2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)

Point Location and Active Learning: Learning Halfspaces Almost Optimally

Max Hopkins
University of California, San Diego
Email: nmhopkin@eng.ucsd.edu

Daniel Kane
University of California, San Diego
Email: dakane@eng.ucsd.edu

Shachar Lovett
University of California, San Diego
Email: slovett@cs.ucsd.edu

Gaurav Mahajan
University of California, San Diego
Email: gmahajan@eng.ucsd.edu

***Abstract*—Given a finite set $X \subset \mathbb{R}^d$ and a binary linear classifier $c : \mathbb{R}^d \to \{0,1\}$, how many queries of the form $c(x)$ are required to learn the label of every point in X? Known as *point location*, this problem has inspired over 35 years of research in the pursuit of an optimal algorithm. Building on the prior work of Kane, Lovett, and Moran (ICALP 2018), we provide the first nearly optimal solution, a randomized linear decision tree of depth $\tilde{O}(d\log(|X|))$, improving on the previous best of $\tilde{O}(d^2\log(|X|))$ from Ezra and Sharir (Discrete and Computational Geometry, 2019). As a corollary, we also provide the first nearly optimal algorithm for actively learning halfspaces in the membership query model. En route to these results, building on the work of Carlen, Lieb, and Loss (J. Geometric Analysis 2004), as well as Dvir, Saraf, and Wigderson (STOC 2014), we prove a novel characterization of Barthe's Theorem (Inventiones Mathematicae, 1998) of independent interest. In particular, we show that X may be transformed into approximate isotropic position if and only if there exists no k-dimensional subspace with more than a k/d-fraction of X, and provide a similar characterization for exact isotropic position.**

The below is an extended abstract. The full work can be found at https://arxiv.org/abs/2004.11380.

***Keywords*-point location; learning theory; active learning; halfspaces; vector scaling**

I. Introduction

Consider the following combinatorial question: an arbitrary finite set of points $X \subset \mathbb{R}^d$ is divided into two parts by an arbitrary hyperplane $h \in \mathbb{R}^d$. If for any $x \in \mathbb{R}^d$ you are allowed to ask its label with respect to h, i.e.

$$\text{sign}(\langle x, h\rangle) \in \{-, 0, +\},$$

what is the minimum number of questions needed to determine the label of every point in X? It is easy to observe that using such ternary questions (known as linear queries), deciding between the worst-case $\Omega(|X|^d)$ possible labelings requires at least $\Omega(d\log(|X|))$ queries. A matching upper bound, on the other hand, seems to be not so simple. Indeed the search for such an algorithm has been ongoing for over 35 years [1–5], and for good reason: slight variants and special cases of this simple question underlie many fundamental problems in computer science and mathematics.

This problem has been studied in the literature from two distinct, but equivalent, viewpoints. In machine learning, the question is a variant of the well studied problem of learning linear classifiers. From this standpoint, X is viewed as an arbitrary set of data points, and h as a hyperplane chosen by some adversary. The hyperplane defines a halfspace, which in turn acts as a binary (linear) classifier for points in X. The goal of the learner is to label every point in X with respect to this classifier in as few linear queries as possible. In the machine learning literature, this type of learning model is referred to as active learning with membership query synthesis [6], a semi-supervised branch of Valiant's [7] PAC-learning model.

On the other hand, in computational geometry and computer graphics, the problem is often viewed from its dual standpoint called *point location*. In this view, X is thought of as a set of hyperplanes in $\mathbb{R}^d$ rather than a set of points, and partitions the space $\mathbb{R}^d$ into cells. Given a point $h \in \mathbb{R}^d$, point location asks the mathematically equivalent question: how quickly can we determine in which cell h lies? Work in this area has often centered around its relation to important combinatorial problems like k-SUM or KNAPSACK [1, 8, 5], which reduce to special cases of point location. Since these views are equivalent, for the remainder of this paper we will adopt the standpoint common to machine learning, but will generally still refer to the problem as point location.

Regardless of which view is taken, up until this point the best algorithm for point location on n points in $\mathbb{R}^d$ took $\tilde{O}(d^2\log(n))$[1] queries, leaving a quadratic gap

MH supported by NSF Award DGE-1650112, DK by NSF Award CCF-1553288 (CAREER) and a Sloan Research Fellowship, and SL by NSF Award CCF-1909634.

[1]We use $\tilde{O}$ to hide poly-logarithmic factors in the dimension d.

2575-8454/20/$31.00 ©2020 IEEE
DOI 10.1109/FOCS46700.2020.00100

between the upper and lower bounds. In this work we close this gap up to a sub-polynomial factor, providing significant progress towards resolving the decades old question regarding the tightness of the $\Omega(d\log(n))$ information theoretic lower bound. In particular, we prove the existence of a nearly optimal randomized linear decision tree (LDT) for the point location problem in two models: bounded-error (δ-reliable), and zero-error (reliable). For the former, we build a randomized LDT of depth $O(d\log^2(d)\log(n/\delta))$, and require an additional additive $d^{1+o(1)}$ factor for the more difficult reliable model. Our bounded-error linear decision tree can be combined with standard results on PAC-learning halfspaces [9, 7, 10, 11] to provide the first nearly optimal learner for this class over arbitrary distributions.

A. Computational and Learning Models

In this work we study point location from the viewpoint of randomized linear decision trees (LDTs), a model common to both computational geometry and machine learning. Randomization allows us to study two different solution regimes: zero-error, and bounded-error.

1) Linear Decision Trees: A linear decision tree T is a tree whose internal nodes represent ternary linear queries, and whose leaves represent solutions to the problem at hand. Given a hyperplane $h \in \mathbb{R}^d$, a linear query is one of the form:

$$Q_x(h) = \operatorname{sign}(\langle x, h\rangle) \in \{-, 0, +\},$$

where $x \in \mathbb{R}^d$. The outgoing edges from an internal node indicate the branch to follow based on the sign of the linear query. The leaves, in our case, stand for labelings $T(h) : X \to \{-, 0, +\}$ of the instance space X. To execute the linear decision tree on input h, we begin at the root node. At each node v, we test the sign of a linear query with respect to some $x \in \mathbb{R}^d$ and proceed along the corresponding outgoing edge. We do this until we reach a leaf, at which point, we output its corresponding labeling. The complexity of a linear decision tree is measured by its depth, denoted $d(T)$, the length of the longest path from the root node to a leaf. Since it will be convenient in the following definitions, we will also define the input specific depth $d(T, h)$ to be the number of internal nodes reached on input h.

