Actively Learning Halfspaces without Synthetic Data

Hadley Black[*] *UC San Diego*

Kasper Green Larsen[†] *Aarhus University*

Arya Mazumdar[*] *UC San Diego*

Barna Saha[*] *UC San Diego*

Geelon So[*] *UC San Diego*

September 26, 2025

Abstract

In the classic *point location* problem, one is given an arbitrary dataset $X \subset \mathbb{R}^d$ of n points with *query access* to an unknown halfspace $f : \mathbb{R}^d \to \{0,1\}$, and the goal is to learn the label of every point in X. This problem is extremely well-studied and a nearly-optimal $\widetilde{O}(d \log n)$ query algorithm is known due to Hopkins-Kane-Lovett-Mahajan (FOCS 2020). However, their algorithm is granted the power to query arbitrary points outside of X (point synthesis), and in fact without this power there is an $\Omega(n)$ query lower bound due to Dasgupta (NeurIPS 2004). Nonetheless, query access to arbitrary synthesized data points is unrealistic in many contexts.

Our objective in this work is to design efficient algorithms for learning halfspaces *without point synthesis*. To circumvent the $\Omega(n)$ lower bound, we consider learning halfspaces whose normal vectors come from a known set of size D, and show tight bounds of $\Theta(D + \log n)$. As a corollary, we obtain an optimal $O(d + \log n)$ query deterministic learner for the fundamental class of *decision stumps* (depth-one decision trees, or axis-aligned halfspaces), closing a previous gap of $O(d \log n)$ vs. $\Omega(d + \log n)$ left open in the active learning literature. In fact, our algorithm solves the more general problem of learning a Boolean function f over n elements which is monotone under at least one of D provided *orderings* of these elements. Our technical insight is to exploit the structure in these orderings to essentially perform a binary search in parallel rather than considering each ordering sequentially, and we believe our approach may be of broader interest.

Furthermore, we use our exact learning algorithm to obtain nearly optimal algorithms for PAC-learning. We show that $O(\min(D + \log(1/\varepsilon), 1/\varepsilon) \cdot \log D)$ queries suffice to learn f within error ε, even in a setting when f can be adversarially corrupted on a $c\varepsilon$-fraction of points, for a sufficiently small constant c. This bound is optimal up to a $\log D$ factor, including in the realizable setting.

[*]Supported by NSF TRIPODS Institute grant 2217058 (EnCORE).

[†]Supported by the European Union (ERC, TUCLA, 101125203). Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

1 Introduction

Halfspaces, or linear threshold functions (LTFs), are one of the most fundamental and well-studied objects in computer science and geometry, with numerous applications in machine learning, optimization, and beyond. The halfspace with normal vector $u \in S^{d-1}$ and threshold $t \in \mathbb{R}$ bisects $\mathbb{R}^d$ into two regions, classifying x using the rule $\mathbf{1}(\langle u, x\rangle > t)$. In this work, we consider the following halfspace learning question. Given a dataset $X \subset \mathbb{R}^d$ partitioned into two classes by an unknown halfspace $f\colon \mathbb{R}^d \to \{0,1\}$, we are allowed to adaptively query the label $f(x)$ for any $x \in X$ and our goal is to learn the label of every point (or an all but ε-fraction of points) in X. If an algorithm is allowed to query points outside of X, then we say it uses *point synthesis*.

With point synthesis, there is an information-theoretic lower bound of $\Omega(d \log n)$ queries and there has been extensive work on obtaining upper bounds [adH84, Mei93, CIO16, KLM18, ES19] culminating[1] in a $O(d \log^2 d \log n)$ query algorithm due to [HKLM20]. However, the ability to synthesize and query arbitrary data points is a strong assumption which may be infeasible in many settings, and our primary motivation in this work is to investigate when and how this can be avoided. The bad news is that, in general, without point synthesis no non-trivial algorithm is possible due to the following simple $\Omega(n)$ lower bound observed by [Das04] which holds even in two dimensions: for a set X of n distinct points on the unit circle, it is straightforward to design a family of halfspaces $\{f_x : x \in X\}$ such that f_x uniquely assigns $f_x(x) = 1$ among all points in X, and thus one must query every point in order to learn every label[2].

Learning halfspaces optimally in bounded directions. The above lower bound comes from the fact that there are an unbounded number of *directions* the halfspace can be in, and each direction can be used to separate a single point in X from the rest. Thus, to circumvent this hardness without weakening the assumption of a worst-case point set, it is *necessary* to bound the number of directions parameterizing the halfspace. It is not too hard to see that if $D \le n$ directions are allowed, then the above argument gives an $\Omega(D)$ lower bound and $\Omega(\log n)$ queries are needed (to determine the threshold) even in one dimension, leading to an $\Omega(D + \log n)$ lower bound. On the other hand it is not too challenging to design an algorithm using $O(D \log n)$ queries by projecting X along each allowed direction, and performing a binary search independently using the ordering on X induced by each projection. This algorithm simply exploits the fact that the labels are monotone along the correct direction. In fact, even for the simple special case of *axis-aligned* halfspaces (depth-one decision trees) in $\mathbb{R}^d$, there is a gap of $O(d \log n)$ vs. $\Omega(d + \log n)$ that exists in the active learning literature [HY15]. We investigate the core algorithmic question underlying this gap: *can we exploit structure beyond monotonicity of the labels to go beyond a naive, independent application of binary search?* We give an affirmative answer, designing a new algorithm that achieves the optimal $O(D + \log n)$ query complexity, thus closing this gap.

A new "parallel" binary search learning algorithm. In fact, we design an optimal algorithm for the following generalization of the above problem, which we believe may be of general interest: given a set X of size n, query access to a function $f\colon X \to \{0,1\}$, and D permutations $\sigma_1, \dots, \sigma_D \colon X \to [n]$ such that $f \circ \sigma_{i^\star}^{-1} \colon [n] \to \{0,1\}$ is *monotone* for some $i^\star \in [D]$, we wish to learn $f(X)$ using few queries[3]. A naive approach is to run an independent binary search on each $f \circ \sigma_i^{-1}$, ignoring the shared information across the $f \circ \sigma_i^{-1}$-s provided by a single query. (See Observation 2.3 for a formal argument.) Exploiting this shared information requires carefully considering the structure of the permutations in order to choose queries to

[1]Many works in this area refer to the dual problem of *point location*. Here the goal is to determine the cell containing a hidden point in an arrangement of n hyperplanes with the ability to query on which side of any given hyperplane the point lies. This is well-known to be equivalent to the halfspace learning question and so they are often referred to interchangeably in the literature.

[2]Note that this argument holds for *any choice* of n distinct points on the unit circle. For instance, this demonstrates an $\Omega(n)$ lower bound even for the class of halfspaces whose normal vector comes from an arbitrarily narrow cone.

[3]The halfspace learning problem we consider corresponds to the special case of $X \subset \mathbb{R}^d$ being a set of n points, f being a halfspace whose normal vector is from a known set $V = \{u_1, \dots, u_D\}$ of unit vectors, and σ_i being the ordering on X by sorting according to the length of the projections $\langle x, u_i\rangle$ for each $x \in X$.

effectively perform these binary searches in parallel. We are able to use $O(1)$ queries to either reduce D by one, or reduce n by a constant factor, performing a step of binary search simultaneously on every $f \circ \sigma_i^{-1}$. At a high level, our algorithm and its analysis demonstrate how one can carefully choose queries to obtain significant speedups for analyzing multiple sequences simultaneously. We believe that our ideas for speeding up parallel binary search, and even our algorithm as a black-box, may be useful for obtaining similar speedups over the naive algorithm for other problems.

Main application: decision stumps, or axis-aligned halfspaces. A natural class of halfspaces with bounded directions is the class of *axis-aligned* halfspaces. Such functions are equivalent to *decision stumps* (depth-one decision trees), which classify data by thresholding with respect to a single feature. These classifiers are extremely fundamental, forming the basic building blocks of decision trees, and are commonly used in practice as base learners in ensemble learning techniques such as bagging and boosting [Hol93, OH94, BHS23]. For learning decision stumps over arbitrary point sets in $\mathbb{R}^d$ there is a gap of $O(d \log n)$ vs. $\Omega(d + \log n)$ left open in the active learning literature [HY15]. As a direct corollary of our main result, we obtain a deterministic algorithm using $O(d + \log n)$ queries, thus closing the book on this very basic concept class.

Beyond depth one, there is a large body of research on decision tree learning over the Boolean hypercube, $\{0,1\}^d$ (e.g. [BLQT22, BLT20, BH19, EH89]); however, we note that there is an $\Omega(n)$ query lower bound for learning *Boolean* decision trees of depth at least two[4] when the domain $X \subset \mathbb{R}^d$ is arbitrary. On the other hand, an alternative generalization is the problem of learning *multi-class* decision trees, where each leaf must correspond to a distinct label, over arbitrary $X \subset \mathbb{R}^d$. We do not know of any prior works studying this question and we believe this may be an interesting direction for future work.

Beyond the axis-aligned setting, a more general class of halfspaces where our results have implications are *sparse discrete* halfspaces: given a sparsity parameter $s \in \{1, \dots, d\}$, suppose we restrict the normal vector to lie in $\{-1,0,1\}^d$ (or a larger hypergrid) with at most s non-zero coordinates. The total number of directions for such halfspaces is bounded by $d^{O(s)}$ and so we obtain a learner with query complexity $d^{O(s)} + O(\log n)$ for exactly learning such a halfspace over an arbitrary point set $X \subset \mathbb{R}^d$. This notion of sparse halfspaces has been considered in numerous works (e.g. [MORS09], [AAB16], [MS25] for a small sampling), but not for the problem we consider, to the best of our knowledge.

