}}(\varepsilon)$ and has total sample complexity $m_{\mathcal{A}}(\varepsilon)$. Then, there is a PQ learner for \mathcal{C} that receives the parameters $(\varepsilon, \eta, \delta)$ as inputs and achieves error $O(\frac{\lambda}{\eta} + \varepsilon)$, rejection rate $O(\eta + \varepsilon)$ and probability of failure δ . In particular, the algorithm runs in time $T_{\mathcal{A}} \cdot \tilde{O}_{\varepsilon, \delta}(m_{\mathcal{A}}^6) + \text{poly}(m_{\mathcal{A}}, \frac{1}{\varepsilon}, \frac{1}{\eta}, \log \frac{1}{\delta})$ and uses $\text{poly}(m_{\mathcal{A}}, 1/\varepsilon, 1/\eta, \log 1/\delta)$ samples from the training and test distributions.*

Proof The algorithm for the agnostic case is very similar to [Algorithm 1](#), with one modification. In particular, after computing the estimates $\hat{p}_{i,t}^b$ up to error $\gamma_1 = \frac{\varepsilon\eta}{4T(T+1)}$ and ensuring that they are both at least equal to p_{\min} , the algorithm checks whether $\hat{p}_{i,t}^1 \leq \eta \cdot \hat{p}_{i,t}^0$, in which case the node (i, t) becomes a leaf node with label $(i, t) = 0$.

Recall that in the realizable case we had three types of leaf nodes of the branching program P constructed by [Algorithm 1](#) (see [Appendix D](#)). The modified algorithm described above creates one additional type of leaf nodes $\mathcal{T}_4 \subseteq \text{leaves}(P)$ for which we have ensured that $\hat{p}_N^1 \leq \eta \cdot \hat{p}_N^0$. For the following, suppose that we are in the event that holds with probability at least $1 - O(\delta)$ where the estimates $\hat{p}_{i,t}^b$ are γ_1 -close to their expected values $p_{i,t}^b$.

Observe that the distribution \mathcal{D}_1 is a mixture distribution with components $\{\mathcal{G}_N^1\}_{N \in \text{leaves}(P)}$. Each component \mathcal{G}_N^1 corresponds to weight p_N^1 . Similarly, \mathcal{D}_0 is a mixture of $\{\mathcal{G}_N^0\}_{N \in \text{leaves}(P)}$ with weights $\{p_N^0\}_{N \in \text{leaves}(P)}$.

Rejection Rate For the rejection rate, the only difference between the agnostic case and the proof of [Theorem 6](#) is the existence of nodes N of the new type \mathcal{T}_4 , where we have $\hat{p}_N^1 \leq \eta \cdot \hat{p}_N^0$. The additional term of the rejection rate that corresponds to these nodes is bounded by the following quantity:

$$\sum_{N \in \mathcal{T}_4} p_N^1 \leq O(\gamma_1 T^2) + \sum_{N \in \mathcal{T}_4} \hat{p}_N^1 \leq O(\gamma_1 T^2) + \eta \cdot \sum_{N \in \mathcal{T}_4} \hat{p}_N^0 \leq O(\gamma_1 T^2) + \eta \cdot \sum_{N \in \text{leaves}(P)} p_N^0 \leq O(\eta)$$

Therefore, the overall rejection rate is $O(\eta + \varepsilon)$.