2) Randomized LDTs: Zero-Error: A randomized linear decision tree $T = (\mathcal{D}, \mathcal{T})$ is a distribution $\mathcal{D}$ over a (possibly infinite) set $\mathcal{T}$ of deterministic LDTs. In computational geometry, many works [12, 4, 5] on the point location problem focus mostly on randomized LDTs which label all points in X without making any errors. We will say that a randomized LDT satisfying this property *reliably* computes the point location problem. Formally,

Definition I.1 (Reliable computation)**.** *A randomized decision tree $T = (\mathcal{D}, \mathcal{T})$ reliably solves the point location problem on a set $X \subset \mathbb{R}^d$ if for any hyperplane $h \in \mathbb{R}^d$, the following holds:*

$$\forall T' \in \mathcal{T}, \forall x \in X : T'(h)(x) = \operatorname{sign}(\langle x, h\rangle)$$

In other words, every deterministic LDT that makes up our randomized LDT must correctly solve the point location problem. The main difference between this randomized model and deterministic LDTs then lies in how we measure the complexity of the object. For a deterministic LDT, the standard measure of complexity is its depth, the number of queries which must be made in order to learn the sign of every point on the worst-case input. In a randomized LDT on the other hand, for a given h there may exist a few trees in $\mathcal{T}$ with large input specific depth–so long as these trees are not chosen too often. More precisely, our measure of complexity will be the worst-case expected input specific depth of a tree drawn from $\mathcal{D}$.

Definition I.2. *The expected depth of a randomized decision tree $T = (\mathcal{D}, \mathcal{T})$, denoted ED$(T)$ is:*

$$ED(T) = \max_{h \in \mathbb{R}^d} \mathop{\mathbb{E}}_{T' \sim \mathcal{D}}[d(T', h)].$$

Equivalently, we can think of this parameter as the expected number of queries to learn the worst case input.

3) Randomized LDTs: Bounded-Error: While reliably solving point location is interesting from a computational geometry perspective and useful for building zero-error algorithms for k-SUM and related problems [8, 5], for many applications such a strict requirement is not necessary. In the PAC-learning model, for instance, we are only interested in approximately learning the hidden hyperplane with respect to some adversarially chosen distribution over $\mathbb{R}^d$. Here we do not need every LDT in our distribution to return the right answer - it is sufficient if most of them do. We will say an LDT δ-reliably computes (or solves) the point location problem on X if with probability at least $1-\delta$ it makes no errors. Formally,

Definition I.3 (δ-reliable computation)**.** *A randomized decision tree $T = (\mathcal{D}, \mathcal{T})$ δ-reliably solves the point location problem on a set $X \subset \mathbb{R}^d$ if for any hyperplane $h \in \mathbb{R}^d$, the following holds:*

$$\Pr_{T' \sim \mathcal{D}}[\exists x \in X : T'(h)(x) \neq \operatorname{sign}(\langle x, h\rangle)] < \delta$$

Since we are now allowing a small probability of error, as is commonly the case in randomized algorithms we will change our complexity measure to be worst-case rather than average. In particular, our complexity measure for δ-reliably computing the point location problem will be the maximum depth of any $T' \in \mathcal{T}$.

Definition I.4. *The maximum depth of a randomized decision tree $T = (\mathcal{D}, \mathcal{T})$, denoted MD$(T)$ is:*

$$MD(T) = \max_{T' \in \mathcal{T}} \left(d(T')\right).$$

Using this worst-case complexity measure will help us transfer our results to an active variant of PAC-learning theory which uses a similar worst-case measure called query complexity.

4) PAC and Active Learning: Probably Approximately Correct (PAC) learning is a learning framework due to Valiant [7] and Vapnik and Chervonenkis [9]. In the PAC-learning model, given a set (instance space) X, and a family of classifiers $\mathcal{H}$ (where $h \in \mathcal{H}$ maps elements in X to a binary label in $\{0,1\}$), first an adversary chooses a distribution D over X and a classifier $h \in \mathcal{H}$. Then, the learner receives labeled samples from the distribution with the goal of approximately learning h (up to ε accuracy) with high probability $(1-\delta)$ in the fewest number of samples. The complexity of learning a concept class $(X, \mathcal{H})$ is given by its *sample complexity*, the minimum number of samples $n(\varepsilon, \delta)$ such that there exists a learner A which maps samples from D to labelings which achieve this goal:

$$\forall D, h : \Pr_{S \sim D^{n(\varepsilon,\delta)}} \left[\Pr_{x \sim D} [A(S)(x) \neq h(x)] \leq \varepsilon \right] \geq 1 - \delta. \tag{1}$$

Active learning, on the other hand, more closely mirrors today's challenges with big data where samples are cheap, but labels are expensive. Using the same adversarial setup, active learning[2] [6] provides the learner with unlabeled samples from D along with an oracle which the learner can query to learn the label $h(x)$ for any $x \in X$. The goal of active learning is to adaptively query the most informative examples, and thus exponentially decrease the required number of labeled points over Valiant's initial "passive" model. In this case, the complexity measure of learning $(X, \mathcal{H})$ is given by its *query complexity* $q(\varepsilon, \delta)$, the minimum number of queries (oracle calls) required to achieve the same guarantee from Equation (1).

5) Non-homogeneous Hyperplanes and Binary Queries: In the previous sections, we assumed that both the labeling of X and the queries making up our LDT are ternary and given by a homogeneous (through the origin) hyperplane. While these assumptions are standard in the point location literature, in learning theory it is more common to consider binary queries, and, when possible, the more general class of non-homogeneous hyperplanes. In this generalized scenario, the labels of X and linear queries of our decision tree instead take the form:

$$Q_x(h, b) = \operatorname{sign}(\langle x, h \rangle + b) \in \{-, +\}.$$

We assume throughout this extended abstract that we are in the ternary, homogeneous case, but show in Section 7 of the full paper how to extend our results to the binary, non-homogeneous case.

[2]The model of active learning we use is referred to as the Pool + Membership Query Synthesis (MQS) model

B. Results

We prove the existence of nearly optimal LDTs (learners) for all three of these regimes. For the bounded-error regime, our result is only a poly $\log(d)$ factor away from being optimal.

