

Monotone Classification with Relative Approximations

Yufei Tao

CUHK

taoyf@cse.cuhk.edu.hk

August 18, 2026

Abstract

In monotone classification, the input is a multiset P of points in $\mathbb{R}^d$, each associated with a hidden label from $\{-1, 1\}$. The goal is to identify a monotone classifier $h : \mathbb{R}^d \rightarrow \{-1, 1\}$ with a small *error*, measured as the number of points $p \in P$ whose labels differ from $h(p)$. The *cost* of an algorithm is the number of point labels it chooses to reveal. This article studies the minimum cost required to find a monotone classifier whose error is at most $(1 + \epsilon) \cdot k^*$ where $\epsilon \geq 0$, and k^* is the error of an optimal monotone classifier. In other words, the algorithm aims for a *relative* approximation, allowing the error to exceed the optimum by at most a factor of $1 + \epsilon$. We present nearly matching upper and lower bounds for the entire range of ϵ . Previous work could guarantee only an additive approximation to the optimal error.

Preliminary versions of this article appeared in PODS'18 and PODS'21.

1 Introduction

This article presents a systematic study of monotone classification, aiming to determine the minimum label-discovery overhead to guarantee a classification error that is higher than the optimum by at most a relative factor. We begin by defining the problem and explaining its practical motivations. We then present our findings and discuss their significance in relation to previous results. Finally, we provide an overview of our algorithmic techniques.

1.1 Problem Definitions

Math Conventions. For an integer $x \geq 1$, the notation $[x]$ represents the set $\{1, 2, \dots, x\}$. Given two non-negative real values x and y , we use the notation $x \leq_{1+\epsilon} y$ to represent the condition $x \leq (1 + \epsilon)y$ where $\epsilon \geq 0$. Given a real value $x \geq 1$, we use $\text{Log } x$ as a short form for $\log_2(1 + x)$. For any real value x , the expression $\exp(x)$ denotes e^x . Given a predicate Q , the notation $\mathbb{1}_Q$ equals 1 if Q holds or 0 otherwise.

Given a point $p \in \mathbb{R}^d$ for some dimensionality $d \geq 1$, the notation $p[i]$ represents the coordinate of p on dimension $i \in [d]$. A point $p \in \mathbb{R}^d$ is said to *dominate* a point $q \in \mathbb{R}^d$ if $p \neq q$ and the condition $p[i] \geq q[i]$ holds for all $i \in [d]$. We use $p \succ q$ to indicate “ p dominating q ” and $p \succeq q$ to indicate “ $p = q$ or $p \succ q$ ”. If $p \succeq q$, we may also write $q \preceq p$.

Monotone Classification. The input is a multiset P of n points in $\mathbb{R}^d$ for some integer $d \geq 1$. The multiset may contain repeated points, that is, distinct elements with identical coordinates. Each element $p \in P$ carries a label from $\{-1, 1\}$, which is represented as $\text{label}(p)$.

A *classifier* is a function $h : \mathbb{R}^d \rightarrow \{-1, 1\}$. We say that h *correctly classifies* an element $p \in P$ if $h(p) = \text{label}(p)$, or *misclassifies* p , otherwise. The *error* of h on P is defined as

$$\text{err}_P(h) = \sum_{p \in P} \mathbb{1}_{h(p) \neq \text{label}(p)} \quad (1)$$

that is, the number of elements in P misclassified by h . A classifier h is *monotone* if $h(p) \geq h(q)$ holds for any two points $p, q \in \mathbb{R}^d$ satisfying $p \succ q$.

Define

$$\mathbb{H}_{\text{mon}} = \text{the set of all monotone classifiers} \quad (2)$$

$$k^* = \min_{h \in \mathbb{H}_{\text{mon}}} \text{err}_P(h). \quad (3)$$

We call k^* the *optimal monotone error* of P . A classifier $h \in \mathbb{H}_{\text{mon}}$ is

- *optimal* on P if $\text{err}_P(h) = k^*$;
- *c-approximate* on P if $\text{err}_P(h) \leq c \cdot k^*$ where $c \geq 1$ is the *approximation ratio* of h .

Example 1.1. Figure 1 shows a 2D input P where a black (resp., white) point has label 1 (resp., -1). Consider the monotone classifier h that maps (i) all the black points to 1 except p_1 , and (ii) all the white points to -1 except p_{11} and p_{15} . Thus, $\text{err}_P(h) = 3$. No other monotone classifier has a smaller error on P , and hence $k^* = 3$. Consider the classifier h^{pos} that maps the entire P to 1. It can be verified that $\text{err}_P(h^{\text{pos}}) = 8$; hence, h^{pos} is $(8/3)$ -approximate on P .

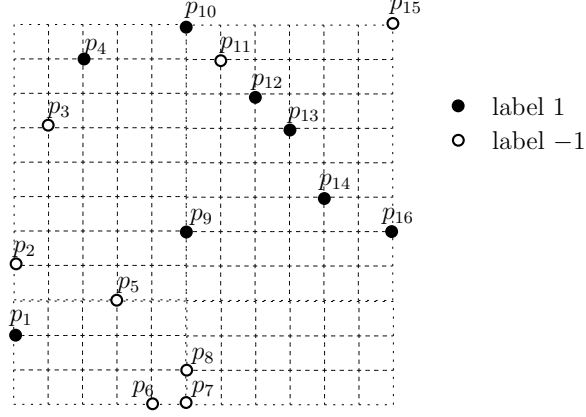


Figure 1: An input set P for Problem 1

The goal of an algorithm $\mathcal{A}$ is to find a classifier from $\mathbb{H}_{mon}$ whose error on P is small. In the beginning, the labels of the elements in P are hidden. There exists an *oracle* that $\mathcal{A}$ can query for labels. Specifically, in each *probe*, the algorithm selects an element $p \in P$ and receives $label(p)$ from the oracle. Its *cost* is the total number of elements probed.

In one extreme, by probing the whole P , $\mathcal{A}$ pays a cost of n , after which it can find an optimal monotone classifier on P with polynomial-time computation¹. In the other extreme, by probing nothing, $\mathcal{A}$ pays a cost of 0 but has to return a classifier based purely on the point coordinates; such a classifier could be very erroneous on P . The intellectual challenge is to understand the lowest cost to ensure the optimal monotone error k^* or an error higher than k^* by a small multiplicative factor.

Problem 1. (MONOTONE CLASSIFICATION WITH RELATIVE PRECISION GUARANTEES) *Given a real value $\epsilon \geq 0$, find a monotone classifier whose error on P is at most $(1 + \epsilon)k^*$. The efficiency of an algorithm is measured by the number of elements probed.*

If $k^* = 0$, we say that P is *monotone* and Problem 1 is *realizable*. Otherwise, P is *non-monotone* and the problem is *non-realizable*.

Every deterministic algorithm $\mathcal{A}_{det}$ for Problem 1 can be modeled as a binary decision tree $\mathcal{T}$ that is determined by the coordinates of the elements in P . Each internal node of $\mathcal{T}$ is associated with an element in P , while each leaf of $\mathcal{T}$ is associated with a monotone classifier. The execution of $\mathcal{A}_{det}$ descends a single root-to-leaf path in $\mathcal{T}$. When $\mathcal{A}_{det}$ is at an internal node, it probes the element $p \in P$ associated with that node and branches left if $label(p) = -1$ or right if $label(p) = 1$. When the algorithm reaches a leaf of $\mathcal{T}$, it outputs the monotone classifier associated with the leaf.

Randomized algorithms have access to an infinite bit string where each bit is independently 0 or 1 with probability 1/2. A randomized algorithm $\mathcal{A}_{ran}$ can be modeled as a function that maps the bit string to a deterministic algorithm (that is, when all the random bits are fixed, $\mathcal{A}_{ran}$ specializes into a deterministic algorithm). Suppose that h is the classifier output by $\mathcal{A}_{ran}$ when executed on P , and X is the number of probes performed by $\mathcal{A}_{ran}$ on P . Both X and h are random variables. The *expected cost* of $\mathcal{A}_{ran}$ is defined as $\mathbf{E}[X]$, while the *expected error* of $\mathcal{A}_{ran}$ is defined as $\mathbf{E}[err_P(h)]$. We say that $\mathcal{A}_{ran}$ guarantees an error k with *high probability* (w.h.p.) if $\Pr[err_P(h) \leq k] \geq 1 - 1/n^c$ where c can be an arbitrarily large constant set before running the algorithm.

¹Once all the point labels are available, an optimal monotone classifier can be found in time polynomial in n and d ; see [2, 47].

Although CPU time is not a main concern in this article, all proposed algorithms can be implemented to run in time polynomial in n and d , as will be duly noted in the technical development.

1.2 Practical Motivations

An important application of monotone classification is “entity matching”. Given two sets of entities E_1 and E_2 , the goal is to decide, for each pair $(e_1, e_2) \in E_1 \times E_2$, whether e_1 and e_2 represent the same entity; if so, they are said to form a “match”. For example, E_1 (resp., E_2) may be a set of advertisements placed on Amazon (resp., eBay). Each advertisement includes attributes like **prod-name**, **prod-description**, **year**, **price**, and so on. The goal is to identify the pairs $(e_1, e_2) \in E_1 \times E_2$ where the advertisements e_1 and e_2 describe the same product.