Learning preliminaries. In addition to exact learning, we design algorithms for approximate learning in both the realizable and tolerant settings. For two Boolean-valued functions $f, g \colon X \to \{0,1\}$ over the same domain, we use the notation $\|f - g\|_1$ to denote their Hamming distance *normalized* by $|X|$. We consider learning a concept class $\mathcal{C} \colon X \to \{0,1\}$ in the settings defined below:

- **Exact learning:** Given $f \in \mathcal{C}$, learn the label of every point in X.
- **Proper, realizable (ε, δ)-PAC learning:** Given $\varepsilon > 0, \delta > 0$, and $f \in \mathcal{C}$, learn $h \in \mathcal{C}$ such that $\Pr_h[\|f - h\|_1 > \varepsilon] < \delta$.
- **Proper, c-tolerant (ε, δ)-PAC learning:** Given $c > 0, \varepsilon > 0, \delta > 0$, and f such that $\|f - g\|_1 \le c\varepsilon$ for some $g \in \mathcal{C}$, learn $h \in \mathcal{C}$ such that $\Pr_h[\|f - h\|_1 > \varepsilon] < \delta$.

Our algorithms for PAC-learning utilize our exact learning algorithm as a black-box, in conjunction with a novel randomized binary search procedure for tolerantly learning monotone functions, and a simple result from property testing of monotone functions.

1.1 Results

We now state our main results. Given $X \subset \mathbb{R}^d$ and a set $V = \{u_1, \dots, u_D\} \subset S^{d-1}$ of D unit vectors, let $\mathcal{H}(V, X)$ denote the set of all (non-homogeneous) halfspaces defined over X with normal vector in V. That

[4]Let $X = \{(i, -i) \colon i \in [-n, n]\} \subset \mathbb{R}^2$ and for each $i \in [-n, n]$ consider the depth two decision tree defined by $f_i(x) = \mathbf{1}(x_1 \ge i \wedge x_2 \ge -i)$. Then, $f_i(i, -i) = 1$ and all other points in X are labeled 0. Thus, learning the labels exactly is equivalent to an unstructured search problem, implying we need $\Omega(n)$ queries.

is, $f \in \mathcal{H}(V, X)$ iff $f(x) = \mathbf{1}(\langle u, x\rangle > t)$ for some $u \in V$ and $t \in \mathbb{R}$. All of our algorithms run in polynomial time.[5]

Theorem 1.1 (D-Directional Halfspace Learning)**.** *Given arbitrary set $X \subset \mathbb{R}^d$ of n points and $V \subset S^{d-1}$ of D unit vectors, the following learning algorithms exist for the concept class $\mathcal{H}(V, X)$.*

1. *A deterministic exact learner using $O(\min(D + \log n, n))$ queries.*
2. *A proper, realizable $(\varepsilon, 0)$-PAC learner using $O(\min(D + \log(\frac{1}{\varepsilon}), \frac{\log D}{\varepsilon}))$ queries.*
3. *A proper c-tolerant (ε, δ)-PAC learner using $O(\min(D + \log(\frac{1}{\varepsilon}), \frac{1}{\varepsilon}) \cdot \log D)$ queries, for any constant $\delta > 0$, and sufficiently small constant $c \in (0, 1)$.*

Decision stumps. Next, we observe that Theorem 1.1 implies improved bounds (in some cases optimal) for decision stumps as an immediate corollary. Given $i \in [d]$ and $t \in \mathbb{R}$, the decision stump $f_{i,t} \colon \mathbb{R}^d \to \{0, 1\}$ is defined as $f_{i,t}(x) = \mathbf{1}(x_i \geq t)$. We denote the class of decision stumps over $\mathbb{R}^d$ as DS_d. The work of [HY15] gives general upper and lower bounds on the query complexity of active learning in terms of a quantity which they call the *star number* of a concept class. For learning decision stumps, their results imply lower and upper bounds of

$$\Omega\left(\min\left(d + \log\frac{1}{\varepsilon}, \frac{1}{\varepsilon}\right) + \min\Big(\log d, \log n\Big)\right) \text{ and } O\Bigg(\min\left(d \log\left(\frac{1}{\varepsilon}\right), \frac{\log d \log(1/\varepsilon)}{\varepsilon}, n\right)\Bigg)$$

queries, respectively for ε-realizable learning with constant success probability. Note that, replacing "d" by "D", this lower bound also holds for the D-directional halfspace learning problem considered in Theorem 1.1, since axis-aligned halfspaces are a special case. For completeness, we give a formal proof of these bounds in Appendix A. For example, the previous known bounds for exact learning are $\Omega(\min(d + \log n), n)$ vs. $O(\min(d \log n, n))$. As an immediate corollary of Theorem 1.1, we obtain the following bounds for learning decision stumps which are tight in the exact learning setting and are within a $O(\log d)$ factor of the lower bound in the approximate learning settings.

Corollary 1.2 (Decision Stump Learning)**.** *Given an arbitrary set $X \subset \mathbb{R}^d$ of n points, the following learning algorithms exist for learning the class DS_d of decision stumps over X.*

1. *A deterministic exact learner using $O(\min(d + \log n, n))$ queries.*
2. *A proper, realizable $(\varepsilon, 0)$-PAC learner using $O(\min(d + \log(\frac{1}{\varepsilon}), \frac{\log d}{\varepsilon}))$ queries.*
3. *A proper, c-tolerant (ε, δ)-PAC learner using $O(\min(d + \log(\frac{1}{\varepsilon}), \frac{1}{\varepsilon}) \cdot \log d)$ queries, for any constant $\delta > 0$, and sufficiently small constant $c \in (0, 1)$.*

Organization. We prove our main result, item (1) of Theorem 1.1, in Section 2. This result is used as a black-box in the proof of items (2) and (3) of Theorem 1.1, which we prove in Section 3.

2 Exact Learning

In this section we prove item (1) of Theorem 1.1. That is, we design a deterministic exact learning algorithm for halfspaces whose normal vector belongs to a known set $V = \{u_1, \ldots, u_D\} \subset S^{d-1}$ of size D, with the optimal $O(\min(D + \log n, n))$ query complexity. In fact, we consider a generalized formulation of our problem. Given a set of n points $X \subset \mathbb{R}^d$, let $\sigma_1, \ldots, \sigma_D \colon X \to [n]$ denote the D permutations on X induced by ordering the projections of each $x \in X$ onto each $u \in V$[6]. That is, $\sigma_i(x) = j$ iff x has the j'th smallest projection, $\langle x, u_i\rangle$. Then, a labeling $f \colon X \to \{0, 1\}$ is consistent with a halfspace $\mathbf{1}(\langle x, u_{i^\star}\rangle > t^\star)$ for some $i^\star \in [D], t^\star \in \mathbb{R}$ iff f is *non-decreasing* when X is ordered according to $\sigma_{i^\star}$. That is, $f \circ \sigma_{i^\star}^{-1} \colon [n] \to \{0, 1\}$ is a monotone function. Therefore, it suffices to consider the following generalized learning problem.

[5]Since our focus is on query complexity we do not describe how to optimize the time complexity of our algorithms, nor do we provide a detailed runtime analysis.

[6]Ties can be broken arbitrarily.

Problem 2.1 (Learning Shuffled Monotone Functions). *Let X be a set of n elements with a Boolean labeling $\ell\colon X \to \{0,1\}$. We are given D permutations $\sigma_1, \dots, \sigma_D\colon X \to [n]$ with the guarantee that for some unknown $i^\star \in [D]$, the labeling $\ell \circ \sigma_{i^\star}^{-1}\colon [n] \to \{0,1\}$ is non-decreasing. The problem of learning shuffled monotone functions asks to learn the labeling of all points in X using as few queries to ℓ as possible.*

In particular, by the argument preceding Problem 2.1, any algorithm for learning shuffled monotone functions with n points and D permutations immediately implies an algorithm for learning $\mathcal{H}(V, X)$ where $X \subset \mathbb{R}^d$ is an arbitrary set of n points and $V \subset S^{d-1}$ is an arbitrary set of D unit vectors.

We start by describing a simple algorithm that achieves $O(D \log n)$ query complexity and then move on to obtaining the optimal $O(D + \log n)$ query learner in Section 2.1. We begin with a simple claim that gives a useful and general subroutine.

Claim 2.2. *Let $f\colon X \to \{0,1\}$ and let $C\colon X \to \{0,1\}$ be a collection of "candidate" labeling functions over X. There is an algorithm* $\mathsf{Contender}(f, C, X)$ *using at most $|C|$ label queries to f which returns some $h \in C$ such that if $f \in C$, then $h = f$.*

Proof. Suppose there exists $x \in X$ such that $h_1(x) \neq h_2(x)$ for some $h_1, h_2 \in C$. Then by querying $f(x)$, we either observe that $h_1(x) \neq f(x)$, in which case we remove h_1 from C, or we observe $h_2(x) \neq f(x)$, in which case we remove h_2 from C. Otherwise, all $h \in C$ define the same labeling over X and now we return this labeling. Observe that if $f \in C$, then it will never be removed by this process. Thus, this algorithm satisfies the guarantee of the claim. □

Now, given $i \in [D]$ and $j \in [n]$, let $h_j^{(i)}\colon X \to \{0,1\}$ be defined by $h_j^{(i)}(x) = \mathbf{1}(\sigma_i(x) \geq j)$. In the shuffled monotone function learning problem (Problem 2.1), we are guaranteed that there is some $i^* \in [D]$, $j^* \in [n]$ such that $h_{j^*}^{(i^*)} = \ell$. We first observe that combining Claim 2.2 with a basic binary search yields a simple $O(D \log n)$ query algorithm.