Theorem I.5. *Let $X \subset \mathbb{R}^d$, $|X| = n$, be an arbitrary finite set. Then there exists a randomized LDT T that δ-reliably computes the point location problem on X with maximum depth:*

$$MD(T) \leq O\left(d \log^2(d) \log\left(\frac{n}{\delta}\right)\right)$$

Combining this with standard PAC-learning results for classes with finite VC-dimension [10], we prove the existence of the first nearly optimal active learner for halfspaces (either homogeneous or non-homogeneous), denoted $\mathcal{H}_d$, over arbitrary distributions.

Corollary I.6. *The query complexity of actively learning $(\mathbb{R}^d, \mathcal{H}_d)$ over any distribution, with membership queries is:*

$$q(\varepsilon, \delta) \leq O\left(d \log^2(d) \log\left(\frac{d}{\varepsilon\delta}\right)\right)$$

By adding a verification step to our LDT that δ-reliably computes point location, we prove the existence of an LDT that reliably computes point location at the cost of an additive $d^{1+o(1)}$ factor.

Theorem I.7. *Let $X \subset \mathbb{R}^d$, $|X| = n$, be an arbitrary finite set. Then there exists a randomized LDT T that reliably computes the point location problem on X with expected depth:*

$$ED(T) \leq O\left(d \log^2(d) \log(n)\right) + d \cdot 2^{O\left(\sqrt{\log(d) \log\log(d)}\right)}.$$

For large enough n, this bound is optimal up to a polylogarithmic factor in d.

C. Related Work

1) Point Location: The early roots of the point location problem as we study it[3] stem from the study of other classic combinatorial problems. As such, Meyer auf Der Heide's [1] early work giving an $O(n^4 \log(n))$ linear search algorithm for the n-dimensional Knapsack problem is often considered the seminal work for this area. Meiser [2] later stated the problem in the form we consider, and provided an $O(d^5 \log(n))$ depth linear decision tree for a general set of n hyperplanes in $\mathbb{R}^d$. These results were later improved by Cardinal, Iacono, and Ooms [3], Kane, Lovett, and Moran [4, 8] and finally by Ezra and Sharir [5] to an expected depth of $\tilde{O}(d^2 \log(n))$.

Our work expands on the margin-based technique introduced by Kane, Lovett, and Moran in [4], which by itself gives an $\tilde{O}(d^3 \log(n))$ expected depth LDT. Kane, Lovett, and Moran use the fact that point location is invariant to invertible linear transformations of the data to transform X into isotropic position. It is then possible to take advantage of the structure introduced by this transformation to apply a margin-based technique from their earlier work with Zhang [12]. We employ a similar overall strategy, repeatedly transforming the remaining unlabeled points into isotropic position, but by novel structural analysis and a new inference technique based upon dimensionality reduction, we improve to a nearly tight $\tilde{O}(d \log(n)) + d^{1+o(1)}$ expected depth randomized LDT.

2) Learning Halfspaces: Learning halfspaces is one of the oldest and most studied problems in learning theory. We cover here only a small fraction of works, those which are either seminal or closely related to our results. The earliest such works are due to Vapnik and Chervonenkis in the late 1960's and early 70's, culminating in the introduction of VC theory [9]. Using these tools, Blumer, Ehrenfeucht, Haussler, and Warmuth [14], showed that d-dimensional halfspaces can be PAC-learned in only $O((d \log(1/\varepsilon) + \log(1/\delta))/\varepsilon)$ labelled examples, and that this bound is nearly tight (much later, Hanneke [11] tightened the result). In the years since, many works showed that active learning could in some cases exponentially decrease the number of labelled examples required to learn. Many of these works focused on learning halfspaces in the Pool-based model (where in contrast to MQS, only sampled points may be queried) with restricted distributions. A series of papers [15–17] showed increasingly improved bounds for learning homogeneous halfspaces over (nearly) uniform distributions on the unit ball. Later this work was extended to more general classes of distributions [18, 19], finally giving a nearly optimal $\tilde{O}(d \log(1/\varepsilon))$ algorithm. These results were then extended to the non-homogeneous case in a more powerful query model that allowed the learner to compare points [12, 20].

Work that focused on learning over adversarial distributions, on the other hand, tended to use the stronger MQS learning model. The most efficient theoretical algorithms in this regime are (implicit) results from the point location literature, most recently Ezra and Sharir's [5] result translates to a roughly $\tilde{O}(d^2 \log(1/\varepsilon))$ query algorithm under adversarial distributions. On the practical end, a number of works [21, 22] presented MQS algorithms that seemed experimentally to achieve the $d \log(1/\varepsilon)$ lower bound, but none could do so provably. Our work is the first to provably match the practical performance of these heuristic methods.

II. Proof Overview

The remainder of this extended abstract sketches the proofs of Theorem I.5, Corollary I.6, and Theorem I.7. The details can be found in the full paper. Our proof sketch is split into four main sections. In Section II-A, we provide a high-level sketch of our approach as a whole. In Section II-B, we examine how to build an efficient bounded-error weak learner that is at the core of both Theorem I.5 and Theorem I.7. In Section II-C, we show how to boost this weak learner to obtain Theorem I.5 and Corollary I.6. In Section II-D, we show how to boost and verify the weak learner of Section II-B to obtain the zero-error LDT of Theorem I.7.

A. Overall Approach

To start, we briefly provide some intuition for our general approach. We will assume throughout the extended abstract that we are in the homogeneous case. We show in Section 7 of the full paper how our arguments generalize to the non-homogeneous case. Recall then the setup: we are given a finite set $X \subset \mathbb{R}^d$, there is an unknown homogeneous hyperplane (given by a normal vector) $h \in \mathbb{R}^d$, and our goal is to label X using linear queries of the form $\operatorname{sign}(\langle x, h\rangle)$, where we can use any $x \in \mathbb{R}^d$.