What makes the problem challenging is that decisions cannot rely on comparing attribute values, because even a pair of matching e_1 and e_2 may differ in attributes. This is evident with attributes like **prod-description** and **price** since e_1 and e_2 might describe or price the same product differently. In fact, e_1 and e_2 may not even agree on “presumably standard” attributes like **prod-name** (e.g., $e_1.\text{prod-name} = \text{“MS Word”}$ vs. $e_2.\text{prod-name} = \text{“Microsoft Word Processor”}$). Nevertheless, it would be reasonable to expect $e_1.\text{year} = e_2.\text{year}$ because advertisements are required to be accurate in this respect. To attain full precision in entity matching, human experts must manually inspect each pair $(e_1, e_2) \in E_1 \times E_2$, which is expensive due to the intensive labor involved. Therefore, it is crucial to develop an algorithm that can minimize human effort by automatically rendering verdicts on most pairs, even if it involves a small margin of error.

Toward the above purpose, a dominant methodology behind the existing approaches (e.g., [3, 14, 45, 6, 15, 20, 24, 35, 50, 39]) is to transform the task into a multidimensional classification problem with the following preprocessing.

1. First, shrink the set of all possible pairs to a subset $S \subseteq E_1 \times E_2$, by eliminating pairs that clearly are not matches. This is known as *blocking*, which is carried out based on application-dependent heuristics. This step is optional; if skipped, then $S = E_1 \times E_2$. In the Amazon-eBay example, S may involve only those advertisement pairs (e_1, e_2) with $e_1.\text{year} = e_2.\text{year}$.
2. For each entity pair $(e_1, e_2) \in S$, create a multidimensional point p_{e_1, e_2} using several — say d — similarity functions $\text{sim}_1, \text{sim}_2, \dots, \text{sim}_d$, each of which is evaluated on a certain attribute and produces a numeric “feature”. The i -th coordinate of p_{e_1, e_2} equals $\text{sim}_i(e_1, e_2)$: a greater value indicates higher similarity between e_1 and e_2 under the i -th feature. This creates a d -dimensional point set $P = \{p_{e_1, e_2} \mid (e_1, e_2) \in S\}$. In our example, from a numerical attribute such as **price**, one may extract a similarity feature $-|e_1.\text{price} - e_2.\text{price}|$, where the negation is needed to ensure “the larger the more similar”. From a text attribute (such as **prod-name** and **prod-description**) one may extract a similarity feature by evaluating the relevance between the corresponding texts of e_1 and e_2 using an appropriate metric (e.g., edit distance, jaccard-distance, cosine similarity, etc.). Multiple features may even be derived on the same attribute; e.g., one can extract two similarity features by computing the edit-distance and jaccard-distance of $e_1.\text{prod-name}$ and $e_2.\text{prod-name}$ separately.
3. Every point $p_{e_1, e_2} \in P$ inherently carries a label, which is 1 if (e_1, e_2) is a match, or -1 otherwise. The original entity matching task on E_1 and E_2 is now converted to inferring the labels of the points in P . Human inspection is the ultimate resort for determining the label of each p_{e_1, e_2} with guaranteed correctness.

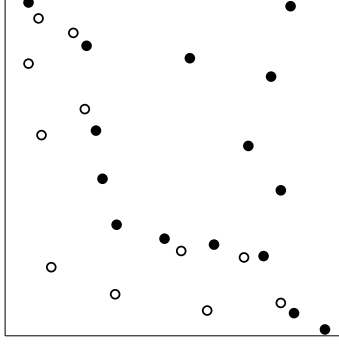


Figure 2: A monotone input set that cannot be separated by linear classifiers

Treating P as the input for monotone classification, we can employ an effective algorithm $\mathcal{A}$ designed for Problem 1 to significantly reduce human labor. Specifically, the human plays the role of oracle: when given a point p_{e_1, e_2} , the human “reveals” the label of p_{e_1, e_2} by manually checking whether e_1 and e_2 are about the same entity. After a number of probes (to the human oracle), the algorithm $\mathcal{A}$ outputs a monotone classifier $h \in \mathbb{H}_{mon}$, which is then used to infer the labels of the un-probed points in P . Such a classifier is also suitable for performing matching on entities received in the future (assuming that E_1 and E_2 are representative of the underlying data distributions). Demanding monotonicity is important for *explainable learning* because it avoids the odd situation of classifying (e_1, e_2) as a non-match but (e'_1, e'_2) as a match when $p_{e_1, e_2} \succ p_{e'_1, e'_2}$. Indeed, this oddity is difficult to explain because the former pair is at least as similar as the latter on every feature.

We remark that, in entity-matching applications such as the one described above, it would also be reasonable to consider a restrictive subset of monotone classifiers. For example, one may resort to *linear classifiers*, each of which is parameterized by a $(d + 1)$ -dimensional vector $\mathbf{w}$ and maps a point $p \in \mathbb{R}^d$ to label 1 if $\sum_{i=1}^d \mathbf{w}[i] \cdot p[i] \geq \mathbf{w}[d + 1]$ (note: $\mathbf{w}[i]$ denotes the i -th component of $\mathbf{w}$ for $i \in [d + 1]$) or label -1 if $\sum_{i=1}^d \mathbf{w}[i] \cdot p[i] < \mathbf{w}[d + 1]$. Equivalently, the classifier maps p to 1 if it falls on or on the “positive side” of a d -dimensional plane or to -1 if it falls on the plane’s “negative” side. Problem 1 can be re-defined with respect to the set of linear classifiers that obey monotonicity (as discussed earlier, monotonicity ensures explainable learning), by redefining k^* in (3) as the smallest error of all such classifiers. In this work, we allow *arbitrary* monotone classifiers to maximize the “classification power”. For instance, the set P in Figure 2 — where the black and white points have labels 1 and -1 , respectively — is monotone, whereas no linear classifier can separate the points of the two labels.

1.3 Related Work

Active Classification. Classification is a fundamental topic in machine learning. In the standard setting, we consider a (possibly infinite) set $\mathcal{P}$ of points in $\mathbb{R}^d$ and an unknown distribution $\mathcal{D}$ over $\mathcal{P} \times \{-1, 1\}$. Given a sample (p, l) drawn from $\mathcal{D}$, we refer to p and l as the sample’s *point* and *label*, respectively. A *classifier* is a function $h : \mathcal{P} \rightarrow \{-1, 1\}$, whose *error rate* with respect to $\mathcal{D}$ is calculated as

$$err\text{-}rate_{\mathcal{D}}(h) = \Pr_{(p, l) \sim \mathcal{D}}[h(p) \neq l].$$

Let $\mathbb{H}$ denote the set of classifiers under consideration, for which the optimal error rate is

$$\nu = \inf_{h \in \mathbb{H}} err\text{-}rate_{\mathcal{D}}(h). \quad (4)$$

The goal is to ensure the following *probabilistically approximately correct* (PAC) guarantee:

With probability at least $1 - \delta$, find a classifier $h \in \mathbb{H}$ such that $\text{err-rate}_{\mathcal{D}}(h) \leq \nu + \xi$ where ξ and δ are problem parameters satisfying $0 < \xi < 1$ and $0 < \delta < 1$.

An algorithm $\mathcal{A}$ is permitted to sample from $\mathcal{D}$ repeatedly. In *passive classification*, each sample (p, l) drawn from $\mathcal{D}$ reveals both p and l directly. The performance of $\mathcal{A}$ is measured by its *sample cost*, defined as the number of samples taken. In many practical applications, producing the point field of a sample requires negligible cost, but determining its label is expensive. This motivates *active classification*, which aims to achieve the PAC guarantee without acquiring the labels of all samples. In this setup, when an algorithm $\mathcal{A}$ draws a sample (p, l) from $\mathcal{D}$, only the point p is shown to $\mathcal{A}$, but the label l is hidden. The algorithm can request l from an *oracle*, thereby doing a *probe*. If $\mathcal{A}$ considers the knowledge of l unnecessary, it may skip the probe. The performance of $\mathcal{A}$ is now measured by its *label cost*, defined as the number of probes performed.

Problem 1 is relevant to *non-realizable active classification* where $\nu > 0$, i.e., even the best classifier in $\mathbb{H}$ has a positive, unknown error rate. This topic has been extensively studied; we refer the reader to [5, 7, 17, 18, 28, 29, 30, 31, 34, 46, 52] for representative works that serve as good entry points into the literature. By combining [5, 28, 29, 31], one can achieve the PAC guarantee with a label cost of

$$\tilde{O} \left(\min\{\theta, \mathfrak{s}, \chi\} \cdot \lambda \cdot \frac{\nu^2}{\xi^2} \right) \quad (5)$$

where $\lambda, \theta, \mathfrak{s}$, and χ are the *VC-dimension*, *disagreement coefficient*, *star number*, and *teaching dimension* of $\mathbb{H}$ on $\mathcal{P}$, respectively, and $\tilde{O}(\cdot)$ hides a polylogarithmic factor. The definitions of $\lambda, \theta, \mathfrak{s}$, and χ are given in Appendix B; for now, it suffices to understand that they are all at least 1. It is worth mentioning that the recent work of [30] described a method that can return a classifier h satisfying $\text{err-rate}_{\mathcal{D}}(h) \leq \nu + \xi$ with a label cost of $O(\frac{\nu^2}{\xi^2} \cdot (\lambda + \log \frac{1}{\delta})) + \tilde{O}(\min\{\mathfrak{s}, 1/\xi\} \cdot \lambda)$ except that the classifier h is not necessarily monotone.