Observation 2.3. *There is a deterministic algorithm for learning shuffled monotone functions using $O(\min(D \log n, n))$ queries.*

Proof. If $n < D \log n$, we can simply query the label of every point. Otherwise, the algorithm is defined as follows. Initialize a candidate set $C \leftarrow \emptyset$. For each $i \in [D]$, perform a standard binary search on $\ell \circ \sigma_i^{-1}\colon [n] \to \{0,1\}$, which returns an index $j_i \in \{0\} \cup [n]$ such that $\ell \circ \sigma_i^{-1}(j_i) = 0$ and $\ell \circ \sigma_i^{-1}(j_i + 1) = 1$. (Define $\ell \circ \sigma_i^{-1}(0) = 0$, $\ell \circ \sigma_i^{-1}(n+1) = 1$.) Add $h_{j_i}^{(i)}$ to C. Afterwards, run the procedure of Claim 2.2 on C.

By our use of binary search we are guaranteed that $j_{i^\star}$ satisfies $\ell \circ \sigma_{i^\star}^{-1}(j) = \mathbf{1}(j > j_{i^\star})$. Then, the correctness of the algorithm follows by Claim 2.2. The number of queries made in the search phase is $O(D \log n)$ and the procedure of Claim 2.2 uses at most D queries. □

On the other hand, the lower bound of Theorem A.1 asserts that $\Omega(\min(D + \log n, n))$ are needed (set $\varepsilon < 1/n$). Note that the algorithm of Observation 2.3 treats each $\ell \circ \sigma_i^{-1}\colon [n] \to \{0,1\}$ separately, by independently running a binary search procedure on each. In particular, it ignores the fact that a single query reveals information in every $\ell \circ \sigma_i^{-1}$ simultaneously. Designing an algorithm which exploits this requires careful consideration of the structure of the permutations $\sigma_1, \dots, \sigma_D$ to choose which points to query. We show that by carefully considering this structure we can obtain an algorithm making $O(\min(D + \log n, n))$ queries, thus closing this gap. At a high level, this query complexity comes from the fact that we are able to spend only $O(1)$ queries to either remove a permutation from consideration (thus reducing D by one), or find a large set of points to safely remove from consideration (reducing n by a constant factor), effectively performing a step of binary search simultaneously in every $\ell \circ \sigma_i^{-1}$.

2.1 Optimal Deterministic Algorithm

In this section we prove the following theorem, which immediately implies item (1) of Theorem 1.1.

Theorem 2.4. *For any D, n, there is a deterministic algorithm (Alg. 1),* $\mathsf{ShuffledMonotoneLearner}$*, that exactly learns a shuffled monotone function using $O(D + \log n)$ label queries.*

Proof. Our goal is to, for every $i \in [D]$, either (1) determine that $i \neq i^\star$ by finding a "decreasing pair" $x, y \in X$ such that $\ell(x) = 1, \ell(y) = 0$ and $\sigma_i(x) < \sigma_i(y)$ or (2) find a "boundary pair" $x, y \in X$ such that $\ell(x) = 0$, $\ell(y) = 1$ and $\sigma_i(y) - \sigma_i(x) = 1$. If (2) holds for some i, then we can put the candidate hypothesis $h^{(i)}_{\sigma_i(y)}$ into a "candidate" set C and remove i from consideration. Once (1) or (2) has been satisfied for every $i \in [D]$, we are guaranteed that $\ell \in C$ and we can then use the procedure of Claim 2.2 using at most D queries to recover the labeling, ℓ.

Let $C \leftarrow \emptyset$ denote the candidate set which is initially empty and let $S \leftarrow [D]$ denote the current set of coordinates under consideration which is initially $[D]$. Let $Z \leftarrow X$ denote the set of points under consideration which is initially X. Let x^*, y^* denote the boundary points for i^*. That is, $\sigma_{i^*}(y^*) = \sigma_{i^*}(x^*) + 1$ and $\ell(x^*) = 0$, $\ell(y^*) = 1$. The following lemma gives the main subroutine.

Lemma 2.5. *Let $S \subseteq [D]$ such that $i^\star \in S$ and let $Z \subseteq X$ such that $x^\star, y^\star \in Z$. Let $n := |Z|$. Suppose we are given for each $i \in S$, an interval $W_i \subset [n]$ with the guarantee that (a) $\sigma_{i^\star}(x^\star), \sigma_{i^\star}(y^\star) \in W_{i^\star}$ and (b) $\min_{i \in S} |W_i| \leq n/3$. Then, there is a procedure* RemoveOrReduce$(S, Z, (W_i : i \in S))$ *using at most 3 queries which returns one of the following:*

1. *An index $i \in S$ such that $i \neq i^\star$.*

2. *An index $i \in S$ and a pair $x, y \in Z$ such that $\sigma_i(y) = \sigma_i(x) + 1$ and $\ell(x) = 0$, $\ell(y) = 1$.*

3. *A set $Z' \subset Z$ of size $|Z'| \leq \frac{2n}{3}$ such that $x^\star, y^\star \in Z'$. Moreover, in this case the procedure returns the labeling $\ell(Z \setminus Z')$ on all points outside of Z'.*

If we remove any of the assumptions that $i^\star \in S$, $x^\star, y^\star \in Z$, or $\sigma_{i^\star}(x^\star), \sigma_{i^\star}(y^\star) \in W_{i^\star}$, then the procedure still returns either $i \in S$ such that $i \neq i^\star$ or a set $Z' \subset Z$ of size $|Z'| \leq \frac{2n}{3}$.

We defer the proof of Lemma 2.5 to Section 2.2 and proceed with the proof of Theorem 2.4. The algorithm is defined in Alg. 1.

Analysis. If the labeling is constant, then clearly the algorithm is correct by step (4) and uses at most $2D$ queries. Otherwise, we will assume there is at least one point with each label. In particular $x^\star$ is the max 0-labeled point in $\sigma_{i^\star}$ and $y^\star$ is the min 1-labeled point in $\sigma_{i^\star}$. The following is the key invariant of the algorithm that guarantees correctness.

Claim 2.6 (Alg. 1 invariant)**.** *Throughout the execution of Alg. 1, we have that if $i^\star \in S$, then $x^\star, y^\star \in Z$, and otherwise we have $\ell = h^{i^\star}_{\sigma_{i^\star}(y^\star)} \in C$.*

Proof. We prove the claim by induction on the size of S and the size of Z. For the base case, when $Z = X$ and $S = [D]$, the claim clearly holds. Now, in steps (7), (8), and (10) a coordinate i is only removed from S if it is known that $i \neq i^\star$ (the correctness of (10) is by Lemma 2.5). In step (11), when a coordinate i is removed from S, a hypothesis $h^{(i)}_j$ is placed into C with the guarantee that if $i = i^\star$, then $h^{(i)}_j = \ell$ by Lemma 2.5 item (2). Thus, in all cases when the size of S reduces, we maintain the invariant. When the size of Z reduces in step (12) it is with the guarantee that $x^\star, y^\star$ remain in the set by Lemma 2.5 item (3). Therefore, the invariant is maintained in all cases. $\square$

By construction of the points $z_1, \ldots, z_4$ in step (6), either the size of S is reduced by one, or the intervals constructed in step (8) satisfy $\min_{i \in S} |W_i| \leq |W_k| < n/3$. Moreover, the interval $W_{i^\star}$ is guaranteed to contain $\sigma_{i^\star}(x^\star)$ and $\sigma_{i^\star}(y^\star)$ and so the collection of intervals $(W_i : i \in S)$ satisfies the conditions for the subroutine Lemma 2.5. Thus, we always make progress either by decreasing the size of S by one, or by decreasing the size of Z by $n/3$. Thus, the total number of queries is $O(D + \log n)$ and given Claim 2.6 and Claim 2.2, the algorithm always returns the correct solution. $\square$

Algorithm 1: Deterministic Algorithm for Learning Shuffled Monotone Functions

```
1  Input: A set $X$ of size $|X| = n$, permutations $\sigma_1, \ldots, \sigma_D : X \to [n]$, and query access to a labeling
     $\ell : X \to \{0,1\}$ such that $\ell \circ \sigma_{i^\star}^{-1}$ is non-decreasing for some $i^\star \in [D]$;
2  Output: A full description of the labeling function $\ell$;
3  Initialize $S \leftarrow [D]$, $Z \leftarrow X$, $C \leftarrow \emptyset$;
4  For every $i \in S$, query $\sigma_i^{-1}(1), \sigma_i^{-1}(n)$. If every such query returns the same label, then halt and
     assert that all points in $X$ have this label;
5  while $n > 10$ and $S \neq \emptyset$ do
6      Choose $k \in S$ arbitrarily. Let $z_1 = \sigma_k^{-1}(1)$ and $z_4 = \sigma_k^{-1}(n)$. Choose $z_2, z_3$ such that
         $|\sigma_k(z_r) - \sigma_k(z_{r+1})| < n/3$ for all $r \in \{1,2,3\}$. Query $z_1, z_2, z_3, z_4$;
7      If $\ell(z_1) = 1$ or $\ell(z_4) = 0$, then set $S = S \setminus \{k\}$, and continue the While-loop from line (5);
8      Else we have at least one of each label among $z_1, \ldots, z_4$. For each $i \in S$, let $a_i$ be the maximum
         index where a 0-label is observed and $b_i$ the minimum index where a 1-label is observed among
         $z_1, z_2, z_3, z_4$ in $\sigma_i$. If $a_i > b_i$ for some $i \in S$, then set $S = S \setminus \{i\}$ and continue the While-loop
         from line (5). Otherwise, set $W_i = [a_i, b_i]$ for every $i \in S$. Note that in this case $\ell \circ \sigma_i^{-1}$ is
         non-decreasing on $\{\sigma_i(z_1), \sigma_i(z_2), \sigma_i(z_3), \sigma_i(z_4)\}$ for every $i \in S$ and therefore
         $\min_{i \in S} |W_i| \leq b_k - a_k < n/3$;
9      Run the procedure RemoveOrReduce$(S, Z, (W_i : i \in S))$ from Lemma 2.5;
10     If the procedure returns $i \in S$ as in Case 1 of Lemma 2.5, then set $S \leftarrow S \setminus \{i\}$;
11     If the procedure returns $i \in S$ and a pair $x, y \in Z$ as in Case 2 of Lemma 2.5, then set
         $S \leftarrow S \setminus \{i\}$ and $C \leftarrow C \cup \{h^{(i)}_{\sigma_i(y)}\}$;
12     If the procedure returns $Z'$ as in Case 3 of Lemma 2.5, then set $Z \leftarrow Z'$ and $n = |Z'|$;
13 end
14 If $S \neq \emptyset$, then we have $n \leq 10$. In this case, query the label of every point in $Z$. For each $i \in S$, if
     we find $x, y$ such that $\ell(x) = 0, \ell(y) = 1$ and $\sigma_i(y) = \sigma_i(x) + 1$, set $C \leftarrow C \cup \{h^{(i)}_{\sigma_i(y)}\}$;
15 Run the procedure Contender$(C, X)$ from Claim 2.2 and return the hypothesis that it outputs;
```