Assume we are given some point of reference $x_{\text{ref}} \in \mathbb{R}^d$ and an orthonormal basis $\{v_1, \dots, v_d\}$ for $\mathbb{R}^d$, and that it is possible to learn for each i up to high accuracy $\frac{\langle v_i, h\rangle}{\langle x_{\text{ref}}, h\rangle}$. Since each point $x \in X$ can be written in terms of this orthonormal basis, this would allow us to estimate $\frac{\langle x, h\rangle}{\langle x_{\text{ref}}, h\rangle}$ up to high accuracy, and thus learn the label of x. This strategy alone will work to label all of X efficiently,

[3]In its most general form, the point location problem asks about any type of partition, and reaches back to the mid 1970's or earlier [13].

unless it contains many points with small margin (inner product with h).

Kane, Lovett, and Moran [4], using a different inference strategy, ran into this same fundamental issue. They circumvent the problem (for sets in general position) via two key facts: point location is invariant to invertible linear transformations, and there exists such a transformation on X that ensures many points have large margin. Unfortunately, if the remaining points have very small margin, then this method loses polynomial factors in the dimension d. Concretely, the result of [4] required $\tilde{O}(d^3 \log n)$ queries.

We solve this problem, and achieve a near-linear dependence on the dimension d, by using these small margin points to apply dimensionality reduction. In essence, we argue that for every hyperplane h and "nice" set of points $X \subset \mathbb{R}^d$, there exists a parameter $1 \leq k \leq d$ for which the following holds. It is possible to split $\mathbb{R}^d$ into two orthogonal subspaces: V, a $(d - O(k))$-dimensional subspace that is nearly orthogonal to h, and $V^\perp$, an $O(k)$-dimensional subspace that nearly contains h. In addition, a k/d fraction of the points in X have large enough margin such that projecting onto $V^\perp$ preserves their label. As a result, we significantly reduce the dimension of the problem (from d to k), which allows our learner to label large margin points in $\tilde{O}(k)$ rather than $\tilde{O}(d)$ queries. In essence, we couple the fraction of points that we label to the number of queries we make.

Formally, the "nice" structure we need is an approximate isotropic position. Building on previous works on "vector scaling" problems [23–26], we give an exact characterization of when a point set X can be transformed to such a configuration (See Lemma 4.19 of the full paper for details). This allows us to extend our analysis to an arbitrary finite set of points. In summary, this procedure allows us to infer a k/d fraction of an arbitrary point set X using only $\tilde{O}(k)$ queries. Theorem I.5 and Theorem I.7 both follow from applying different boosting procedures to this process. The former uses a weighting scheme between iterations to ensure every point is learned with high probability, while the latter runs a slightly more costly verification process on each learner to ensure that no mistakes have been made.

B. Weak Learner

Both of our main results, then, are based upon boosting a highly efficient weak learner: a randomized algorithm A that makes linear queries and returns (abusing notation) a partial labeling $A : X \to \{-, 0, +, \perp\}$ in which $\perp$ is interpreted as "don't know." Before introducing our core weak learner, we need to introduce some further notation and terminology.

Definition II.1. *Let $X \subset \mathbb{R}^d$ be a finite set of unit vectors, and μ a distribution over X. We denote such a pair by (X, μ). In cases where the ambient dimension is not clear from context, we denote it by $(X \subset \mathbb{R}^d, \mu)$.*

Note that assuming X consists of unit vectors does not lose any generality for point location or active learning homogeneous hyperplanes, since normalizing points does not change their label. For notational simplicity, we often refer to a fraction of X with respect to μ simply as a fraction of (X, μ). For instance, it is useful when dealing with weak learners to be able to talk about the fraction of (X, μ) they label. We call this value their *coverage*.

Definition II.2 (Coverage)**.** *Given a pair (X, μ) and A a weak learner, A's coverage with respect to (X, μ), $C_\mu(A)$, is a random variable (over the internal randomness of A) denoting the measure of X learned:*

$$C_\mu(A) = \Pr_{x \sim \mu}[A(x) \neq \perp].$$

It will also be useful to have a way to talk about the correctness of a weak learner. For this, we adopt notation introduced in [27]:

Definition II.3 (Reliability)**.** *Given a pair (X, μ), we say that a learning algorithm A is p-reliable if with probability $1 - p$ over the internal randomness of A, the labeling output by A makes no errors. If A never makes an error, we call it reliable.*

These definitions are all we need to present our core weak learner, PartialLearn:

Theorem II.4 (Informal Theorem 4.4, full paper: PartialLearn)**.** *Given a pair $(X \subset \mathbb{R}^d, \mu)$ and $p > 0$, there exists a p-reliable weak learner* PartialLearn *with the following guarantees. For any (unknown) hyperplane $h \in \mathbb{R}^d$, with probability $1 - p$ there exists $\frac{1}{10} \leq k \leq d$ such that:*

1) PartialLearn *has coverage at least k/d.*
2) PartialLearn *makes at most $O(k \log(d) \log(d/p))$ queries.*

1) Isotropic Position: We will start by proving an intermediary result, IsoLearn: a weak learner for a pairs (X, μ) which satisfy a structural property called *ε-isotropic position.*

Definition II.5 (ε-isotropic Position)**.** *A pair $(X \subset \mathbb{R}^d, \mu)$ lies in ε-isotropic position if:*

$$\forall v \in \mathbb{R}^d : (1 - \varepsilon)\frac{1}{d} \leq \sum_{x \in X} \mu(x) \frac{\langle x, v \rangle^2}{\|v\|^2} \leq (1 + \varepsilon)\frac{1}{d}$$

The key to our learner's efficiency lies in its ability to exploit this structure to find a parameter k such that it can infer the labels of a $\frac{k}{d}$ fraction of (X, μ), while using only $\tilde{O}(k)$ queries.

Lemma II.6 (Informal Lemma 4.6, full paper: IsoLearn)**.** *Let the pair $(X \subset \mathbb{R}^d, \mu)$ be in $1/4$-isotropic position, and let $p > 0$. There exists a p-reliable weak learner* IsoLearn *with the following guarantees. For any (unknown) hyperplane $h \in \mathbb{R}^d$, with probability $1 - p$ there exists $\frac{1}{10} \leq k \leq d$ such that:*

1) IsoLearn *has coverage at least k/d.*
2) IsoLearn *makes at most $O(k \log(d) \log(d/p))$ queries.*

Throughout the rest of this section, we assume the (unknown) hyperplane is non-zero (this situation may occur in our algorithm if points in X lie on the hyperplane h). We may assume this without loss of generality as it is easy to test up front by checking if the sign of a few random points is zero. We will break down the construction of IsoLearn into three parts: finding the structure in X, dimensionality reduction, and inference.