In the context of Problem 1, assuming that the optimal monotone error k^* is known, we can utilize the above results to settle the problem by

- setting $\mathbb{H}$ to $\mathbb{H}_{\text{mon}}$;
- setting $\mathcal{P}$ to the input P of Problem 1;
- defining $\mathcal{D}$ as the uniform distribution over $\{(p, \text{label}(p)) \mid p \in P\}$.

Consequently, every monotone classifier h satisfies $\text{err-rate}_{\mathcal{D}}(h) = \text{err}_P(h)/n$ where $n = |P|$ and $\text{err}_P(h)$ is given in (1). In other words:

$$\nu = k^*/n. \quad (6)$$

Thus, by choosing

$$\xi = \epsilon \cdot k^*/n \quad (7)$$

and applying the result in (5), we can find a $(1 + \epsilon)$ -approximate monotone classifier w.h.p. with a cost bounded by

$$\tilde{O} \left(\min\{\theta, \mathfrak{s}, \chi\} \cdot \lambda \cdot \frac{1}{\epsilon^2} \right). \quad (8)$$

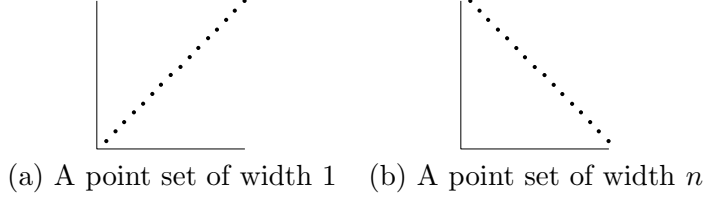


Figure 3: Illustration of the dominance width

When k^* is unknown, however, one can no longer set ξ “correctly” as in (7). While the algorithms of [5, 28, 29, 30, 31] require no knowledge of ν (and, hence, k^*) to run, they all demand a concrete ξ . One potential remedy would be to start with a large ξ (e.g., 0.5) and then iteratively reduce it until ξ reaches the value in (7). However, it is difficult (if not impossible) to apply such a remedy to the existing algorithms. A source of difficulty is that those algorithms work by bounding the *discrepancy* between an optimal classifier h^* and the returned classifier h , ensuring that the discrepancy can cause an “excess” error rate at most ξ . They cannot afford to infer the error rate caused by the points on which h^* and h assign the same labels — doing so would incur the same cost as passive learning. As a result, those algorithms do not have a reliable estimate about ν , making it difficult for them to decide whether ξ has come close to (7). In fact, to genuinely settle Problem 1 (with no assumptions), an active learning algorithm would need to return a classifier with an error rate at most $\nu(1 + \epsilon)$ for a given value of ϵ . We are not aware of any such algorithms.

Hypercube learning is a special case of monotone classification (i.e., the classification problem with $\mathbb{H} = \mathbb{H}_{mon}$) that has received particular attention. Here, $\mathcal{P} = \{0, 1\}^d$, every point $p \in \mathcal{P}$ carries a label $label(p)$, and $\mathcal{D}$ is the uniform distribution over $\{(p, label(p)) \mid p \in \mathcal{P}\}$. Improving over [9, 36], Lange and Vasilyan [37] achieved the PAC guarantee with a sample cost (hence, also a label cost) of $2^{\tilde{O}(\sqrt{d}/\xi)}$, where $\tilde{O}(\cdot)$ hides a logarithmic factor.

Monotonicity Testing. Given a multiset P of n labeled points as defined in Problem 1, *monotonicity testing* — a special instance of the general framework of *property testing* — is the problem of deciding whether P is monotone (i.e., having optimal monotone error $k^* = 0$) or far from being so. Formally, given an input parameter $\xi > 0$, the output must always be “yes” if P is monotone. If $k^* \geq \xi n$, the output should be “no” with probability at least $2/3$. In the scenario where $0 < k^* < \xi n$, the output can be either way. In the beginning, all the point labels are hidden. An algorithm $\mathcal{A}$ interacts with an oracle in the same manner as in Problem 1; its performance is measured by the number of probes carried out. Fischer et al. [23] gave an algorithm that solves the problem with $O(\sqrt{n}/\xi)$ probes. The problem has also been explored in other settings less relevant to this article; the interested readers may refer to [10, 11, 13, 26] and the references therein.

1.4 Our Results

For $\epsilon = 0$, Problem 1 aims to find an optimal monotone classifier. We prove that any algorithm achieving the purpose with a probability greater than $2/3$ needs to probe $\Omega(n)$ points in expectation (Theorem 10). The lower bound holds even when the dimensionality d is 1 and the optimal monotone error k^* is known to the algorithm.

The above hardness result justifies an investigation of Problem 1 under $\epsilon > 0$. In this regime, we show that the probing complexity is determined by the *width* w of the input P , formally defined as:

$$w = \text{the size of the largest } S \subseteq P \text{ such that } \nexists \text{ distinct } p, q \in S \text{ satisfying } p \succeq q. \quad (9)$$

ϵ	probing cost	refs	remarks
any	$\tilde{O}(w^2/\epsilon^2)$ if k^* known	[5, 28, 29, 31]	w.h.p. error
any	$\tilde{O}(\frac{w}{\epsilon^2} + \min\{w^2, \frac{n \cdot w}{\epsilon \cdot k^*}\})$ if k^* known	[30]	w.h.p. error
1	$O(w \log \frac{n}{w})$ expected	Thm. 1	expected error
any	$O(\frac{w}{\epsilon^2} \cdot \log \frac{n}{w} \cdot \log n)$	Thm. 6 + Sec. 1.5	w.h.p. error
0	$\Omega(n)$	Thm. 10	success probability $> 2/3$
constant	$\Omega(w \log \frac{n}{(k^*+1)w})$ expected	Thm. 13	expected error
any	$\Omega(w/\epsilon^2)$ expected	Thm. 14	expected error

Table 1: A summary of our and previous results on Problem 1

Any one-dimensional P has $w = 1$. Once the dimensionality d reaches 2, w can be anywhere between 1 and n ; see Figure 3 for two extreme examples.

Example 1.2. *Regarding the input P in Figure 1, the set $S = \{p_{10}, p_{11}, p_{12}, p_{16}, p_{13}, p_{14}\}$ satisfies the condition on the right hand side of (9). On the other hand, given any 7 points of P , we can always find two points where one point dominates the other. The dominance width w of P is 6.*

Equipped with w , we can expand our discussion on active learning in the context of Problem 1. As explained in Section 1.3, the existing algorithms are applicable only if k^* is known. While this is an unreasonable assumption, for the sake of a qualitative comparison, let us favor those algorithms by making k^* available for free, thus enabling them to achieve a label cost bounded by (8). In Appendix B, we prove that λ (the VC-dimension), θ (the disagreement coefficient), $\mathfrak{s}$ (the star number), and χ (the teaching dimension) can all be $\Omega(w)$. As a result, the best complexity from (8) is $\tilde{O}(w^2/\epsilon^2)$ where $\tilde{O}(\cdot)$ hides a polylogarithmic factor.

This article shows that the “true” complexity of Problem 1 is on the order of w/ϵ^2 up to polylogarithmic factors. We achieve the purpose by establishing nearly-matching upper and lower bounds. Regarding upper bounds:

- We design an algorithm that guarantees an expected error at most $2k^*$ with an expected cost of $O(w \log \frac{n}{w})$ (Theorem 1). When $k^* = 0$ — i.e., Problem 1 is realizable — the algorithm always finds an optimal classifier.
- For any $\epsilon > 0$, we design an algorithm that guarantees an error of $(1 + \epsilon)k^*$ w.h.p. at a cost of $O(\frac{w}{\epsilon^2} \cdot \log \frac{n}{w} \cdot \log n)$ (Theorem 6).

Regarding lower bounds:

- For any constant $c > 1$, we prove that any algorithm ensuring an expected error at most ck^* must probe $\Omega(w \log \frac{n}{(k^*+1)w})$ points in expectation (Theorem 13). Our first algorithm (with expected error at most $2k^*$) is thus asymptotically optimal when $k^* \leq (n/w)^{1-\delta}$ where $\delta > 0$ is an arbitrarily small constant.
- For any $\epsilon > 0$, we prove that any algorithm ensuring an expected error at most $(1 + \epsilon)k^*$ must probe $\Omega(w/\epsilon^2)$ points in expectation (Theorem 14). Hence, our second algorithm is asymptotically optimal up to a factor of $O(\log \frac{n}{w} \cdot \log n)$.

Table 1 summarizes all the results mentioned earlier.

As a side product, our findings imply a new result for monotonicity testing. Recall from Section 1.3 that the input to monotonicity testing includes a multiset P of n labeled points and

a real-valued parameter $\xi \in (0, 1)$. In Section 2.3, we solve the problem with an expected cost of $O(w \log \frac{n}{w}) + 2/\xi$, where w is the width of P . Recall that the state of the art on this problem probes $O(\sqrt{n}/\xi)$ points [25]. Our solution is expected to probe less points when w is small.

Preliminary versions of this work appeared in PODS'18 [48] and PODS'21 [49]. In terms of technical content, the current article differs from those papers in four ways: (i) we present a new, simpler analysis of the RPE algorithm (see Section 1.5); (ii) we establish a new coresnet result (Theorem 6); (iii) we prove a lower bound for Problem 1 under arbitrary ϵ values; (iv) we strengthen the relevance to active learning, including the derivation of lower bounds for λ , θ , $\mathfrak{s}$, and χ in the setting of Problem 1.