Error Rate For the error rate on the test distribution, we focus on the nodes of type \mathcal{T}_3 (see [Appendix D](#)), i.e., the nodes N where the TDS learner has produced hypotheses h_N for which:

$$\Pr_{(\mathbf{x}, \mathbf{y}) \sim \bar{\mathcal{G}}_{N,P}^0} [h_N(\mathbf{x}) \neq \mathbf{y}] \leq A \cdot \underbrace{\min_{f \in \mathcal{C}} \left(\Pr_{(\mathbf{x}, \mathbf{y}) \sim \bar{\mathcal{G}}_N^1} [f(\mathbf{x}) \neq \mathbf{y}] + \Pr_{(\mathbf{x}, \mathbf{y}) \sim \bar{\mathcal{G}}_N^0} [f(\mathbf{x}) \neq \mathbf{y}] \right)}_{\lambda_N} + \varepsilon$$

The total error incurred by these nodes is bounded as follows by using the fact that $\mathcal{T}_3 \cap \mathcal{T}_4 = \emptyset$ and therefore for any $N \in \mathcal{T}_3$ we have $\hat{p}_N^0 < \frac{\hat{p}_N^1}{\eta}$.

$$\begin{aligned} \sum_{N \in \mathcal{T}_3} p_N^0 \cdot \Pr_{(\mathbf{x}, \mathbf{y}) \sim \bar{\mathcal{G}}_{N,P}^0} [h_N(\mathbf{x}) \neq \mathbf{y}] &\leq O(\varepsilon) + \sum_{N \in \mathcal{T}_3} \hat{p}_N^0 \cdot \Pr_{(\mathbf{x}, \mathbf{y}) \sim \bar{\mathcal{G}}_{N,P}^0} [h_N(\mathbf{x}) \neq \mathbf{y}] \\ &\leq A \cdot \min_{f \in \mathcal{C}} \sum_{N \in \mathcal{T}_3} \left(\hat{p}_N^0 \cdot \Pr_{(\mathbf{x}, \mathbf{y}) \sim \bar{\mathcal{G}}_N^1} [f(\mathbf{x}) \neq \mathbf{y}] + \frac{\hat{p}_N^1}{\eta} \cdot \Pr_{(\mathbf{x}, \mathbf{y}) \sim \bar{\mathcal{G}}_N^0} [f(\mathbf{x}) \neq \mathbf{y}] \right) + O(\varepsilon) \\ &\leq \frac{A}{\eta} \cdot \min_{f \in \mathcal{C}} \sum_{N \in \mathcal{T}_3} \left(p_N^0 \cdot \Pr_{(\mathbf{x}, \mathbf{y}) \sim \bar{\mathcal{G}}_N^1} [f(\mathbf{x}) \neq \mathbf{y}] + p_N^1 \cdot \Pr_{(\mathbf{x}, \mathbf{y}) \sim \bar{\mathcal{G}}_N^0} [f(\mathbf{x}) \neq \mathbf{y}] \right) + O(\varepsilon) \\ &\leq \frac{A}{\eta} \cdot \min_{f \in \mathcal{C}} \sum_{N \in \text{leaves}(P)} \left(p_N^0 \cdot \Pr_{(\mathbf{x}, \mathbf{y}) \sim \bar{\mathcal{G}}_N^1} [f(\mathbf{x}) \neq \mathbf{y}] + p_N^1 \cdot \Pr_{(\mathbf{x}, \mathbf{y}) \sim \bar{\mathcal{G}}_N^0} [f(\mathbf{x}) \neq \mathbf{y}] \right) + O(\varepsilon) \\ &\leq \frac{A \cdot \lambda}{\eta} + O(\varepsilon). \end{aligned}$$

The error incurred by the leaf nodes that lie outside \mathcal{T}_3 remains unchanged, which concludes the proof of [Theorem 31](#). \blacksquare

Remark 32 [Goldwasser et al. \(2020\)](#) showed that access to an agnostic ERM oracle in the standard (i.e., no distribution shift) setting suffices to construct an agnostic PQ learner using only a polynomial number of oracle calls. However, our result is not implied by theirs for the following reasons:

1. Upon acceptance, our TDS learner is allowed to output a constant-factor approximate hypothesis, whereas the reduction of [Goldwasser et al. \(2020\)](#) requires a hypothesis achieving optimal error.
2. Their approach further requires the learning algorithm to be proper (i.e., an ERM), whereas our reduction does not impose such a requirement.