2) Finding the Structure in X: The structure at the core of IsoLearn is one common to the machine learning literature, the concept of *margin*.

Definition II.7 (Margin)**.** *Given a hyperplane $h \in \mathbb{R}^d$, the margin of a point $x \in \mathbb{R}^d$ is its inner product with h, $\langle x, h \rangle$. Since h is not restricted to be unit length, we will often work with the* ***normalized margin*** *of x, which we denote by $m(x, h)$:*

$$m(x, h) = \frac{\langle x, h \rangle}{\|h\|}.$$

Margin-based algorithms are common in both the active learning and point location literature (e.g. [18, 12, 4]). In their recent work on point location, Kane, Lovett, and Moran [4] observed that any set of unit vectors in ε-isotropic position must have an $\Omega(1/d)$ fraction of points with normalized margin $\Omega(1/\sqrt{d})$. This follows from the fact that the sum of the squared normalized margins of points in X is at least $(1-\varepsilon)\frac{n}{d}$—if not enough points have large margin, their squared sum cannot be this large. Through the cleaner inference technique presented in this work, this observation alone is enough to build a randomized LDT with expected depth $\tilde{O}(d^2 \log(n))$. Reaching near-linear depth, however, requires insight into the finer-grain structure of X. The idea is to show X has a "gap" in margin, that is parameters t and s such that not too many points have normalized margin between t and t/s. IsoLearn will work by exploiting this gap, ignoring in a sense points with margin less than t/s in order to learn the fraction with margin greater than t. We formalize this key structure in the following definition.

Definition II.8 ((k, t, s, c)-structured)**.** *We call a pair $(X \subset \mathbb{R}^d, \mu)$ (k, t, s, c)-structured with respect to a hyperplane $h \in \mathbb{R}^d \setminus \{0\}$ if it satisfies the following properties:*

1) *Many points in X have normalized margin at least t:*
$$\Pr_{x \sim \mu}[|m(x, h)| \geq t] \geq \frac{k}{d} \tag{2}$$
2) *Many points in X have normalized margin at most t/s:*
$$\Pr_{x \sim \mu}[|m(x, h)| \leq t/s] \geq 1 - \frac{ck}{d} \tag{3}$$

When clear from context, we often drop the phrase "with respect to h". Similarly, throughout the paper we will use the shorthand (k, t, s)-structured for $(k, t, s, 5)$-structured.

IsoLearn relies on the following structural insight concerning pairs (X, μ) in $1/4$-isotropic position:

Lemma II.9. *Let the pair $(X \subset \mathbb{R}^d, \mu)$ be in $1/4$-isotropic position. Then for any hyperplane $h \in \mathbb{R}^d \setminus \{0\}$ and $s > 2$, there exist parameters k and t satisfying:*

1) *$1 \leq k \leq d$.*
2) *$t \geq s^{-O(\log(d))}$.*
3) *(X, μ) is (k, t, s)-structured.*

In the full paper, we prove an algorithmic variant of this result (Lemma 4.10). In particular, we show that with high probability it is possible to find parameters k and t such that X is (k, t, s)-structured[4].

3) Dimensionality Reduction: For the moment, assume we have found some parameters $k \leq d$, $t \geq d^{-O(\log(d))}$, $s = d^{\Omega(1)}$, such that X is (k, t, s)-structured. When $k = \Omega(d)$, Equation (2) implies that a constant fraction of X has normalized margin at least t. Since we can afford to use $O(d)$ queries in this case, no dimensionality reduction is required. Thus, we focus on the case where $k \ll d$, where our goal will be to reduce the dimension of the problem from d to k.

Our basic strategy will be to decompose $\mathbb{R}^d$ into two orthogonal subspaces: V, a high dimensional subspace that is nearly orthogonal to h (we will refer to this as the "nearly orthogonal subspace" hereafter), and $V^{\perp}$, a

[4]As we will note later in greater detail, t here is found implicitly in terms of the margin of some reference point $x_{\text{ref}} \in \mathbb{R}^d$.

low dimensional subspace that nearly contains h. The structural result we need is the following.

Lemma II.10. *Let $(X \subset \mathbb{R}^d, \mu)$ be (k,t,s)-structured with respect to a hyperplane $h \in \mathbb{R}^d \setminus \{0\}$ where $k < d/10$. Then there exists a subspace V with the following properties:*

1) V is high dimensional:

$$Dim(V) = d - O(k)$$

2) V has small margin with respect to t:

$$\forall v \in V, |m(v,h)| \leq O\left(\|v\| d \frac{t}{s}\right)$$

Note that the existence of such a subspace (which is trivial from taking the orthogonal complement of h) is not sufficient for our purposes—we must be able to find it efficiently as well. In the full paper, we prove an algorithmic version of this result (Lemma 4.15) that finds V efficiently by taking the approximate span of a small sample from X whose points have margin less than t/s. For simplicity, we denote below $e = \dim(V^\perp)$ where $e = O(k)$. Further, we set $s = d^{\Omega(1)}$ to guarantee a large enough gap between "large margin" and "small margin".

4) Inference: For any subspace V, we can write any $x \in X$ in terms of an orthonormal basis v_i for V, and w_i for $V^\perp$:

$$x = \sum_{i=1}^{d-e} \alpha_i v_i + \sum_{i=1}^{e} \beta_i w_i$$

for some set of constants $-1 \leq \alpha_i, \beta_i \leq 1$ (recall that x is a unit vector). Since the inner product is bi-linear, this means we can express the margin of x through the margins of v_i and w_i:

$$\langle x, h\rangle = \sum_{i=1}^{d-e} \alpha_i \langle v_i, h\rangle + \sum_{i=1}^{e} \beta_i \langle w_i, h\rangle. \tag{4}$$

The idea behind finding a high-dimensional subspace V with the properties given in Lemma II.10 is that the lefthand term, x's projection onto V, does not have much effect on x's margin. This is because the basis vectors v_i are unit length, and thus satisfy a small margin condition guaranteeing that:

$$m(x,h) \in \left(\sum_{i=1}^{e} \beta_i m(w_i,h)\right) \pm \frac{t}{d^{\Omega(1)}}.$$

Inferring the sign of x then reduces to learning information about the smaller dimensional space $V^\perp$. In particular, imagine that we could learn up to an additive error of $\frac{t}{d^{\Omega(1)}}$ the normalized margin of each w_i. Calling this value γ_i, we would be able to express the normalized margin of h as:

$$m(x,h) \in \left(\sum_{i=1}^{e} \beta_i \gamma_i\right) \pm \frac{t}{d^{\Omega(1)}}.$$

Recall that (X,μ) is (k,t,s)-structured, meaning at least a k/d measure of points have normalized margin at least t. For such a point x, notice that most of its margin must come from the lefthand sum. Further, as long as this sum is at least $\frac{t}{d^{\Omega(1)}}$, we can infer the sign of x. This process would allow us to infer the sign of any point with margin at least t.