1.5 Overview of Our Algorithms

This subsection highlights the algorithmic techniques developed in this work.

A Randomized Algorithm. Our first algorithm — named RPE (random probes with elimination) — is rather simple:

algorithm RPE (P)

1. $Z = \emptyset$
2. **while** $P \neq \emptyset$
3. probe an element $z \in P$ chosen uniformly at random and add z to Z
4. **if** $\text{label}(z) = 1$ **then** remove from P every $p \in P$ satisfying $p \succeq z$
5. **else** remove from P every $p \in P$ satisfying $z \succeq p$
- /* note: z is removed by Line 4 or 5 */
6. **return** Z

The algorithm outputs the set Z of points probed. Given Z , we define the following classifier:

$$h_{\text{RPE}}(p) = \begin{cases} 1 & \text{if } \exists z \in Z \text{ such that } \text{label}(z) = 1 \text{ and } p \succeq z \\ -1 & \text{otherwise} \end{cases} \quad (10)$$

The classifier must be monotone. Otherwise, there exist $p, q \in \mathbb{R}^d$ such that $p \succ q$ but $h_{\text{RPE}}(p) = -1$ and $h_{\text{RPE}}(q) = 1$. Let z be an arbitrary element in Z satisfying $\text{label}(z) = 1$ and $q \succeq z$ (such z must exist by the definition in (10)). It follows that $p \succ z$, in which case $h_{\text{RPE}}(p)$ should be 1 according to (10), giving a contradiction.

RPE can also be understood as the following labeling process. After learning the label of a random element $z \in P$, we decide to map z to $\text{label}(z)$. If $\text{label}(z) = 1$, by monotonicity, we are obliged to map all elements $p \in P$ satisfying $p \succeq z$ to 1; conversely, if $\text{label}(z) = -1$, we must map all elements $p \in P$ satisfying $z \succeq p$ to -1 . The elements that have their labels thus mapped are removed from P . The process is then repeated on the remaining elements.

Example 1.3. Let us illustrate RPE on the input P in Figure 1. Assume that the algorithm randomly probes p_1 first. Acquiring $\text{label}(p_1) = 1$, it eliminates the entire P except p_6, p_7 , and p_8 . Suppose that RPE then randomly probes p_8 . Acquiring $\text{label}(p_8) = -1$, it removes all the remaining points in P . With $Z = \{p_1, p_8\}$, the classifier h_{RPE} in (10) maps all the points to label 1 except p_6, p_7 , and p_8 . Its error on P is $\text{err}_P(h_{\text{RPE}}) = 5$.

It may be unclear why the classifier in (10) ensures an expected error at most $2k^*$ — after all, probing a “wrong” element of P may immediately force h_{RPE} to misclassify a large number of elements (e.g., imagine what if p_{15} is probed first in Figure 1). Furthermore, it may be unclear why

RPE probes $O(w \log \frac{n}{w})$ elements in expectation, particularly since the algorithm never computes the value of w . We answer these questions in Section 2.

Relative-Comparison Coresets. The approximation ratio 2 is tight for RPE (this will be explained in Section 2.1). To achieve an approximation ratio of $1 + \epsilon$ for any $\epsilon > 0$, we aim to produce a function $F : \mathbb{H}_{mon} \rightarrow \mathbb{R}$ having the property below.

The relative ϵ -comparison property: $F(h) \leq F(h')$ implies $err_P(h) \leq_{1+\epsilon} err_P(h')$ for any classifiers $h, h' \in \mathbb{H}_{mon}$.

Given F , we can identify a monotone classifier h^* with the lowest $F(h^*)$. This classifier satisfies $err_P(h^*) \leq (1 + \epsilon)k^*$ (noticing that $F(h^*) \leq F(h^*)$ where h^* is an optimal classifier on P , i.e., $err_P(h^*) = k^*$) and thus can be returned as an output of Problem 1.

A barrier in finding such a function F is that we cannot hope to accurately estimate the $err_P(h)$ of every monotone classifier h . This is true even if the dimensionality is 1. To see why, consider the classifier h^{pos} that maps everything to 1. Its error is 0 if all elements of P have label 1, or error 1 if all but one element in P has label 1. Hence, estimating $err_P(h^{pos})$ up to a relative ratio, say, $1/2$ requires identifying the only element in P having label -1 or declaring the absence of such an element. It is not hard to prove that achieving the purpose with a constant probability demands $\Omega(n)$ probes in expectation.

Instead, we produce F by ensuring the following inequality on every $h \in \mathbb{H}_{mon}$:

$$err_P(h) \cdot \left(1 - \frac{\epsilon}{4}\right) + \Delta \leq F(h) \leq err_P(h) \cdot \left(1 + \frac{\epsilon}{4}\right) + \Delta \quad (11)$$

where Δ is an unknown real value common to all h . If the exact value of Δ *were* available, then $F(h) - \Delta$ *would* serve as an estimate of $err_P(h)$ with a relative ratio at most $\epsilon/4$, which would require $\Omega(n)$ probes as discussed. The key behind our solution is to compute $F(h)$ *without* knowing Δ , as long as the *existence* of Δ can be assured.

The relative ϵ -comparison property is a corollary of (11) because

$$\begin{aligned} err_P(h) &\leq \frac{1}{1 - \epsilon/4} \cdot (F(h) - \Delta) \\ (\text{by } F(h) \leq F(h')) &\leq \frac{1}{1 - \epsilon/4} \cdot (F(h') - \Delta) \\ (\text{applying (11)}) &\leq \frac{1 + \epsilon/4}{1 - \epsilon/4} \cdot err_P(h') \\ &\leq (1 + \epsilon) \cdot err_P(h') \end{aligned}$$

where the last inequality used the fact that $\frac{1+\epsilon/4}{1-\epsilon/4} \leq 1 + \epsilon$ for all $\epsilon \in (0, 1]$. The existence of Δ already permits the derivation to proceed — its concrete value is unnecessary.

We derive a function F meeting the condition in (11) from a *coreset* of P , which is a subset $Z \subseteq P$ where each element $p \in Z$

- has its label revealed, and
- is associated with a positive real value $weight(p)$, called the *weight* of p .

Given a classifier $h \in \mathbb{H}_{\text{mon}}$, define its *weighted error* on Z as:

$$w\text{-err}_Z(h) = \sum_{p \in Z} \text{weight}(p) \cdot \mathbb{1}_{h(p) \neq \text{label}(p)}. \quad (12)$$

With $O(\frac{w}{\epsilon^2} \text{Log } \frac{n}{w} \cdot \log n)$ probes, we can obtain w.h.p. a coreset Z of size $O(\frac{w}{\epsilon^2} \text{Log } \frac{n}{w} \cdot \log n)$, such that every $h \in \mathbb{H}_{\text{mon}}$ satisfies

$$\text{err}_P(h) \cdot \left(1 - \frac{\epsilon}{4}\right) + \Delta \leq w\text{-err}_Z(h) \leq \text{err}_P(h) \cdot \left(1 + \frac{\epsilon}{4}\right) + \Delta \quad (13)$$

for some unknown Δ . The function $w\text{-err}_Z$ serves as the desired function F . We refer to Z as a *relative-comparison coreset* because its purpose is to enable the relative ϵ -comparison property.

There is a large body of work on coresets — see [1, 4, 8, 12, 16, 21, 22, 32, 33, 43] and the references therein — which generally refers to a small set of elements that approximately preserves a specified property of the original input set. Our target of preservation — namely, the inequalities in (11) — is closely related to the coreset formulation in [22] (see in particular Definition 13 therein), which likewise uses an additive Δ to support relative comparisons, albeit in a different setting. As will be clear in Section 3, a key challenge in our analysis is to argue for the *existence* of a working Δ in (11) without explicitly computing it — as noted before, deriving Δ precisely demands $\Omega(n)$ probes. The idea of using an “unknown parameter” such as Δ for coreset construction is not new; e.g., it is implicit in the sampling technique of [40] for clustering polygonal curves. Our contribution is to make the idea work in the context of finding a good monotone classifier.

We close the section with a remark on the relevance of our work to coresets in machine learning that are designed for approximate regression within a relative factor (see [1, 38, 42, 44, 51] and the references therein). As discussed in Section 1.2, Problem 1 can be re-defined as a problem over any restrictive subset of monotone classifiers, e.g., linear classifiers that obey monotonicity. Because, as explained before, the function F we construct guarantees the relative ϵ -comparison property for all monotone classifiers, it (trivially) offers such a guarantee for all linear monotone classifiers. Therefore, our method also solves Problem 1 with respect to linear monotone classifiers at the same probing cost (more generally, the same holds for any subset of monotone classifiers). Although linear classifiers are well-studied, we are not aware of any coreset result that can match ours for Problem 1. Perhaps the most related result in this regard is the recent work of [1], which proves the existence of small coresets for linear classifiers under logistic regression. Recall that the imprecision measure in Problem 1 (see (1)) is the standard “0-1 error”, for which the logistic loss (i.e., the loss function to be minimized in logistic regression) is a widely used surrogate in practice. In theory, however, a relative- ϵ approximation of logistic loss does not imply a relative- ϵ approximation of the 0-1 error. Thus, the guarantee of [1] does not imply the guarantee required by Problem 1, even for linear monotone classifiers. Similar observations apply to coresets designed to approximate other surrogate objectives.