2.2 Main Subroutine: Proof of Lemma 2.5

Proof. We begin with some notation. Let $t := |\{x \in Z : \ell(x) = 1\}|$ denote the Hamming weight of the labeling. We are given an interval W_i for each $i \in S$ with the guarantee that $\min_{i \in S} |W_i| < n/3$ and $\sigma_{i^\star}(x^\star), \sigma_{i^\star}(y^\star) \in W_{i^\star}$. Note this implies $t \in W_{i^\star}$. We will use the notation "$x < W_i$" or "$x > W_i$" to mean that x lies entirely below or above the interval W_i.

Since $t \in W_{i^\star}$, we know that $\ell(x) = 0$ whenever $\sigma_{i^\star}(x) < W_{i^\star}$ and similarly $\ell(x) = 1$ whenever $\sigma_{i^*}(x) > W_{i^\star}$. Thus, if we ever discover some i and x where this does not hold, then we know $i \neq i^\star$ and we can safely return i which satisfies the guarantee specified in item (1) of the lemma. With this in mind, we define the following subsets of Z:

- Blue points: $B = \{x \in Z : \exists i \in S, \sigma_i(x) < W_i\}$.
- Red points: $R = \{x \in Z : \exists i \in S, \sigma_i(x) > W_i\}$.
- Purple points: $P = \{x \in Z : \sigma_i(x) \in W_i, \forall i \in S\}$.

The utility of blue and red points is given by the following claim.

Claim 2.7 (Blue/red points). *(a) Suppose we query $x \in B$ and we observe that $\ell(x) = 1$. Then we can find a coordinate $i \in S$ satisfying item (1) of the lemma. (b) Similarly, suppose we query $x \in R$ and we observe that $\ell(x) = 0$. Then we can find a coordinate $i \in S$ satisfying item (1) of the lemma.*

Proof. Suppose $x \in B$ and we observe $\ell(x) = 1$. By definition of B, there exists $i \in S$ such that $\sigma_i(x) < W_i$. We are guaranteed that $i \neq i^*$ and so we have satisfied item (1) of the lemma. Similarly, suppose $x \in R$

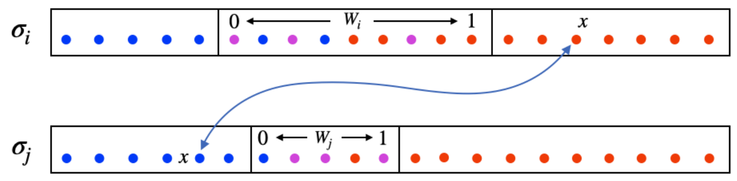

Figure 1: Accompanying illustration for case 1 in the proof of Lemma 2.5. Here $x > W_i$ and $x < W_j$ and so by querying $\ell(x)$ we rule out one of σ_i, σ_j as the monotone permutation. Thus, after case 1 we may assume $B \cap R = \emptyset$ and so the red, blue, purple coloring scheme is well-defined.

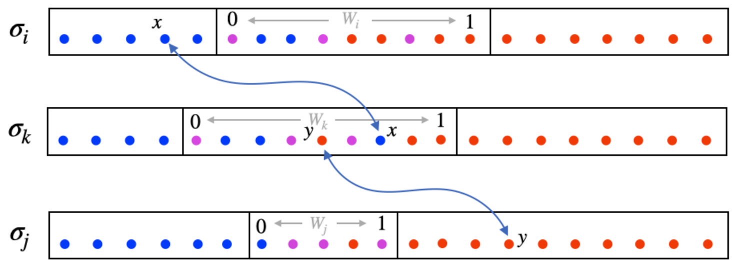

Figure 2: Accompanying illustration for case 2 in the proof of Lemma 2.5. Here, a red point y appears before a blue point x in σ_k. By definition of the colors, we have $x < W_i$ and $y > W_j$ for some i, j. Again by querying the labels of x, y we rule out one of $\sigma_i, \sigma_j, \sigma_k$ as the monotone permutation. The picture for case 3 is exactly the same, except that x appears just before y in σ_k. Here, when $\ell(x) = 0, \ell(y) = 1$ we instead have found a "boundary pair" for σ_k, satisfying item (2) of Lemma 2.5.

and we observe that $\ell(x) = 0$. Then by definition of R, there exists $i \in S$ such that $\sigma_i(x) > W_i$. We are guaranteed that $i \neq i^*$ and so we have satisfied item (1) of the lemma. $\square$

We now continue with the proof of Lemma 2.5.

Case 1: Suppose there exists $x \in B \cap R$. Then we query this point to obtain $\ell(x)$. If $\ell(x) = 1$, then we satisfy item (a) of Claim 2.7. Otherwise, we satisfy item (b) of Claim 2.7. In either case we are guaranteed to find some $i \neq i^*$ and so we satisfy item (1) of the lemma. This case uses 1 query.

(Refer to Fig. 1.) Thus, we may now assume that $B \cap R = \emptyset$, and in particular we have $Z = B \sqcup P \sqcup R$. We now define the blue and red "boundary points" of σ_i as

$$\partial_{i,B} = \arg\max_{x \in B} \sigma_i(x) \text{ and } \partial_{i,R} = \arg\min_{x \in R} \sigma_i(x).$$

Case 2: Suppose that for some $i \in S$ we have $\sigma_i(\partial_{i,R}) < \sigma_i(\partial_{i,B})$. I.e. there is a red point appearing before a blue point in σ_i. Then, we query $x := \partial_{i,B}$ and $y := \partial_{i,R}$. If $\ell(x) = 1$, then we satisfy item (a) of Claim 2.7. If $\ell(y) = 0$, then we satisfy item (b) of Claim 2.7. In either case we are guaranteed to find some $i' \neq i^*$ and so we satisfy item (1) of the lemma. Otherwise, we have $\ell(x) = 0$ and $\ell(y) = 1$ and since $\sigma_i(y) < \sigma_i(x)$ we know that $i \neq i^*$ and so we can simply return i satisfying item (1) of the lemma. This case uses 2 queries. (Refer to Fig. 2.)

Thus, we may now assume that $\sigma_i(\partial_{i,B}) < \sigma_i(\partial_{i,R})$ for every $i \in S$. I.e. in every σ_i all blue points appear before all red points.

Case 3: Suppose that $P = \emptyset$. In this case we pick an arbitrary $i \in S$ and observe that $\sigma_i(\partial_{i,R}) = \sigma_i(\partial_{i,B}) + 1$. Thus, we query $x := \partial_{i,B}$ and $y := \partial_{i,R}$. If $\ell(x) = 1$ or $\ell(y) = 0$, then again we satisfy the premise of Claim 2.7

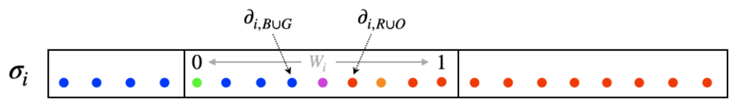

Figure 3: Illustration for the blue, red, orange, green, purple coloring scheme for a single permutation. The left-most point in the interval W_i was previously purple, and then was colored green since it appears before some blue point. Similarly, the orange point was previously purple, but appears after some red point.

and can return some $i' \in S$ satisfying item (1) of the lemma. Otherwise, we have $\ell(x) = 0$ and $\ell(y) = 1$ and thus we can return i and x, y which together satisfy item (2) of the lemma. This case uses 2 queries.

Thus, we now assume that $P \neq \emptyset$. We now further partition P into three sets as follows:

- Green points: $G = \{x \in P \colon \exists i \in S,\, \sigma_i(x) < \sigma_i(\partial_{i,B})\}$.
- Orange points: $O = \{x \in P \colon \exists i \in S,\, \sigma_i(x) > \sigma_i(\partial_{i,R})\}$.
- Strong purple points: $P' = \{x \in P \colon \sigma_i(\partial_{i,B}) < \sigma_i(x) < \sigma_i(\partial_{i,R}),\, \forall i \in S\}$.

In words, green points are the "non-blue" points which come before the blue boundary in *some permutation* and orange points are the "non-red" points that come after the red boundary in *some permutation*. The remaining "strong" purple points lie strictly between the blue and red boundaries in *every permutation*. The utility of green and orange points is given by the following claim.

Claim 2.8 (Green/orange points)**.** *(a) Suppose we query $x \in G$ and we observe that $\ell(x) = 1$. Then we can make at most one more query and find a coordinate $i \in S$ satisfying item (1) of the lemma. (b) Similarly, suppose we query $x \in O$ and we observe that $\ell(x) = 0$. Then we can make one more query and find a coordinate $i \in S$ satisfying item (1) of the lemma.*

Proof. Suppose $z \in G$ and we observe that $\ell(z) = 1$. By definition of G, there is some $x \in B$ such that $\sigma_i(z) < \sigma_i(x)$. Thus, we query this x and obtain $\ell(x)$. If $\ell(x) = 0$, then we know $i \neq i^*$ and so we can return i which satisfies item (1) of the lemma. Otherwise, if $\ell(x) = 1$, then by case (a) of Claim 2.7 we can again find an $i' \in S$ satisfying item (1) of the lemma.