Unfortunately, it is not clear that it is possible to learn the margin of each w_i. To combat this, our algorithm for finding (k,t,s)-structure (Lemma 4.15, full paper) outputs a reference point x_{ref} whose normalized margin is t. Then by querying the sign of the point $w_i - \alpha x_{\text{ref}}$, we can check relations of the form:

$$m(w_i,h) \overset{?}{\geq} \alpha t.$$

Note that there exists some $|\alpha_w| \leq t^{-1}$ such that

$$m(w_i,h) = \alpha_w t,$$

since the normalized margin of w_i is at most 1. Finding α_w up to an additive error of $\frac{1}{d^{\Omega(1)}}$ is then sufficient for our inference. This can be done by binary searching over α in only $\log\left(\frac{t^{-1}}{d^{\Omega(1)}}\right) = O(\log^2(d))$ queries. Since there are only $e = O(k)$ of these vectors, we only need a total of $O(k\log^2(d))$ queries in total.

As a final note, we can use this same inference method for the case where $k \geq \frac{d}{10}$, but do not require the dimensionality reduction step – $V^\perp$ will just be all of $\mathbb{R}^d$.

In Section 3 of the full paper, we cover in more detail the mechanism behind the reference point x_{ref}, found via taking a random combination of points in our set. This technique turns out to be crucial both for identifying (k,t,s)-structure, and for efficiently finding the subspace V. These processes are covered in Section 4 of the full paper, where we show how both may be done with probability $1-p$ in no more than $O(k\log(d)\log(d/p))$ queries, which completes the proof.

5) Arbitrary Sets and Distributions: IsoLearn is only an intermediary result since most pairs (X,μ) are not in $1/4$-isotropic position. However, drawing on the results of Barthe [23], Forster [24], Carlen, Lieb, and Loss [25], and Dvir, Saraf, and Wigderson [26], we can prove a related structural result: any pair (X,μ) contains a subspace dense in X with respect to μ which may be transformed into $1/4$-isotropic position.

Proposition II.11. *Given a pair $(X \subset \mathbb{R}^d, \mu)$, for some $1 \le k \le d$, for all $\varepsilon > 0$ there exist:*

1) *A k-dimensional subspace V with the property $\mu(X \cap V) \ge \frac{k}{d}$ and*
2) *An invertible linear transformation $T : V \to V$ such that the pair $((X \cap V)_T, (\mu|_{X\cap V})_T)$:*

$$(X \cap V)_T = \left\{ x_T = \frac{T(x)}{\|T(x)\|} : x \in X \cap V \right\},$$

$$(\mu|_{X\cap V})_T = \frac{\mu(x)}{\mu(X \cap V)}$$

is in ε-isotropic position.

The proof of Theorem II.4 follows from Proposition II.11 via an observation of Kane, Lovett, and Moran [4]: point location is invariant to invertible linear transformations. In greater detail, let $h' = (T^{-1})^\top(h)$, and $x' = \frac{T(x)}{\|T(x)\|}$. Kane, Lovett, and Moran observe that $\langle x', h' \rangle = \langle \frac{x}{\|T(x)\|}, h \rangle$. Thus not only is it sufficient to learn the labels of x' with respect to h', but we can simulate linear queries on h' simply by normalizing x by an appropriate constant. Applying Proposition II.11, we can then apply Barthe's isotropic transformation [23] on some dense subspace, and apply IsoLearn to complete the proof. The details are covered in Section 4.2 of the full paper.

C. Bounded-Error: Boosting

Now we will show how to boost this weak learner into an LDT that δ-reliably solves the point location problem.

Theorem II.12. *Let $X \subset \mathbb{R}^d$, $|X| = n$. Then there exists a randomized LDT T that δ-reliably computes the point location problem on X with maximum depth:*

$$MD(T) \le O\left(d \log^2(d) \log\left(\frac{n}{\delta}\right)\right).$$

To make this boosting process simpler, we will start by applying the boosting process from [4] to build a stronger weak learner with constant coverage.

Lemma II.13 (Informal Lemma 5.2, full paper :WeakLearn)**.** *Let $X \subseteq \mathbb{R}^d$ be a set and μ a distribution over X. There exists a weak learner* WeakLearn *with the following guarantees. For any (unknown) hyperplane $h \in \mathbb{R}^d$, the following holds:*

1) WeakLearn *is* $.01$*-reliable*
2) *With probability at least* $.99$, WeakLearn *has coverage at least* $.99$.
3) WeakLearn *uses at most $O(d \log^2(d))$ queries*

The strategy for proving Lemma II.13 is simple. At each step, we restrict to the set of un-inferred points and run another instance of PartialLearn with probability parameter $p = 1/\text{poly}(d)$. Since each instance learns a k/d measure of points in $O(k \log(d) \log(d/p))$ queries for some k, repeating this process until we have used $O(d \log^2(d))$ queries is sufficient (see Section 5 of the full paper for details).

Unfortunately, continuing this strategy to learn all of X would force us to set our probability parameter p to be inverse polynomial in $\log(n)$, costing an additional $\log\log(n)$ factor in the depth of our LDT. Instead, we will employ the fact that WeakLearn can learn over any distribution to apply a boosting process that re-weights X at each step. The idea is that by multiplicatively reducing the weight on points learned by a certain iteration, we can ensure that, with high probability, every point is labeled in at least 5% of the learners. Since each learner is correct with 99% probability, a Chernoff bound tells us that the majority label for each point will then be correct with probability at least $1 - \delta$ as desired. We give the details of this process in Section 5 of the full paper.