2 Random Probes with Elimination

This section proves the following theorem on the RPE algorithm in Section 1.5:

Theorem 1. *For Problem 1, RPE probes $O(w \text{Log } \frac{n}{w})$ elements in expectation, and the classifier h_{RPE} in (10) has an expected error at most $2k^*$, where n is the size of the input P , w is its width, and k^* is its optimal monotone error.*

Our proof is divided into two parts: Section 2.1 analyzes the expected error, and Section 2.2 examines the expected cost. The following proposition states a useful property of the classifier h_{RPE} , establishing symmetry between labels -1 and 1 .

Proposition 1. *For any $p \in P$, $h_{\text{RPE}}(p) = -1$ if and only if $\exists z \in Z$ satisfying $\text{label}(z) = -1$ and $z \succeq p$, where Z is the set of elements probed by RPE.*

Proof. We first show that Z is monotone, namely:

- Z contains no two distinct elements p and q with $p = q$;
- for any $p, q \in Z$: $p \succ q \Rightarrow \text{label}(p) \geq \text{label}(q)$.

The first bullet holds because once an element $z \in P$ is probed at Line 3 of the pseudocode in Section 1.5, all the elements of P at the same location as z are removed from P at Line 4 or 5. Regarding the second bullet, assume that there exist $p, q \in Z$ such that $p \succ q$, $\text{label}(p) = -1$, and $\text{label}(q) = 1$. Which element was probed earlier by RPE? If it was p , then q should have been removed from P at Line 5 after the probing of p at Line 3, contradicting $q \in Z$. Likewise, probing q first would contradict $p \in Z$.

Next, we prove that, for any $p \in P$, the condition $h_{\text{RPE}}(p) = -1$ holds if and only if $\exists z \in Z$ satisfying $\text{label}(z) = -1$ and $z \succeq p$. Let us first consider the “only-if direction” (i.e., $\Rightarrow$). Our argument proceeds differently in two cases.

- Case $p \in Z$: This implies $\text{label}(p) = -1$ (indeed, if $\text{label}(p) = 1$, then $h_{\text{RPE}}(p) = 1$ because we can set $z = p$ in (10)). We can set $z = p$ to fulfill the only-if statement.
- Case $p \notin Z$: From $h_{\text{RPE}}(p) = -1$, we can assert by (10) that no element $z' \in Z$ satisfies $\text{label}(z') = 1$ and $p \succeq z'$. Thus, RPE must have removed p after probing an element z satisfying $\text{label}(z) = -1$ and $z \succeq p$. This z fulfills the only-if statement.

Finally, we consider the “if direction” (i.e., $\Leftarrow$). The designated element z rules out the existence of any element $z' \in Z$ satisfying $\text{label}(z') = 1$ and $p \succeq z'$; otherwise, $z \succeq z'$ and the labels of z and z' suggest that Z is not monotone. Hence, $h_{\text{RPE}}(p) = -1$ by (10). $\square$

2.1 The Expected Error of RPE

Fix an arbitrary classifier $h \in \mathbb{H}_{\text{mon}}$. An element $p \in P$ is said to be *h-good* if $h(p) = \text{label}(p)$ or *h-bad* otherwise. We prove in this subsection:

Lemma 2. *The number of h-good elements misclassified by the classifier h_{RPE} in (10) is at most $\text{err}_P(h)$ in expectation.*

The lemma implies $\mathbf{E}[\text{err}_P(h_{\text{RPE}})] \leq 2k^*$. To understand why, set h to an optimal monotone classifier h^* on P . By Lemma 2, the number of h^* -good elements misclassified by h_{RPE} is at most $\text{err}_P(h^*) = k^*$ in expectation. Because exactly k^* elements of P are h^* -bad, the total number of elements misclassified by h_{RPE} is at most $2k^*$ in expectation.

Our proof of Lemma 2 works by induction on the size n of P . When $n = 1$, the classifier h_{RPE} has error 0 on P , and the claim holds. Next, assuming that the claim holds for $n \leq m - 1$ (where $m \geq 2$), we establish its correctness for $n = m$. Define

$$X = \text{the number of } h\text{-good elements in } P \text{ misclassified by } h_{\text{RPE}}, \quad (14)$$

Our goal is to show that $\mathbf{E}[X] \leq \text{err}_P(h)$.

For each h -bad element p , we define its *influence set* $I_{\text{bad}}(p)$ as follows:

- If $label(p) = -1$, then $I_{bad}(p)$ consists of all the h -good elements $q \in P$ satisfying $p \succeq q$ and $label(q) = 1$;
- If $label(p) = 1$, then $I_{bad}(p)$ consists of all the h -good elements $q \in P$ satisfying $q \succeq p$ and $label(q) = -1$.

Intuitively, $I_{bad}(p)$ includes all the h -good elements q of P such that once RPE probes p , it will mis-classify q . For each h -good element $q \in P$, define its *influence set* $I_{good}(q)$ as follows:

- If $label(q) = -1$, then $I_{good}(q)$ consists of all the h -bad elements $p \in P$ satisfying $q \succeq p$.
- If $label(q) = 1$, then $I_{good}(q)$ consists of all the h -bad elements $p \in P$ satisfying $p \succeq q$.

Intuitively, $I_{good}(q)$ includes all the h -bad elements p of P such that once RPE probes q , it will mis-classify p .

Lemma 3. *For any h -bad $p \in P$ and any h -good $q \in P$, we have $q \in I_{bad}(p) \Leftrightarrow p \in I_{good}(q)$.*

Proof. First, we prove $q \in I_{bad}(p) \Rightarrow p \in I_{good}(q)$. By symmetry, we discuss only the case $label(p) = -1$. Because $q \in I_{bad}(p)$, the definition of $I_{bad}(p)$ tells us $p \succeq q$ and $label(q) = 1$. It thus follows that $p \in I_{good}(q)$.

Next, we prove $p \in I_{good}(q) \Rightarrow q \in I_{bad}(p)$. By symmetry, we discuss only the case $label(q) = -1$. Because $p \in I_{good}(q)$, the definition of $I_{good}(q)$ tells us $q \succeq p$. Because q is h -good, we know $h(q) = label(q) = -1$, which by monotonicity implies $h(p) = -1$. As p is h -bad, we have $label(p) = 1$. It thus follows that $q \in I_{bad}(p)$. $\square$

Lemma 3 indicates

$$\sum_{h\text{-good } q \in P} |I_{good}(q)| = \sum_{h\text{-bad } p \in P} |I_{bad}(p)|. \quad (15)$$

Example 2.1. *Let P be the set of points in Figure 1, and h be the classifier that maps all the black points to 1 except p_1 and all the white points to -1 except p_{11} and p_{15} . Thus, p_1 , p_{11} , and p_{15} are h -bad, while the other points of P are h -good. The following are some representative influence sets.*

$$\begin{aligned} I_{bad}(p_{15}) &= \{p_4, p_9, p_{10}, p_{12}, p_{13}, p_{14}, p_{16}\} \\ I_{bad}(p_1) &= \{p_2, p_3, p_5\} \\ I_{good}(p_3) &= \{p_1\} \\ I_{good}(p_9) &= \{p_{11}, p_{15}\}. \end{aligned}$$

RPE is an iterative algorithm. We refer to Lines 3-5 of its pseudocode as an *iteration*. Let z be the point probed in the first iteration (at Line 3). The revelation of $label(z)$ instructs the algorithm to remove z and possibly some other elements from P (at Line 4 or 5). Define

$$P_z = \text{the set of remaining elements at the end of the first iteration.} \quad (16)$$

The next proposition relates the error of h on P_z to the error of h on P .

Proposition 2.

$$err_{P_z}(h) \leq \begin{cases} err_P(h) - 1 & \text{if } z \text{ is } h\text{-bad} \\ err_P(h) - |I_{good}(z)| & \text{if } z \text{ is } h\text{-good} \end{cases}$$

Proof. If z is h -bad, the inequality $\text{err}_{P_z}(h) \leq \text{err}_P(h) - 1$ follows immediately from the fact that P_z has lost at least one h -bad element (i.e., z).

Consider that z is h -good. Assume $\text{label}(z) = 1$. In the first iteration, an element $p \in P$ is removed by Line 4 if and only if $p \succeq z$. By definition of $I_{\text{good}}(z)$, an element p removed by Line 4 is h -bad if and only if $p \in I_{\text{good}}(z)$. Hence, P_z loses exactly $|I_{\text{good}}(z)|$ h -bad elements compared to P , giving $\text{err}_{P_z}(h) = \text{err}_P(h) - |I_{\text{good}}(z)|$. A symmetric argument applies to the case where $\text{label}(z) = -1$. $\square$

Define

$$Y_z = \text{the number of } h\text{-good elements in } P_z \text{ misclassified by } h_{\text{RPE}}$$

Under the event that the first element probed is z , we have

$$X = Y_z + \text{the number of } h\text{-good elements in } P \setminus P_z \text{ misclassified by } h_{\text{RPE}} \quad (17)$$

where X is defined in (14).

Proposition 3. *The number of h -good elements in $P \setminus P_z$ misclassified by h_{RPE} is*

- $|I_{\text{bad}}(z)|$ if z is h -bad;
- 0 if z is h -good.