Suppose $z \in O$ and we observe that $\ell(z) = 0$. By definition of O, there is some $x \in R$ such that $\sigma_i(x) < \sigma_i(z)$. Thus, we query this x and obtain $\ell(x)$. If $\ell(x) = 1$, then we know $i \neq i^*$ and so we can return i which satisfies item (1) of the lemma. Otherwise, if $\ell(x) = 0$, then by case (b) of Claim 2.7 we can again find an $i' \in S$ satisfying item (1) of the lemma. □

We now continue with the proof of Lemma 2.5.

Case 4: Suppose that there exists $z \in G \cap O$. Then we query z and obtain $\ell(z)$. If $\ell(z) = 1$, then we satisfy item (a) of Claim 2.8. Otherwise, if $\ell(z) = 0$, then we satisfy item (b) of Claim 2.8. In either case we find some $i \in S$ satisfying item (1) of the lemma. This case uses at most 2 queries.

Thus, we now assume that $G \cap O = \emptyset$, and in particular we have $P = G \sqcup P' \sqcup O$. We now define the blue-green and red-orange "boundary points" of σ_i as

$$\partial_{i,B\cup G} = \operatorname*{arg\,max}_{x \in B\cup G} \sigma_i(x) \text{ and } \partial_{i,R\cup O} = \operatorname*{arg\,min}_{x \in R\cup O} \sigma_i(x).$$

Case 5: Suppose that for some $i \in S$ we have $\sigma_i(\partial_{i,R\cup O}) < \sigma_i(\partial_{i,B\cup G})$. Then we query $x := \partial_{i,B\cup G}$ and $y := \partial_{i,R\cup O}$. If $\ell(x) = 1$, then we find $i' \in S$ satisfying item (1) of the lemma by item (a) of either Claim 2.7 or Claim 2.8, depending on whether x is blue or green. Similarly, if $\ell(y) = 0$, then we find $i' \in S$ satisfying item (1) of the lemma by item (b) of either Claim 2.7 or Claim 2.8, depending on whether y is red or orange. Otherwise, if $\ell(x) = 0$ and $\ell(y) = 1$, then we know $i \neq i^*$ and so we again satisfy item (1). This cases uses at most 3 queries. (Refer to Fig. 4.)

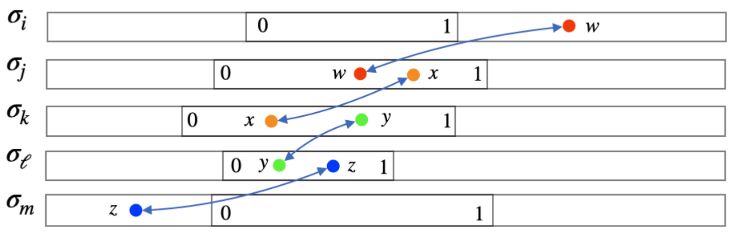

Figure 4: Accompanying example illustration for case 5 in the proof of Lemma 2.5 in which an orange point x appears before a green point y in some σ_k. Since y is green, y appears before a blue point z in some σ_ℓ, which in turn means that $z < W_m$ for some $m \in [D]$. Similarly, since x is orange, it appears after a red point w in some σ_j, which means that $w > W_i$ for some $i \in [D]$. Querying these points reveals that one of $\sigma_i, \sigma_j, \sigma_k, \sigma_\ell, \sigma_m$ does not have monotone labels. An accompanying illustration for case 6 is exactly the same, except that y appears just before x in σ_k.

Thus, we may now assume that $\sigma_i(\partial_{i,B\cup G}) < \sigma_i(\partial_{i,R\cup O})$. I.e. in every σ_i all blue and green points appear before all red and orange points.

Case 6: Suppose that $P' = \emptyset$. This case is similar to case 3. We pick an arbitrary $i \in S$ and observe that $\sigma_i(\partial_{i,R\cup O}) = \sigma_i(\partial_{i,B\cup G}) + 1$. Thus, we query $x := \partial_{i,B\cup G}$ and $y := \partial_{i,R\cup O}$ to obtain $\ell(x)$ and $\ell(y)$. If $\ell(x) = 1$ or $\ell(y) = 0$, then we again satisfy the premise of item (a) or (b) of either Claim 2.7 or Claim 2.7 and we can return some $i' \in S$ satisfying item (1) of the lemma. Otherwise, we have $\ell(x) = 0$ and $\ell(y) = 1$ and thus we can return i and x, y which together satisfy item (2) of the lemma. This case uses at most 3 queries.

Thus, we may now assume that $P' \neq \emptyset$. Recall that by definition of such points ("strong purple points"), any point $x \in P'$ separates all blue points from all red points in *every permutation*. We will use this very strong property to satisfy item (3) of the lemma statement. Observe that in light of case 1, we know that $Z = B \sqcup P \sqcup R$ and so $|B| + |P| + |R| = n$. Moreover, by definition of P and the bound $\min_{i\in S} |W_i| < n/3$ given by assumption in the lemma statement, we have $|P| \leq \min_{i\in S} |W_i| < n/3$. Thus, we are guaranteed that either $|B| \geq n/3$ or $|R| \geq n/3$.

Case 7: Suppose $|B| \geq n/3$. In this case we wish to "throw away" B and let $Z' = P \cup R$. Recall the definition of P' and the fact that $P' \neq \emptyset$ (in light of case 6). Pick an arbitrary $i \in S$ and let $z = \arg\min_{z\in P'} \sigma_i(z)$ denote the left-most strong purple point for σ_i. Let x be its left-neighbor, i.e. $\sigma_i(x) = \sigma_i(z) - 1$. Observe that $x \in B \cup G$. We query z and x to obtain $\ell(z)$ and $\ell(x)$. If $\ell(x) = 1$, then we can satisfy item (1) of the lemma by either Claim 2.7 or Claim 2.8 depending on whether x is blue or green. Thus, let's assume $\ell(x) = 0$. Then, if $\ell(z) = 1$, we can return i with x, z which together satisfy item (2) of the lemma. Otherwise, we have $\ell(z) = 0$. Now, since $z \in P'$, in every σ_i, all blue points appear before z. Moreover, in σ_{i^*} we are guaranteed that the boundary points x^* and y^* appear at, or after, z. Putting these observations together, we know that $x^*, y^* \in P \cup R$. Thus, we return $Z' = P \cup R$ satisfying item (3) of the lemma statement. This case uses at most 3 queries.

Case 8: Suppose $|R| \geq n/3$. In this case we wish to "throw away" R and let $Z' = P \cup B$. Recall the definition of P' and the fact that $P' \neq \emptyset$ (in light of case 6). Pick an arbitrary $i \in S$ and let $z = \arg\max_{z\in P'} \sigma_i(z)$ denote the right-most strong purple point for σ_i. Let y be its right-neighbor, i.e. $\sigma_i(y) = \sigma_i(z) + 1$. Observe that $y \in R \cup O$. We query z and x. If $\ell(y) = 0$, then we can satisfy item (1) of the lemma by either Claim 2.7 or Claim 2.8 depending on whether y is red or orange. Thus, let's assume $\ell(y) = 1$. Then, if $\ell(z) = 0$, we can return i with x, z which together satisfy item (2) of the lemma. Otherwise, we have $\ell(z) = 1$. Now, since $z \in P'$, in every σ_i, all red points appear after z. Moreover, in σ_{i^*} we are guaranteed that the boundary points x^* and y^* appear at, or before, z. Putting these observations together, we know that $x^*, y^* \in P \cup B$. Thus, we return $Z' = P \cup B$ satisfying item (3) of the lemma statement. This case uses at most 3 queries. (Refer to Fig. 5.)

This completes the proof. □

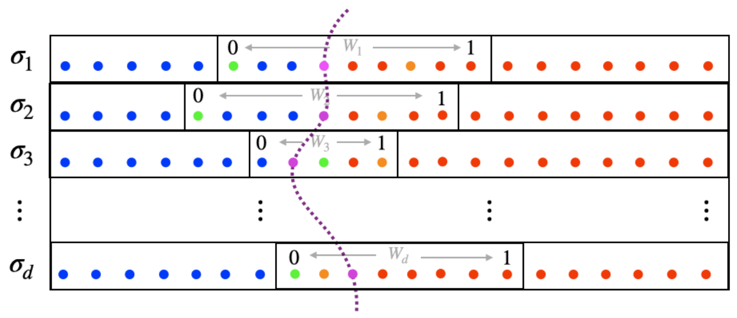

Figure 5: Accompanying illustration for cases 7 and 8 in the proof of Lemma 2.5. Here, we have found a single purple point which separates all blue points from all red points in every permutation simultaneously.

3 Approximate Learning

In this section we consider ε-learning a halfspace over an arbitrary set $X \subset \mathbb{R}^d$ of n points whose normal vector comes from a known set $V = \{u_1, \ldots, u_D\} \subset S^{d-1}$ of D unit vectors. We prove items (3) and (2) of Theorem 1.1 in Section 3.1 and Section 3.2, respectively.

Our algorithms in this section crucially invoke our algorithm for exactly learning shuffled monotone functions obtained in Section 2.1. This algorithm is combined with a randomized binary search procedure which gives a tolerant learner in the 1-dimensional setting, along with a basic technique from property testing of monotone functions, and a careful probabilistic analysis.

Throughout the section, let $\mathcal{M}$ denote the class of monotone Boolean functions over $[n]$, and for $f\colon [n] \to \{0,1\}$, let $d(f,\mathcal{M}) = \min_{g \in \mathcal{M}} \|f-g\|_1$ denote the minimum Hamming distance of f to any monotone function. Note that here $\|f-h\|_1 = |\{j \in [n] \colon f(j) \neq h(j)\}| \cdot n^{-1}$. We use the notation h_S to denote the restriction of h to a subset S of its domain.