D. Zero-error: Verification

PartialLearn (Theorem II.4) comes with a small probability of error. In this section, we explain our strategy for verifying the weak learner to build a randomized LDT that reliably computes point location.

Theorem II.14. *Let $X \subset \mathbb{R}^d$ be an arbitrary finite set. Then there exists a randomized LDT T that reliably computes the point location problem on X with expected depth:*

$$ED(T) \le O\left(d \log^2(d) \log(n)\right) + d \cdot 2^{O\left(\sqrt{\log(d) \log\log(d)}\right)}.$$

The core of this theorem lies in showing how verifying the labelings which stem from naively boosting PartialLearn (in the sense of Lemma II.13) reduces to a related combinatorial problem we term *matrix verification*:

Definition II.15 (Matrix Verification)**.** *Let $S \subset \mathbb{R}^d$ be a subset of size m, $h \in \mathbb{R}^d$ a hyperplane, and $\{C_{ij}\}_{i,j=1}^m$ a constraint matrix in $\mathbb{R}^{m\times m}$. We call the problem of determining whether for all i, j:*

$$\langle x_i, h \rangle \le C_{ij} \langle x_j, h \rangle$$

a ***matrix verification problem*** *of size m. Further, we denote by $V(m)$ the minimum expected number of queries made across randomized algorithms which solve verification problems of size m in any dimension.*

In fact, we show a two-way equivalence between point location and matrix verification: without too much overhead, point location reduces to a small matrix verification problem, and matrix verification reduces in turn to solving several point location problems in fewer dimensions. This recursive structure allows us to efficiently solve both problems.

In this section, we sketch both directions of this equivalence and show how to use them to build the zero-error LDT from Theorem I.7. To start, we examine how verifying point location reduces to matrix verification. To do so, we must first examine the source of errors in the weak learners we wish to verify.

1) The Source of Errors: We have not yet covered exactly where the error is introduced into PARTIALLEARN (Theorem II.4). The culprit is the "nearly orthogonal" subspace V, which with some small probability, may not actually satisfy the key assumption:

$$\forall v \in V : |\langle v, h\rangle| \leq \frac{\|v\|}{d^{\Omega(1)}} |\langle x_{\text{ref}}, h\rangle|. \tag{5}$$

In more detail, we build V via the covariance matrix of a sample S of $O(d\log(d))$ small margin points from X. Our weak learner checks probabilistically that points in S satisfy:

$$\forall s \in S : |\langle s, h\rangle| \leq \frac{1}{d^{\Omega(1)}} |\langle x_{\text{ref}}, h\rangle|, \tag{6}$$

which if true, verifies Equation (5) (See Lemma 4.14, full paper). The source of the error is then tied intrinsically to this set of equations for S. If we can verify that these equations hold, we can assure that we have not mislabeled any points.

2) Learning Relative Margins: We would like to show that verifying Equation (6) over boosted instances of PARTIALLEARN reduces to a small matrix verification problem. The key idea in this reduction lies in noticing that the sets of equations for each instance are related. This stems from the fact that our inferences in PARTIALLEARN come with extra information: their relative margin to x_{ref}.

Recall our inference strategy from the proof sketch of ISOLEARN. Writing a point $x \in \mathbb{R}^d$ in terms of orthonormal bases v_i for V and w_i for $V^\perp$, we use our knowledge of their relation to x_{ref} to bound x's normalized margin:

$$\frac{\langle x, h\rangle}{\|h\|} \in \frac{\langle x_{\text{ref}}, h\rangle}{\|h\|} \left(\sum_{i=1}^{e} \gamma_i \beta_i \pm \frac{1}{d^{\Omega(1)}} \right).$$

In Section II-B4, we used this to show that as long as the sum $\left|\sum_{i=1}^{e} \gamma_i \beta_i\right|$ is large enough, we can infer the sign of x. However, the equation actually provides additional information that was not useful until this point: $\sum_{i=1}^{e} \gamma_i \beta_i$ also acts as an approximation of the relative margin of x to x_{ref}.

3) From Point Location to Matrix Verification: It turns out that this approximation is all we need to reduce to matrix verification. Assume for a moment that $|X| = \text{poly}(d)$, and boost PARTIALLEARN naively until all of X is labeled. Since X is small, this only takes $O(d\log(d))$ rounds. For the ith round, let S_i denote the sample S of small points, and x_i denote the reference point x_{ref}. For each round, our goal is to verify the $O(d\log(d))$ equations from Equation (6).

To reduce this problem to matrix verification, we will start by verifying the final weak learner (which we can do directly). Assume now, inductively, that we have verified equations for all but the first i learners. Since all of X is labeled in this process and S_i is a sample from X, each $s_i \in S_i$ must be labeled by some future weak learner. By the previous observation, this further means that the relative margin of each $s_i \in S_i$ is approximately known to one of the reference points x_j for $j > i$. To verify Equation (6), it is then sufficient to check a set inequalities of the form:

$$\langle x_i, h\rangle \leq C_{ij} \langle x_j, h\rangle \tag{7}$$

for some constant C_{ij} (see Section 6 of full paper for details). These equations then form a matrix verification problem of size $m = O(d\log(d))$ on the reference points x_i. If we set the probability parameter p of our weak learner to be $1/\text{poly}(d)$, then we can also be assured that verification will succeed with at least 50% probability, allowing us to bound the expected number of queries for point location on any $X \subset \mathbb{R}^d$ of size $n = \text{poly}(d)$, denoted $T(n, d)$, by:

$$T(d^{O(1)}, d) \leq C_1 d \log^3(d) + 2V(C_2 d\log(d)), \tag{8}$$

for some constants C_1, C_2. The full details are covered in Lemma 6.4 of the full paper.

4) From Matrix Verification to Point Location: Solving a d-dimensional matrix verification problem of size m naively requires checking $O(m^2)$ inequalities, which would cause our upper bound for point location in Equation (8) to become quadratic in the dimension. To escape this, observe that for each row i, there is a single inequality that is necessary and sufficient to verify the rest: the minimum over j of $C_{ij}\langle x_j, h\rangle$. If we can efficiently compute the minimum index for each row, we can solve the problem in only $\tilde{O}(m)$ queries.