Proof. Consider the case $\text{label}(z) = -1$. The set $P \setminus P_z$ consists of every element $p \in P$ satisfying $z \succeq p$. By Proposition 1, the classifier h_{RPE} maps every such an element to -1 . Hence, the number of h -good points in $P \setminus P_z$ misclassified by h_{RPE} equals the number — let it be x — of h -good points in $P \setminus P_z$ whose labels are 1. If z is h -bad, then x is exactly $|I_{\text{bad}}(z)|$ by definition of $I_{\text{bad}}(z)$. Suppose that z is h -good. By monotonicity of h , we have $h(p) = -1$ for all $p \succeq z$ (recall that $\text{label}(z) = -1$). If p is h -good, then $\text{label}(p) = h(p) = -1$. The number x is thus 0 if z is h -good.

A symmetric argument applies to the case $\text{label}(z) = 1$. $\square$

Combining (17) and Proposition 3 yields

$$X = \begin{cases} Y_z + |I_{\text{bad}}(z)| & \text{if } z \text{ is } h\text{-bad} \\ Y_z & \text{if } z \text{ is } h\text{-good} \end{cases} \quad (18)$$

The subsequent execution of **RPE** on P_z can be regarded as invoking **RPE** directly on P_z . As P_z has at least one less element than P (because z is removed), the inductive assumption tells us:

$$\mathbf{E}[Y_z] \leq \text{err}_{P_z}(h). \quad (19)$$

We can now derive

$$\begin{aligned} \mathbf{E}[X] &= \sum_{z \in P} \mathbf{E}[X \mid z \text{ is probed first}] \cdot \Pr[z \text{ is probed first}] \\ &= \frac{1}{m} \sum_{z \in P} \mathbf{E}[X \mid z \text{ is probed first}] \\ &= \frac{1}{m} \left(\sum_{h\text{-good } z \in P} \mathbf{E}[X \mid z \text{ is probed first}] + \sum_{h\text{-bad } z \in P} \mathbf{E}[X \mid z \text{ is probed first}] \right) \end{aligned}$$

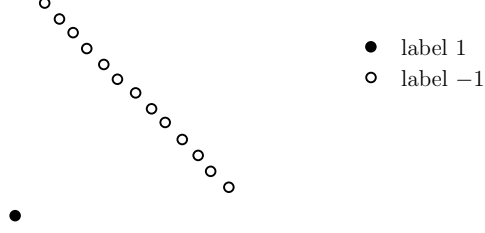


Figure 4: The approximation ratio 2 is tight for RPE

$$\begin{aligned}
(\text{by (18)}) &= \left(\frac{1}{m} \sum_{h\text{-good } z \in P} \mathbf{E}[Y_z] \right) + \left(\frac{1}{m} \sum_{h\text{-bad } z \in P} \mathbf{E}[Y_z] + |I_{\text{bad}}(z)| \right) \\
(\text{by (19)}) &\leq \left(\frac{1}{m} \sum_{h\text{-good } z \in P} \text{err}_{P_z}(h) \right) + \left(\frac{1}{m} \sum_{h\text{-bad } z \in P} \text{err}_{P_z}(h) + |I_{\text{bad}}(z)| \right) \\
(\text{Proposition 2}) &\leq \left(\frac{1}{m} \sum_{h\text{-good } z \in P} \text{err}_P(h) - |I_{\text{good}}(z)| \right) + \left(\frac{1}{m} \sum_{h\text{-bad } z \in P} \text{err}_P(h) - 1 + |I_{\text{bad}}(z)| \right) \\
&\leq \left(\frac{1}{m} \sum_{h\text{-good } z \in P} \text{err}_P(h) - |I_{\text{good}}(z)| \right) + \left(\frac{1}{m} \sum_{h\text{-bad } z \in P} \text{err}_P(h) + |I_{\text{bad}}(z)| \right) \\
&= \text{err}_P(h) + \frac{1}{m} \left(\sum_{h\text{-bad } z \in P} |I_{\text{bad}}(z)| - \sum_{h\text{-good } z \in P} |I_{\text{good}}(z)| \right) \\
(\text{by (15)}) &= \text{err}_P(h).
\end{aligned}$$

This completes the inductive step of the proof of Lemma 2.

Remark. The approximation ratio 2 is the best possible for RPE. To see this, consider the input P in Figure 4, where $n - 1$ white points have label -1 and the only black point has label 1 . The optimal monotone error k^* is 1 (achieved by the monotone classifier that maps all points to -1). If RPE probes the black point first — which happens with probability $1/n$ — then h_{RPE} misclassifies all the white points and, thus, incurs an error of $n - 1$. On the other hand, if the first point probed is white, then h_{RPE} misclassifies only the black point and incurs an error of 1. The expected error of h_{RPE} is therefore $\frac{1}{n} \cdot (n - 1) + (1 - \frac{1}{n}) \cdot 1 = 2 - \frac{2}{n}$, which gets arbitrarily close to 2 as n increases.

2.2 The Expected Cost of RPE

Chains. Let us first review a property of the width w defined in (9). Call a subset $S \subseteq P$ a *chain* if we can arrange the elements of S into a sequence $p_1 \preceq p_2 \preceq \dots \preceq p_{|S|}$. A *chain decomposition* of P is a collection of disjoint chains $C_1, C_2, \dots, C_t$ ($t \geq 1$) whose union is P . Dilworth's Theorem [19] states that there must exist a chain decomposition of P containing w chains.

Example 2.2. The input P in Figure 1 can be decomposed into 6 chains: $C_1 = \{p_1, p_2, p_3, p_4, p_{10}\}$, $C_2 = \{p_{11}\}$, $C_3 = \{p_5, p_9, p_{12}\}$, $C_4 = \{p_{16}\}$, $C_5 = \{p_{13}\}$, and $C_6 = \{p_6, p_7, p_8, p_{14}, p_{15}\}$. The points $p_{10}, p_{11}, p_{12}, p_{16}, p_{13}$, and p_{14} indicate the absence of any chain decomposition of P having less than 6 chains.

Attrition and Elimination. Before discussing the cost of RPE, let us take a detour to discuss an *attrition-and-elimination* (A&E) game between Alice and Bob. The input is a chain C of $m \geq 1$ points in $\mathbb{R}^d$. In each round:

- Bob performs “attrition” by either doing nothing or arbitrarily deleting some points from C ;
- Alice then carries out “elimination” by picking a point $p \in C$ uniformly at random, and deleting from C all the points $q \succeq p$.

The game ends when C becomes empty. The number of rounds is a random variable depending on Bob’s strategy. The question is how Bob should play to maximize the expectation of this variable. The game is a generalization of “random binary search”, in which Bob always chooses to do nothing in each round.

Lemma 4. *Regardless of Bob’s strategy, the game has $O(\log m)$ rounds in expectation.*

Proof. Let X be the number of elements left in C after the first round. We show $\Pr[X \leq m/2] > 1/2$. Let C' be the content of C after Bob’s attrition in the first round. Set $m' = |C'|$. Let us arrange the points of C' such that $p_1 \preceq p_2 \preceq \dots \preceq p_{m'}$. If Alice picks $p = p_i$ ($i \in [m']$), then at most $i - 1$ points are left after her elimination. Hence, $X \leq m'/2$ as long as $i \leq 1 + m'/2$, which occurs with probability greater than $1/2$. The fact $\Pr[X \leq m/2] > 1/2$ now follows from $m' \leq m$.

We call a round *successful* if it reduces the number of elements in C by at least a factor of 2. The total number of successful rounds cannot be more than $\log_2(1 + m)$. Let Y be the total number of rounds. If $Y \geq 2\log_2(1 + m)$, there must be at least $\log_2(1 + m)$ unsuccessful rounds; as all the rounds are independent, this can happen with probability less than $(1/2)^{\log_2(1+m)} < 1/m$. Because Y is trivially bounded by m , we have $\mathbf{E}[Y] \leq 2\log_2(1 + m) + m \cdot \Pr[\text{at least } 2\log_2(1 + m) \text{ rounds}] = O(\log m)$. $\square$

Cost Analysis of RPE. Returning to RPE, let $\{C_1, C_2, \dots, C_w\}$ be an arbitrary chain decomposition of P with w chains. Note that RPE is unaware of these chains. For each $i \in [w]$, break the chain C_i into two disjoint subsets:

- $C_i^{\text{pos}} = \{p \in C_i \mid \text{label}(p) = 1\}$, and
- $C_i^{\text{neg}} = \{p \in C_i \mid \text{label}(p) = -1\}$.

Lemma 5. *RPE probes $O(\log |C_i^{\text{pos}}|)$ points from C_i^{pos} in expectation.*

Proof. The operations that RPE performs on C_i^{pos} can be modeled as an A&E game on an initial input $C = C_i^{\text{pos}}$, as explained next.

In each round (of the A&E game), Bob formulates his strategy according to the execution of RPE. Suppose that RPE probes an element outside C_i^{pos} at Line 3 (of the pseudocode in Section 1.5). Recall that the algorithm removes some elements from P at Line 4 or 5. Accordingly, Bob carries out attrition by deleting from C all the elements of C_i^{pos} removed by RPE. After that, Bob observes the next probe of RPE and performs attrition in the same way as long as the element probed is outside C_i . If, on the other hand, RPE probes an element $z \in C_i^{\text{pos}}$, he passes the turn to Alice.