3.1 Tolerant Learning

The following lemma gives an optimal c-tolerant algorithm for learning monotone functions in 1-dimension, for sufficiently small constant c. We use this algorithm as a subroutine for our ε-learners in both the tolerant and realizable settings.

Lemma 3.1 (Randomized Binary Search)**.** *Suppose we are given query access to a function $f\colon [n] \to \{0,1\}$ such that $d(f,\mathcal{M}) \leq \varepsilon$. There is a randomized algorithm $\mathcal{A}$ using $O(\log(1/\varepsilon) + \log(1/\delta))$ queries which returns a monotone function $g\colon [n] \to \{0,1\}$ such that $\|f-g\|_1 \leq 10\varepsilon/\delta$ with probability at least $1-\delta$.*

We defer the proof of Lemma 3.1 to Section 3.3. Recall that $\mathcal{H}(V,X)$ denotes the set of all (non-homogeneous) halfspaces defined over X with normal vector in V.

Theorem 3.2. *Let $X \subset \mathbb{R}^d$ be an arbitrary set of n points and $V \subset S^{d-1}$ an arbitrary set of D unit vectors. For any $\varepsilon, \delta > 0$, there is a randomized algorithm (Alg. 2) which, given query access to $f\colon X \to \{0,1\}$ such that $\|f-g\|_1 \leq \frac{\varepsilon\delta}{400}$ for some $g \in \mathcal{H}(V,X)$, uses $O(\ln(D/\delta) \cdot \min(D + \log(\frac{1}{\varepsilon\delta}), \frac{1}{\varepsilon\delta}))$ queries and returns a halfspace $h \in \mathcal{H}(V,X)$ such that $\|f-h\|_1 \leq \varepsilon$ with probability $1-2\delta$.*

Proof. Our algorithm is defined in Alg. 2. Let $\sigma_1, \ldots, \sigma_D\colon X \to [n]$ denote the D permutations on X induced by ordering the projections of each $x \in X$ onto each $u \in V$. That is, $\sigma_i(x) = j$ iff x has the j'th smallest projection, $\langle x, u_i \rangle$. First, since $\|f-g\|_1 \leq \frac{\varepsilon\delta}{400}$ for some $g \in \mathcal{H}(V,X)$, it follows that $d(f \circ \sigma_{i^\star}^{-1}, \mathcal{M}) \leq \frac{\varepsilon\delta}{400}$ for some $i^\star \in [D]$. The following claim (whose proof we defer momentarily) is the main technical component of the proof of correctness.

Claim 3.3. *With probability $1-\delta$, the index i_R obtained in line (8) satisfies $d(f \circ \sigma_{i_R}^{-1}, \mathcal{M}) \leq \varepsilon\delta/10$.*

```
Algorithm 2: Tolerant Learner for Decision Stumps
1  Input: A set $X \subset \mathbb{R}^d$ of size $|X| = n$, a set $V = \{u_1, \dots, u_D\} \subset S^{d-1}$ of unit vectors, and query
     access to $f\colon X \to \{0,1\}$ such that $\|f - g\|_1 \leq \frac{\varepsilon\delta}{400}$ for some $g \in \mathcal{H}(V, X)$;
2  Output: A halfspace $h \in \mathcal{H}(V, X)$ such that $\|f - h\|_1 \leq \varepsilon$ with probability $1 - \delta$;
3  For $i \in [D]$, let $\sigma_i\colon X \to [n]$ be the ordering on $X$ induced by the projections of $X$ onto $u_i$;
4  for $j \leq 50 \ln(D/\delta)$ do
5      Draw a set $S_j \subset X$ of $s := \frac{40}{\varepsilon\delta}$ iid uniform samples from $X$;
6      Run the exact learning algorithm ShuffledMonotoneLearner$(f_{S_j}, S_j, \sigma_1, \dots, \sigma_D)$ of Theorem 2.4 on
         the restriction $f_{S_j}$ using $O(\min(D + \log(\frac{1}{\varepsilon\delta}), \frac{1}{\varepsilon\delta}))$ queries and let $f'_{S_j}\colon S_j \to \{0,1\}$ denote the
         labeling of $S_j$ it returns;
7      For each $i \in [D]$, if $f'_{S_j} \circ \sigma_D$ is non-decreasing on $S_j$, set $Z_{ij} = 1$. Else, set $Z_{ij} = -1$;
8  end
9  For each $i \in [D]$, let $Z_i = \sum_j Z_{ij}$ and set $i_R \leftarrow \arg\max_i Z_i$;
10 Run RandomBinarySearch$(f \circ \sigma_{i_R}^{-1}, \varepsilon\delta/10, \delta)$, and output the labeling function it returns;
```

By Claim 3.3, the call to RandomBinarySearch in line (9) successfully returns a monotone function $h\colon [n] \to \{0,1\}$ such that $\|f - h \circ \sigma_{i_R}\|_1 \leq \varepsilon$ with probability $1 - \delta$ by Lemma 3.1. By a union bound, the algorithm succeeds with probability $1 - 2\delta$. Moreover, since h is monotone, we have $h \circ \sigma_{i_R} \in \mathcal{H}(V, X)$ and so the algorithm is a proper learner. This completes the proof. □

Proof of Claim 3.3. Let $F = \{i \in [D]\colon d(f \circ \sigma_i^{-1}, \mathcal{M}) > \varepsilon\delta/10\}$ and observe that

$$\Pr[d(f \circ \sigma_{i_R}^{-1}, \mathcal{M}) > \varepsilon\delta/10] \leq \Pr[\exists i \in F\colon Z_i \geq Z_{i^\star}] \leq \sum_{i \in F} \Pr[Z_i - Z_{i^\star} \geq 0] \tag{1}$$

where $Z_i = \sum_j Z_{ij}$ is the random variable defined in lines (7-8) of the algorithm. Note that $Z_i - Z_{i^\star} = \sum_j Y_j$ where $Y_j := Z_{ij} - Z_{i^\star j}$ are iid random variables supported on $\{-2, 0, 2\}$. We now bound $\mathbb{E}[Y_j]$ and then bound $\Pr[Z_i - Z_{i^\star}]$ by standard concentration bounds and independence of the Y_j-s.

Let $\mathcal{E}_{ij}$ denote the event that $f_{S_j} \circ \sigma_i^{-1}$ is non-decreasing. Now, when $\mathcal{E}_{i^\star j}$ holds, the restricted labeling f_{S_j} satisfies the pre-condition for the exact learning algorithm ShuffledMonotoneLearner of Theorem 2.4 to successfully compute the labeling. I.e., $\mathcal{E}_{i^\star j}$ implies $f'_{S_j} = f_{S_j}$ in line (6). Note that when $\mathcal{E}_{i^\star j}$ does not hold, we have no guarantee about f'_{S_j}. Nonetheless, by a union bound, we have

$$\begin{aligned}\Pr[Y_j \neq -2] &\leq \Pr[Z_{ij} = 1] + \Pr[Z_{i^\star j} = -1] \\ &\leq \Pr[\neg\mathcal{E}_{i^\star j} \vee \mathcal{E}_{ij}] + \Pr[\neg\mathcal{E}_{i^\star j}] \leq 2\Pr[\neg\mathcal{E}_{i^\star j}] + \Pr[\mathcal{E}_{ij}]\end{aligned} \tag{2}$$

We now lower bound the probability of $\mathcal{E}_{i^\star j}$. Let $h\colon [n] \to \{0,1\}$ be a monotone function such that $\left\|f \circ \sigma_{i^\star}^{-1} - h\right\|_1 \leq \frac{\varepsilon\delta n}{400}$ and observe that $|\Delta(f, h)| = |\{x \in X\colon f(x) \neq h \circ \sigma_{i^\star}(x)| < \frac{\varepsilon\delta n}{400}$. Moreover, if $S_j \cap \Delta(f, h) = \emptyset$, then $\mathcal{E}_{i^\star j}$ holds. Thus,

$$\Pr_{S_j}[\mathcal{E}_{i^\star j}] \geq \Pr_{S_j}[S_j \cap \Delta(f, h) = \emptyset] \geq \left(1 - \frac{\varepsilon\delta}{400}\right)^{\frac{40}{\varepsilon\delta}} \geq e^{-1/10} > 9/10. \tag{3}$$

We now upper bound $\Pr[\mathcal{E}_{ij}]$ for $i \in F$. We will use the following lemma from the literature on property testing of monotone functions.

Lemma 3.4 (Lemma 16, [DGL+99])**.** *Let $g\colon [n] \to \{0,1\}$ such that $d(f, \mathcal{M}) \geq \varepsilon$. Then, choosing $s \geq 2$ points $j_1, \dots, j_s \in [n]$ iid and uniformly at random from $[n]$, we observe a violation of monotonicity for g with probability at least $(1 - e^{-\varepsilon s/2})^2$.*

Using Lemma 3.4 and the definition of F, we have

$$i \in F \implies \Pr[\mathcal{E}_{ij}] \le 1 - \left(1 - e^{-(\frac{\varepsilon\delta}{10}\cdot\frac{40}{\varepsilon\delta})/2}\right)^2 = e^{-2} < 1/7 \tag{4}$$

and plugging eq. (3) and eq. (4) into eq. (2) yields $\Pr[Y_j \ne -2] < \frac{1}{5} + \frac{1}{7} < \frac{2}{5}$, which implies

$$\mathbb{E}[Y_j] \le -2\Pr[Y_j = -2] + 2\Pr[Y_j \ne -2] < -2 \cdot \frac{3}{5} + 2 \cdot \frac{2}{5} = -\frac{2}{5} \tag{5}$$

Therefore, recalling that $Z_i - Z_{i^\star} = \sum_{j \le t} Y_j$ we have $\mathbb{E}[Z_i - Z_{i^\star}] < -(2/5) \cdot 50 \ln(D/\delta) = -20 \ln(D/\delta)$ and since the Y_j-s are independent, we have by Hoeffding's inequality that

$$\Pr[Z_i - Z_{i^\star} \ge 0] \le \Pr[|(Z_i - Z_{i^\star}) - \mathbb{E}[Z_i - Z_{i^\star}]| > 20\ln(D/\delta)] \le \exp\left(-\frac{2 \cdot 400 \ln^2(D/\delta)}{16 \cdot 50 \ln(D/\delta)}\right) = \frac{\delta}{D}$$

and plugging this bound into eq. (1) completes the proof. □

3.2 Realizable Learning

In this section we obtain a slightly improved algorithm for the realizable setting by adapting the exact learning algorithm for shuffled monotone functions in Theorem 2.4.