The key observation of this section is that finding this minimum can be rephrased as a point location problem. In particular, we can compute the minimum by knowing for all distinct i, j, k:

$$\text{sign}\left(C_{ij}\langle x_j, h\rangle - C_{ik}\langle x_k, h\rangle\right) = \text{sign}\left(\langle C_{ij}x_j - C_{ik}x_k, h\rangle\right),$$

which is a point location problem on poly(m) points in d dimensions. By a similar, divide and conquer argument, we can reduce finding the minimum to several point location problems in *fewer* than d dimensions. Consider dividing the constraint matrix C into batches of columns with indices $C_1 = \{1, \ldots, b\}, C_2 = \{b+1, \ldots, 2b\}$,... and finding the minimum index just within each subset. By directly comparing these minima, we can stitch together a global solution. Further, each sub-problem corresponds to a point location problem in b dimensions, since its point set lies in the span of at most b vectors (e.g. $\{x_1, \ldots, x_b\}$). This implies that we can bound the expected number of queries to solve matrix verification by:

$$V(m) \leq 2m + \frac{m^2}{b} + \frac{2m}{b} T(mb^2, b). \tag{9}$$

The full details are covered in Lemma 6.5 of the full paper.

5) Putting it all Together: Combining Equations (8) and (9) sets up a recurrence that implies the following bound on $T(n, d)$ for $n = \text{poly}(d)$:

$$T(d^{O(1)}, d) \leq d \cdot 2^{O\left(\sqrt{\log(d) \log\log(d)}\right)}. \tag{10}$$

However, since we are interested in arbitrarily large X, we must make one final adjustment. Instead of initially running the entire boosting process before verification, we will verify batches of d^2 learners at a time. While we cannot directly apply the above process to a batch of weak learners (since as described, the process requires learning *all* of X to work), we can turn the verification of the batch into a d dimensional point location problem on poly(d) points. The expected number of queries to verify a batch is then given by Equation (10), and combining this with the query bounds on our weak learner from Theorem II.4 proves that the overall process gives a randomized LDT satisfying the conditions of Theorem I.7. The full details are covered in Corollary 6.6 and Theorem 6.1 of the full paper.

References

[1] Friedhelm Meyer auf der Heide. A polynomial linear search algorithm for the n-dimensional knapsack problem. In *Annual ACM Symposium on Theory of Computing: Proceedings of the fifteenth annual ACM symposium on Theory of computing*, volume 1983, pages 70–79, 1983.

[2] Stefan Meiser. Point location in arrangements of hyperplanes. *Information and Computation*, 106 (2):286–303, 1993.

[3] Jean Cardinal, John Iacono, and Aurélien Ooms. Solving k-SUM using few linear queries. *arXiv preprint arXiv:1512.06678*, 2015.

[4] Daniel Kane, Shachar Lovett, and Shay Moran. Generalized comparison trees for point-location problems. In *International Colloquium on Automata, Languages and Programming*, 2018.

[5] Esther Ezra and Micha Sharir. A nearly quadratic bound for point-location in hyperplane arrangements, in the linear decision tree model. *Discrete & Computational Geometry*, 61(4):735–755, 2019.

[6] Dana Angluin. Queries and concept learning. *Machine learning*, 2(4):319–342, 1988.

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

[8] Daniel M Kane, Shachar Lovett, and Shay Moran. Near-optimal linear decision trees for k-SUM and related problems. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 554–563. ACM, 2018.

[9] Vladimir Vapnik and Alexey Chervonenkis. Theory of pattern recognition, 1974.

[10] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM (JACM)*, 36(4):929–965, 1989.

[11] Steve Hanneke. The optimal sample complexity of pac learning. *The Journal of Machine Learning Research*, 17(1):1319–1333, 2016.

[12] Daniel M Kane, Shachar Lovett, Shay Moran, and Jiapeng Zhang. Active classification with comparison queries. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 355–366. IEEE, 2017.

[13] David Dobkin and Richard J Lipton. Multidimensional searching problems. *SIAM Journal on Computing*, 5(2):181–186, 1976.

[14] Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Learnability and the Vapnik-Chervonenkis dimension. *Journal of the ACM (JACM)*, 36(4):929–965, 1989.

[15] Maria-Florina Balcan, Andrei Broder, and Tong Zhang. Margin based active learning. In *International Conference on Computational Learning Theory*, pages 35–50. Springer, 2007.

[16] Sanjoy Dasgupta. Coarse sample complexity bounds for active learning. In *Advances in neural information processing systems*, pages 235–242, 2006.

[17] Yoav Freund, H Sebastian Seung, Eli Shamir, and

Naftali Tishby. Selective sampling using the query by committee algorithm. *Machine learning*, 28 (2-3):133–168, 1997.

[18] Maria-Florina Balcan and Phil Long. Active and passive learning of linear separators under log-concave distributions. In *Conference on Learning Theory*, pages 288–316, 2013.

[19] Maria-Florina F Balcan and Hongyang Zhang. Sample and computationally efficient learning algorithms under s-concave distributions. In *Advances in Neural Information Processing Systems*, pages 4796–4805, 2017.

[20] Max Hopkins, Daniel M Kane, and Shachar Lovett. The power of comparisons for actively learning linear classifiers. *arXiv preprint arXiv:1907.03816*, 2019.

[21] Lin Chen, Hamed Hassani, and Amin Karbasi. Near-optimal active learning of halfspaces via query synthesis in the noisy setting. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[22] Ibrahim Alabdulmohsin, Xin Gao, and Xiangliang Zhang. Efficient active learning of halfspaces via query synthesis. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[23] Franck Barthe. On a reverse form of the Brascamp-Lieb inequality. *Inventiones mathematicae*, 134(2): 335–361, 1998.

[24] Jürgen Forster. A linear lower bound on the unbounded error probabilistic communication complexity. *Journal of Computer and System Sciences*, 65(4):612–625, 2002.

[25] Eric A Carlen, Elliott H Lieb, and Michael Loss. A sharp analog of young's inequality on s n and related entropy inequalities. *The Journal of Geometric Analysis*, 14(3):487–520, 2004.

[26] Zeev Dvir, Shubhangi Saraf, and Avi Wigderson. Breaking the quadratic barrier for 3-LCCs over the reals. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 784–793, 2014.

[27] Max Hopkins, Daniel Kane, Shachar Lovett, and Gaurav Mahajan. Noise-tolerant, reliable active classification with comparison queries. *arXiv preprint arXiv:2001.05497*, 2020.