From Alice’s perspective, conditioned on $z \in C_i^{\text{pos}}$, RPE must have chosen z uniformly at random from the current C , namely, the set of elements from C_i^{pos} still in P . Hence, Alice can take z as her choice in the A&E game to perform elimination. Because z has label 1, after probing z at Line 3, RPE shrinks P by removing at Line 4 every element $p \succeq z$. As far as C is concerned, the shrinking deletes elements from C in exactly the way Alice should do in her elimination. This completes a round of the A&E game. The next round starts and proceeds in the same fashion.

By Lemma 4, the A&E game lasts for $O(\log |C_i^{\text{pos}}|)$ rounds in expectation regardless of Bob’s strategy. Hence, RPE probes $O(\log |C_i^{\text{pos}}|)$ elements from C_i^{pos} in expectation. $\square$

By a symmetric argument, RPE probes $O(\text{Log } |C_i^{neg}|)$ elements from C_i^{neg} in expectation. Therefore, the expected number of elements probed by RPE in total is given by

$$O\left(\sum_{i=1}^w \text{Log } |C_i^{pos}| + \text{Log } |C_i^{neg}|\right) = O\left(\sum_{i=1}^w \text{Log } |C_i|\right) = O\left(w \text{Log } \frac{n}{w}\right)$$

where the last step used $\sum_{i=1}^w |C_i| = n$ and the fact that $\sum_{i=1}^w \text{Log } |C_i|$ is maximized when all chains have the same size (a standard application of Jensen's inequality²). This completes the proof of Theorem 1.

2.3 Application to Monotonicity Testing

We finish this section with a remark on monotonicity testing. As reviewed in Section 1.3, given a multiset P of n points in $\mathbb{R}^d$ and a parameter $\xi \in (0, 1)$, the output of monotonicity testing is

- always “yes” if P is monotone;
- “no” with probability at least $2/3$ if $k^* \geq \xi n$ where k^* is the optimal monotone error of P (see (3));
- either “yes” or “no” if $0 < k^* < \xi n$.

Consider the following simple algorithm:

1. run RPE on P and obtain h_{RPE} from (10)
2. take a set S of $2/\xi$ uniform samples of P with replacement and obtain their labels
3. **if** h_{RPE} misclassifies any element in S **then return** “no”
4. **else return** “yes”

By Theorem 1, the algorithm probes $O(w \text{Log } \frac{n}{w}) + 2/\xi$ elements of P in expectation. Next, we explain why it fulfills the output requirements of monotonicity testing. First, if P is monotone, then the algorithm definitely outputs “yes” because, as mentioned before, h_{RPE} is guaranteed to classify all elements of P correctly in this case. On other hand hand, assume that $k^* \geq \xi n$. Because $\text{err}_P(h_{\text{RPE}}) \geq k^*$, the probability for h_{RPE} to misclassify a uniformly random element of P is at least $k^*/n \geq \xi$. Hence, the probability for h_{RPE} to be correct on all the elements in S is at most $(1 - \xi)^{2/\xi} < 1/e^2 < 1/3$. This means that the algorithm outputs “no” with probability at least $2/3$.

3 Relative-Comparison Coresets

This section solves Problem 1 up to an approximation ratio of $1 + \epsilon$ w.h.p. assuming $\epsilon \leq 1$ (for $\epsilon > 1$, reset it to 1). The central step is to find a relative-comparison coreset of the input P . Recall from Section 1.5 that this is a subset $Z \subseteq P$ where every element has its label revealed and carries a positive weight such that the weighted error of every monotone classifier h on Z — namely, $w\text{-err}_Z(h)$ defined in (12) — satisfies the condition in (13) (remember that the Δ value in (13) may be unknown). Formally, we establish:

²The discrete version of Jensen's inequality states that, for a concave function f and any values $x_1, \dots, x_w$ in its domain, $\frac{1}{w} \sum_{i=1}^w f(x_i) \leq f(\frac{1}{w} \sum_{i=1}^w x_i)$. Applying this inequality to the concave function $f(x) = \text{Log } x = \log_2(1 + x)$ and $x_i = |C_i| > 0$ yields $\frac{1}{w} \sum_{i=1}^w \text{Log } |C_i| \leq \text{Log } (\frac{1}{w} \sum_{i=1}^w |C_i|) = \text{Log}(n/w)$. Equality holds when $|C_1| = \dots = |C_w| = n/w$.

Theorem 6. *Let n and w be the size and width of the input P to Problem 1, respectively. In $O(\frac{w}{\epsilon^2} \text{Log } \frac{n}{w} \cdot \log \frac{n}{\delta})$ probes, we can obtain with probability at least $1 - \delta$ a relative-comparison coreset Z of P with size $|Z| = O(\frac{w}{\epsilon^2} \text{Log } \frac{w}{n} \cdot \log \frac{n}{\delta})$.*

As explained in Section 1.5, given the coreset Z in Theorem 6, we can solve Problem 1 by finding a monotone classifier h^* minimizing $w\text{-err}_Z(h^*)$. This requires no more probing and can be done in CPU time polynomial in $|Z|$ and d (see [2, 47]). This leads us to:

Corollary 7. *For Problem 1, there is an algorithm that finds w.h.p. a monotone classifier with an error at most $(1 + \epsilon)k^*$ by probing $O(\frac{w}{\epsilon^2} \text{Log } \frac{w}{n} \cdot \log n)$ elements, where n is the size of the input P , w is its width, and k^* is its optimal monotone error.*

The rest of the section serves as a proof of Theorem 6. The main difficulty arises from establishing its correctness for $d = 1$. Most of our discussion revolves around the following problem.

Problem 2. *Let P be a multiset of 1D labeled points as defined in Problem 1 ($d = 1$), and ϵ be a value in $(0, 1]$. Find a function $F : \mathbb{H}_{\text{mon}} \rightarrow \mathbb{R}$ such that every $h \in \mathbb{H}_{\text{mon}}$ satisfies the following two inequalities:*

$$|F(h) - \text{err}_P(h)| \leq \epsilon|P|/64 \quad (20)$$

$$\text{err}_P(h) \cdot \left(1 - \frac{\epsilon}{4}\right) + \Delta \leq F(h) \leq \text{err}_P(h) \cdot \left(1 + \frac{\epsilon}{4}\right) + \Delta \quad (21)$$

where Δ is a possibly unknown value with

$$|\Delta| \leq \epsilon|P|/64. \quad (22)$$

The efficiency of an algorithm is measured by the number of elements probed.

Sections 3.1-3.4 settles Problem 2 with probability at least $1 - \delta$ by probing $O(\frac{1}{\epsilon^2} \text{Log } \frac{1}{n} \cdot \log \frac{n}{\delta})$ elements. Section 3.5 then utilizes our solution to build a relative-comparison coreset that meets the requirements of Theorem 6 for $d = 1$. Section 3.6 extends the discussion to $d > 1$.

3.1 Warm Up: A Special Case

Solving Problem 2 requires proving the existence of Δ without deciding its value explicitly. This subsection illustrates the principle in the special case where all elements of P have an identical value. Any monotone classifier h maps the entire P to 1 or -1 . Denote by h^{pos} (resp., h^{neg}) the monotone classifier that always outputs 1 (resp., -1). It suffices to construct a function $F : \{h^{\text{pos}}, h^{\text{neg}}\} \rightarrow \mathbb{R}$ that satisfies (20)-(22) for $h \in \{h^{\text{pos}}, h^{\text{neg}}\}$.

It suffices to return an arbitrary function $F : \{h^{\text{pos}}, h^{\text{neg}}\} \rightarrow \mathbb{R}$ satisfying (20). It is standard to construct such a function via random sampling; the details are given in Section 3.4. Next, we argue for the existence of Δ that meets the requirements in (21) and (22). W.l.o.g., assume that the optimal monotone error k^* is achieved by h^{pos} , namely, $k^* = \text{err}_P(h^{\text{pos}})$. Our argument distinguishes two scenarios depending on whether k^* is large.

When $k^* \geq |P|/16$. In this scenario, we set $\Delta = 0$, which trivially satisfies (22). To prove (21), recall that $F(h)$ estimates $\text{err}_P(h)$ up to an additive offset of $\epsilon|P|/64$ for $h \in \{h^{\text{pos}}, h^{\text{neg}}\}$. Since $\text{err}_P(h) \geq |P|/16$, the offset is at most $\frac{\epsilon}{4}\text{err}_P(h)$. This yields $\text{err}_P(h) \cdot (1 - \frac{\epsilon}{4}) \leq F(h) \leq \text{err}_P(h) \cdot (1 + \frac{\epsilon}{4})$.

When $k^* < |P|/16$. In this scenario, we set

$$\Delta = F(h^{pos}) - k^*. \quad (23)$$

Note that Δ is not computable because k^* is unknown. Furthermore, as F satisfies (20) for $h = h^{pos}$, we have $|\Delta| = |F(h^{pos}) - k^*| = |F(h^{pos}) - \text{err}_P(h^{pos})| \leq \epsilon|P|/64$; hence, (22) holds.