Theorem 3.5. *Let $X \subset \mathbb{R}^d$ be an arbitrary set of n points and $V \subset S^{d-1}$ an arbitrary set of D unit vectors. For any $\varepsilon, \delta > 0$, there is a randomized algorithm which, given query access to $f\colon X \to \{0,1\}$ such that $f \in \mathcal{H}(V, X)$, uses $O(\min(D + \log(\frac{1}{\varepsilon}), \frac{\ln(D/\delta)}{\delta\varepsilon}))$ queries and returns a halfspace $h \in \mathcal{H}(V, X)$ such that $\|f - h\|_1 \le \varepsilon$ with probability $1 - \delta$. In the case that $D + \log(\frac{1}{\varepsilon}) \le \frac{\ln D}{\delta\varepsilon}$, the algorithm is deterministic.*

Proof. First, there is clearly a $O(\frac{\ln(D/\delta)}{\delta\varepsilon})$ query algorithm simply by invoking the tolerant algorithm of Theorem 3.2. We show that in the realizable setting when $D + \log(1/\varepsilon)) \ll \frac{\ln(D/\delta)}{\delta\varepsilon}$, we can remove the $\ln(D/\delta)$ factor from the query complexity in Theorem 3.2. To achieve this, we describe how to slightly modify the exact learning algorithm of Theorem 2.4 to obtain a deterministic algorithm that correctly labels all but an ε-fraction of X using $O(D + \log(1/\varepsilon))$ queries.

Theorem 3.6. *For any D, n and $\varepsilon > 0$, there is a deterministic algorithm that ε-learns a shuffled monotone function using $O(D + \log(1/\varepsilon))$ label queries. Moreover, the returned labeling function is monotone.*

Again, since Theorem 3.6 returns a monotone hypothesis, we obtain a proper learner. □

Proof of Theorem 3.6. We consider a slight modification of the algorithm described in Alg. 1. Change line (5) so that the while-loop terminates when $n < \varepsilon n$ (instead of $n < 10$). That is, we run the algorithm of Theorem 2.4 until there are $|Z| \le \varepsilon n$ points remaining. This uses $O(D + \log(1/\varepsilon))$ queries and returns a set C of candidate hypotheses which by Claim 2.6 is guaranteed to contain the true labeling ℓ in the case that $i^\star \notin S$. Now, by Claim 2.2 we can use at most D queries and return a single representative $h^C \in C$ from this collection with the guarantee that if $\ell \in C$, then $h^C = \ell$. In summary, if $i^\star \notin S$, then we have $h^C = \ell$.

Next, we define at candidate hypothesis $h^Z\colon X \to \{0,1\}$ using the remaining points, Z. By item (3) of Lemma 2.5, whenever we remove points from Z, we in fact learn the true label of these points and so we can simply define $h^Z(x) = \ell(x)$ whenever a point is removed. Finally, set $h^Z(Z) = 0$. Now, note that if $i^\star \in S$, then by Claim 2.6, we have $x^\star, y^\star \in Z$ and so $h^Z(x) = \ell(x)$ for every $x \in X \setminus Z$, and thus $\|f - h^Z\| \le \varepsilon$. In summary, if $i^\star \in S$, then $\|f - h^Z\| \le \varepsilon$.

The only remaining task is to determine which of h^C or h^Z to return. If there exists a point $x \in X \setminus Z$ such that $h^C(x) \ne h^Z(x)$, then query $\ell(x)$ and return the hypothesis which agrees with ℓ at x. Otherwise, return h^C. If $i^\star \in S$, then $h^C = \ell$, as argued above. In this case we will always return h^C. Otherwise, as argued, we have $\|f - h^Z\| \le \varepsilon$. If $h^Z(X \setminus Z) = h^C(X \setminus Z)$, then we also have $\|f - h^C\| \le \varepsilon$ and this is what we return. Otherwise, we return h^Z. The returned hypothesis is ε-close to f in every case. Moreover, in both cases the returned function is monotone as claimed. □

3.3 Randomized Binary Search

Proof of Lemma 3.1. The algorithm performs a randomized binary search until there are less than $C\varepsilon n$ unknown elements remaining (we will set $C \approx 1/\delta$ later on in the proof). The algorithm is defined in Alg. 3.

Algorithm 3: Randomized Binary Search Learner

```
1  Input: $\varepsilon, \delta > 0$ and query access to $f\colon [n] \to \{0,1\}$ such that $\|f-h\|_1 \le \varepsilon$ for some monotone
     function $h\colon [n] \to \{0,1\}$;
2  Output: Monotone function $g\colon [n] \to \{0,1\}$ such that $\|f-g\|_1 \le 10\varepsilon/\delta$ with probability $1-\delta$;
3  Let $\mathcal{I} \leftarrow [n]$;
4  while $|\mathcal{I}| > C\varepsilon n$ do
5      Partition $\mathcal{I}$ into three subintervals of equal size $\mathcal{I} = \mathcal{I}_L \sqcup \mathcal{I}_M \sqcup \mathcal{I}_R$;
6      Choose $i$ uniformly at random from $\mathcal{I}_M$[7] and query $f(i)$;
7      If $f(i) = 0$, then set $g(\mathcal{I}_L) = 0$ for all $z \le i$ and set $\mathcal{I} \leftarrow \mathcal{I}_M \cup \mathcal{I}_R$;
8      If $f(i) = 1$, then set $g(R) = 1$ for all $z \ge i$ and set $\mathcal{I} \leftarrow \mathcal{I}_L \cup \mathcal{I}_M$;
9  end
10 For each $z \in \mathcal{I}$, set $g(z) = 0$;
```

Let $h \in \mathcal{M}$ be the monotone function minimizing $\|f-h\|_1$. Let $\Delta(f,h) = \{i \in [n] \colon f(i) \neq h(i)\}$ and note that $|\Delta(f,h)| \le \varepsilon n$. Observe that if every query of the above algorithm is from $[n] \setminus \Delta(f,h)$, then the output g is guaranteed to satisfy $\|h-g\|_1 \le C\varepsilon$ and therefore by the triangle inequality

$$\|f-g\|_1 \le \|f-h\|_1 + \|h-g\|_1 \le (C+1)\varepsilon.$$

Thus, it suffices to show that with high probability every query misses $\Delta(f,h)$.

We first bound the number of queries q made by the algorithm. Let $\mathcal{I}_t$ denote the interval $\mathcal{I}$ after the first $t-1$ queries. Note that by design of the algorithm, we always have $|\mathcal{I}_t| = (2/3)|\mathcal{I}_{t-1}|$ and so $|\mathcal{I}_t| = (2/3)^t n$. Therefore, the total number of queries is bounded by

$$q \le \log_{3/2} n - \log_{3/2}(C\varepsilon n) = \log_{3/2}(1/C\varepsilon).$$

Now, let B_t denote the bad event that the t-th query lands in $\Delta(f,h)$ and let B denote the union of these events for $t \le q$. Since we choose the t-th query in step (2b) from the middle third of the interval $\mathcal{I}_t$, we have $\Pr[B_t] \le \frac{\varepsilon n}{(1/3)|\mathcal{I}_t|}$. Then, using our bounds on $\Delta(f,h)$, q, $|\mathcal{I}_t|$, and a union bound we have

$$\Pr[B] \le \sum_{t \le q} \frac{3\varepsilon n}{(2/3)^t n} = 3\varepsilon \sum_{t \le q} (3/2)^t \le 3\varepsilon \cdot 3(3/2)^q \le 9/C$$

and setting $C = 9/\delta$ completes the proof. □

References

[AAB16] Hasan Abasi, Ali Z. Abdi, and Nader H. Bshouty. Learning boolean halfspaces with small weights from membership queries. *Theor. Comput. Sci.*, 2016. 3

[adH84] Friedhelm Meyer auf der Heide. A polynomial linear search algorithm for the n-dimensional knapsack problem. *J. ACM*, 31(3):668–676, 1984. 2

[BH19] Nader H. Bshouty and Catherine A. Haddad-Zaknoon. Adaptive exact learning of decision trees from membership queries. In *Algorithmic Learning Theory, ALT*, 2019. 3

[BHS23] Kiarash Banihashem, Mohammad Hajiaghayi, and Max Springer. Optimal sparse recovery with decision stumps. In Brian Williams, Yiling Chen, and Jennifer Neville, editors, *Conference on Artificial Intelligence, AAAI*, 2023. 3

[BLQT22] Guy Blanc, Jane Lange, Mingda Qiao, and Li-Yang Tan. Properly learning decision trees in almost polynomial time. *J. ACM*, 2022. 3

[BLT20] Guy Blanc, Jane Lange, and Li-Yang Tan. Top-down induction of decision trees: Rigorous guarantees and inherent limitations. In *Innovations in Theoretical Computer Science Conference, ITCS*, 2020. 3

[CIO16] Jean Cardinal, John Iacono, and Aurélien Ooms. Solving k-sum using few linear queries. In *European Symposium on Algorithms, ESA*, 2016. 2

[Das04] Sanjoy Dasgupta. Analysis of a greedy active learning strategy. In *Advances in Neural Information Processing Systems NIPS*, 2004. 2

[DGL$^+$99] Yevgeniy Dodis, Oded Goldreich, Eric Lehman, Sofya Raskhodnikova, Dana Ron, and Alex Samorodnitsky. Improved testing algorithms for monotonicity. In *RANDOM-APPROX*, 1999. 12

[EH89] Andrzej Ehrenfeucht and David Haussler. Learning decision trees from random examples. *Inf. Comput.*, 82(3):231–246, 1989. 3

[ES19] Esther Ezra and Micha Sharir. A nearly quadratic bound for point-location in hyperplane arrangements, in the linear decision tree model. *Discret. Comput. Geom.*, 61(4):735–755, 2019. 2

[HKLM20] Max Hopkins, Daniel Kane, Shachar Lovett, and Gaurav Mahajan. Point location and active learning: Learning halfspaces almost optimally. In *Foundations of Computer Science, FOCS*, 2020. 2

[Hol93] Robert C. Holte. Very simple classification rules perform well on most commonly used datasets. *Mach. Learn.*, 11:63–91, 1993. 3

[HY15] Steve Hanneke and Liu Yang. Minimax analysis of active learning. *J. Mach. Learn. Res.*, 16:3487–3602, 2015. 2, 3, 4, 16

[KLM18] Daniel M. Kane, Shachar Lovett, and Shay Moran. Generalized comparison trees for point-location problems. In *International Colloquium on Automata, Languages, and Programming, ICALP*, 2018. 2

[Mei93] Stefan Meiser. Point location in arrangements of hyperplanes. *Inf. Comput.*, 106(2):286–303, 1993. 2

[MORS09] Kevin Matulef, Ryan O'Donnell, Ronitt Rubinfeld, and Rocco A. Servedio. Testing ± 1-weight halfspace. In *APPROX-RANDOM*, 2009. 3

[MS25] Arya Mazumdar and Neha Sangwan. Exact recovery of sparse binary vectors from generalized linear measurements. *CoRR*, abs/2502.16008, 2025. 3

[OH94] Jonathan J. Oliver and David J. Hand. Averaging over decision stumps. In *Machine Learning: ECML-94, European Conference on Machine Learning*, volume 784 of *Lecture Notes in Computer Science*, pages 231–241. Springer, 1994. 3

[Yil15] Olcay Taner Yildiz. Vc-dimension of univariate decision trees. *IEEE Trans. Neural Networks Learn. Syst.*, 26(2):378–387, 2015. 16

A Lower Bound for Learning Decision Stumps

Theorem A.1 (Theorem 3 of [HY15]). *Fix $n, d \in \mathbb{N}$ and $\varepsilon \geq \frac{1}{2n}$. Any membership query algorithm for ε-realizable learning of decision stumps in $\mathbb{R}^d$ over n points must use in the worst case at least*

$$\Omega\left(\min\left(d + \log\frac{1}{\varepsilon}, \frac{1}{\varepsilon}\right) + \min\Big(\log d, \log n\Big)\right)$$

queries, and there is an algorithm using $O(\min(d \log(1/\varepsilon), \frac{\log d \log(1/\varepsilon)}{\varepsilon}, n)$ queries.

Proof. Note that it is without loss of generality that we require $\varepsilon \geq \frac{1}{2n}$. When $\varepsilon < \frac{1}{n}$, it is the case that ε-learning is equivalent to exact learning.

Let $\Lambda(\mathcal{H})$ be the query complexity of any active learning algorithm for a concept class $\mathcal{H}$. Theorem 3 of [HY15] lower bounds $\Lambda(\mathcal{H})$ in the ε-realizable setting with respect to three quantities: the star number $\mathfrak{s}(\mathcal{H})$, the VC dimension $\mathrm{VC}(\mathcal{H})$, and the cardinality $|\mathcal{H}|$,

$$\Lambda(\mathcal{H}) \gtrsim \max\left\{\min\left(\mathfrak{s}(\mathcal{H}), \frac{1}{\varepsilon}\right), \mathrm{VC}(\mathcal{H}), \log\left(\min\left(\frac{1}{\varepsilon}, |\mathcal{H}|\right)\right)\right\}.$$

The same theorem also provides the upper bound:

$$\Lambda(\mathcal{H}) \lesssim \min\left\{\mathfrak{s}(\mathcal{H}), \frac{\mathrm{VC}(\mathcal{H})}{\varepsilon}, \frac{\mathfrak{s}(\mathcal{H}) \cdot \mathrm{VC}(\mathcal{H})}{\log \mathfrak{s}(\mathcal{H})}\right\} \log\frac{1}{\varepsilon}.$$

In the following, we let $X = \{x_1, \ldots, x_n\}$ be a finite point set in $\mathbb{R}^d$, and let $\mathcal{H}$ contain the set of all possible distinct labelings of X by the class of decision stumps DS_d,

$$\mathcal{H} = \Big\{(f(x_1), \ldots, f(x_n) : f \in \mathrm{DS}_d\Big\} \subset \{0,1\}^X.$$

We can select X so that the following complexity measures of $\mathcal{H}$ are attained:

- The star number of decision stumps DS_d is $2d$, by Lemma A.3. This implies the upper bound $\mathfrak{s}(\mathcal{H}) \leq 2d$, and directly from the definition of the star number of DS_d, there is also a set X such that $\mathfrak{s}(\mathcal{H}) = \min\{2d, n\}$.
- The VC dimension of DS_d is $\Theta(\log d)$ by [Yil15]. This implies the upper bound $\mathrm{VC}(\mathcal{H}) \leq \mathrm{VC}(\mathrm{DS}_d)$. Letting X be a shattered set of DS_d, we obtain that $\mathrm{VC}(\mathcal{H}) = \min\{\log d, \log n\}$.
- Let X be any point set such that the first coordinate of each point is distinct. Then, $|\mathcal{H}| \geq n$.

By assumption, have $\min\left\{n, \frac{1}{\varepsilon}\right\} = \frac{1}{\varepsilon}$. This implies the following lower bound:

$$\begin{aligned}\Lambda(\mathcal{H}) &\geq \Omega\left(\min\left(d, \frac{1}{\varepsilon}\right) + \min\Big(\log d, \log n\Big) + \min\left(\log\frac{1}{\varepsilon}, \log n\right)\right) \\ &= \Omega\left(\min\left(d + \log\frac{1}{\varepsilon}, \frac{1}{\varepsilon}\right) + \min\Big(\log d, \log n\Big)\right).\end{aligned}$$

□

Definition A.2 (Star Number, [HY15]). *The star number $\mathfrak{s}$ of a function class $\mathcal{H}$ is the largest integer s such that there exist $x_1, \ldots, x_s \in X$ and $h_0, h_1, \ldots, h_s \in \mathcal{H}$ such that for each $i, j \in [s]$:*

$$h_0(x_j) \neq h_i(x_j) \iff i = j. \tag{6}$$

That is, the hypothesis h_i disagrees with h_0 only on the ith element of $\{x_1, \ldots, x_s\}$. If there is no maximum, then we say that the star number is infinite, $\mathfrak{s} = \infty$.

Lemma A.3 (Star Number of Decision Stumps)**.** *The star number of* DS_d *is* $2d$.

Proof. We first show that the star number of DS_d is at least $2d$. Let $e_1, \ldots, e_d$ be the standard basis vectors and let $\mathbf{1} = e_1 + \cdots + e_d$. Define the following set with $2d$ instances $\{x_{i,-}, x_{i,+} : i \in [d]\}$,

$$x_{i,-} = \left(-1, \ldots, -1, -\frac{1}{2}, -1, \ldots, -1\right) \quad \text{and} \quad x_{i,+} = \left(+1, \ldots, +1, +\frac{1}{2}, +1, \ldots, +1\right),$$

where $x_{i,-} = -\mathbf{1} + \frac{1}{2}e_i$ and $x_{i,+} = \mathbf{1} - \frac{1}{2}e_i$. Define the following collection of hypotheses:

$$h_{i,-} = f_{i,-3/4} \qquad \text{and} \qquad h_{i,+} = f_{i,+3/4},$$

in addition to the base hypothesis $h_0 = f_{1,0}$. By construction, each hypothesis assigns $x_{i,-}$ the label 0 except for $h_{x,-}$, which assigns it 1. The reverse is true for $x_{i,+}$, where all hypotheses except for $h_{i,+}$ assigns it the label 1. Thus, eq. (6) holds, so that $\mathfrak{s}(\mathrm{DS}_d) \geq 2d$.

To show an upper bound on the star number, let $\{x_1, \ldots, x_s\} \in \mathbb{R}^d$ and $h_0, h_1, \ldots, h_s \in \mathrm{DS}_d$ be any collection of instances and hypotheses satisfying eq. (6). Suppose that $s > 2d$. Then, by the pigeonhole principle, there are at least three hypotheses that threshold on the same coordinate. Without loss of generality, suppose that they are the first three hypotheses h_1, h_2, h_3, and that they also threshold on the first coordinate:

$$h_1 = f_{1,t_1} \qquad h_2 = f_{1,t_2} \qquad h_3 = f_{1,t_3},$$

where $t_1 < t_2 < t_3$. By assumption, we have that h_2 disagrees with both h_1 and h_3 on x_2, since:

$$h_2(x_2) \neq h_0(x_2) = h_1(x_2) = h_3(x_2).$$

However, h_1 and h_2 disagree only on points where the first coordinate is between $[t_1, t_2)$, while h_2 and h_3 disagree only on points where the first coordinate is between $[t_2, t_3)$. Therefore, there is no point x_2 that can satisfy eq. (6). This implies the upper bound $\mathfrak{s}(\mathrm{DS}_d) \leq 2d$. $\square$