Next, we prove (21) for $h \in \{h^{pos}, h^{neg}\}$. The case $h = h^{pos}$ is immediate because (21) reduces to $k^*(1 - \epsilon/4) \leq k^* \leq k^*(1 + \epsilon/4)$, which is clearly true. For h^{neg} , since $k^* < |P|/16$, we have $\text{err}_P(h^{neg}) = |P| - \text{err}_P(h^{pos}) \geq \frac{15}{16}|P|$. Thus, from (20):

$$|F(h^{neg}) - \text{err}_P(h^{neg})| \leq \frac{\epsilon|P|}{64} \leq \frac{\epsilon}{60} \cdot \text{err}_P(h^{neg}).$$

On the other hand, as explained earlier, $|\Delta| \leq \frac{\epsilon|P|}{64}$, which yields $|\Delta| \leq \frac{\epsilon}{60} \cdot \text{err}_P(h^{neg})$. Hence

$$|F(h^{neg}) - \text{err}_P(h^{neg}) - \Delta| \leq \frac{\epsilon}{60} \cdot \text{err}_P(h^{neg}) + \frac{\epsilon}{60} \cdot \text{err}_P(h^{neg}) < \frac{\epsilon}{4} \cdot \text{err}_P(h^{neg}).$$

We thus conclude that (21) holds for $h = h^{neg}$.

3.2 A Recursive Framework for Problem 2

This subsection discusses Problem 2 in its generic setting. If $P = \emptyset$, we return $F(h) = 0$ for all $h \in \mathbb{H}_{mon}$; if $|P| = 1$, we return

$$F(h) = \text{err}_P(h) \quad (24)$$

for all $h \in \mathbb{H}_{mon}$. In both cases, the function F satisfies (20)-(22) with $\Delta = 0$.

Our subsequent discussion assumes $|P| \geq 2$. When $d = 1$, a monotone classifier h takes the form

$$h(p) = \begin{cases} 1 & \text{if } p > \tau \\ -1 & \text{otherwise} \end{cases} \quad (25)$$

which is parameterized by a value τ . We sometimes make the parameter explicit by representing the classifier in (25) as h^τ .

Our framework assumes the availability of a function $G_1 : \mathbb{H}_{mon} \rightarrow \mathbb{R}$ having two properties:

- **G1-1:** G_1 approximates err_P up to an absolute error of $\epsilon|P|/64$, namely,

$$|G_1(h) - \text{err}_P(h)| \leq \epsilon|P|/64 \quad (26)$$

for all $h \in \mathbb{H}_{mon}$.

- **G1-2:** for any $\tau \in \mathbb{R}$, it holds that $G_1(h^\tau) = G_1(h^{\tau'})$ where τ' is the predecessor of τ in P (i.e., τ' is the largest element of P that is at most τ); specially, if τ' does not exist, then $G_1(h^\tau) = G_1(h^{-\infty})$ (note: $h^{-\infty}$ is simply h^{pos}).

Define

$$\alpha = \text{the smallest } \tau \in P \cup \{-\infty\} \text{ with } G_1(h^\tau) < |P| \cdot \left(\frac{1}{4} - \frac{\epsilon}{64}\right) \quad (27)$$

$$\beta = \text{the largest } \tau \in P \cup \{-\infty\} \text{ with } G_1(h^\tau) < |P| \cdot \left(\frac{1}{4} - \frac{\epsilon}{64}\right). \quad (28)$$

If neither α nor β exists — which occurs when $G_1(h) \geq |P|(\frac{1}{4} - \frac{\epsilon}{64})$ for all $h \in \mathbb{H}_{mon}$ due to property **G1-2** — define $\alpha = \beta = \text{null}$. Note that either α and β both exist, or neither does.

Break P into:

$$P_\alpha = \begin{cases} \emptyset & \text{if } \alpha = \beta = \text{null} \\ \{p \in P \mid p = \alpha\} & \text{otherwise} \end{cases} \quad (29)$$

$$P_{mid} = \begin{cases} \emptyset & \text{if } \alpha = \beta = \text{null} \\ \{p \in P \mid \alpha < p \leq \beta\} & \text{otherwise} \end{cases} \quad (30)$$

$$P_{rest} = P \setminus (P_\alpha \cup P_{mid}). \quad (31)$$

These are multisets where each point inherits its label in P .

Proposition 4. $|P_{mid}| < |P|/2$.

Proof. This is trivial if $\alpha = \beta = \text{null}$. Next, we consider that both α and β exist.

P_{mid} must have less than $|P|/4$ elements of label 1. To see why, assume that P has at least $|P|/4$ elements in $(\alpha, \beta]$ having label 1. Thus, $err_P(h^\beta) \geq |P|/4$, which together with (26) tells us $G_1(h^\beta) \geq |P|(\frac{1}{4} - \frac{\epsilon}{64})$, contradicting the definition of β . Similarly, P_{mid} must have less than $|P|/4$ elements of label -1 . To see why, assume that P has at least $|P|/4$ elements in $(\alpha, \beta]$ having label -1 . Thus, $err_P(h^\alpha) \geq |P|/4$, which together with (26) tells us $G_1(h^\alpha) \geq |P|(\frac{1}{4} - \frac{\epsilon}{64})$, contradicting the definition of α . $\square$

We also assume the availability of another function $G_2 : \mathbb{H}_{mon} \rightarrow \mathbb{R}$ having two properties:

- **G2-1:** G_2 approximates $err_{P_{rest}}$ up to absolute error $\epsilon|P_{rest}|/64$, namely, for any $h \in \mathbb{H}_{mon}$

$$|G_2(h) - err_{P_{rest}}(h)| \leq \epsilon|P_{rest}|/64; \quad (32)$$

- **G2-2** (applicable only if α and β exist): Define

$$\beta_{next} = \text{the smallest element of } P \text{ greater than } \beta \text{ or } \infty \text{ if no such element exists.} \quad (33)$$

For any $\tau \in [\alpha, \beta_{next})$, it holds that

$$G_2(h^\tau) = G_2(h^\beta). \quad (34)$$

By solving Problem 2 on P_α using the solution in Section 3.1 (all the elements in P_α have an identical value, which is α) and on P_{mid} recursively, we obtain functions $F_\alpha : \mathbb{H}_{mon} \rightarrow \mathbb{R}$ and $F_{mid} : \mathbb{H}_{mon} \rightarrow \mathbb{R}$ such that every $h \in \mathbb{H}_{mon}$ satisfies (35)-(38):

$$|F_\alpha(h) - err_{P_\alpha}(h)| \leq \epsilon|P_\alpha|/64 \quad (35)$$

$$|F_{mid}(h) - err_{P_{mid}}(h)| \leq \epsilon|P_{mid}|/64 \quad (36)$$

$$err_{P_\alpha}(h) \cdot \left(1 - \frac{\epsilon}{4}\right) + \Delta_\alpha \leq F_\alpha(h) \leq err_{P_\alpha}(h) \cdot \left(1 + \frac{\epsilon}{4}\right) + \Delta_\alpha \quad (37)$$

$$err_{P_{mid}}(h) \cdot \left(1 - \frac{\epsilon}{4}\right) + \Delta_{mid} \leq F_{mid}(h) \leq err_{P_{mid}}(h) \cdot \left(1 + \frac{\epsilon}{4}\right) + \Delta_{mid} \quad (38)$$

where Δ_α and Δ_{mid} are (unknown) real values such that

$$|\Delta_\alpha| \leq \epsilon|P_\alpha|/64 \quad (39)$$

$$|\Delta_{mid}| \leq \epsilon|P_{mid}|/64 \quad (40)$$

The target function F for Problem 2 can now be finalized as

$$F(h) = G_2(h) + F_\alpha(h) + F_{mid}(h). \quad (41)$$

As a remark, if $\alpha = \beta = \text{null}$, then $F_\alpha(h) = F_{mid}(h) = 0$, leaving $F(h) = G_2(h)$.

3.3 Correctness of the Framework

Next, we prove that the above framework always produces a function F obeying (20)-(22). The base case where $|P| = 0$ or 1 has been resolved in Section 3.2. Inductively, assuming that (36), (38), and (40) hold on P_{mid} , we show that the function F in (41) satisfies (20)-(22) with

$$\Delta = \begin{cases} 0 & \text{if } \alpha = \beta = \text{null} \\ \Delta_\alpha + \Delta_{mid} + G_2(h^\beta) - \text{err}_{P_{rest}}(h^\beta) & \text{otherwise} \end{cases} \quad (42)$$

Proof of (20). For any $h \in \mathbb{H}_{mon}$, it holds that

$$\text{err}_P(h) = \text{err}_{P_\alpha}(h) + \text{err}_{P_{mid}}(h) + \text{err}_{P_{rest}}(h). \quad (43)$$

Combining the above with (41) gives:

$$\begin{aligned} |F(h) - \text{err}_P(h)| &\leq |F_\alpha(h) - \text{err}_{P_\alpha}(h)| + |F_{mid}(h) - \text{err}_{P_{mid}}(h)| + |G_2(h) - \text{err}_{P_{rest}}(h)| \\ (\text{by (32), (35), and (36)}) &\leq \frac{\epsilon|P_\alpha|}{64} + \frac{\epsilon|P_{mid}|}{64} + \frac{\epsilon|P_{rest}|}{64} \\ &\leq \frac{\epsilon|P|}{64}. \end{aligned}$$

Proof of (22). If $\alpha = \beta = \text{null}$, then $\Delta = 0$ trivially satisfies (22). Otherwise, from (42), (32), (39), and (40), we know

$$|\Delta| = |\Delta_\alpha + \Delta_{mid} + G_2(h^\beta) - \text{err}_{P_{rest}}(h^\beta)| \leq \frac{\epsilon|P_\alpha|}{64} + \frac{\epsilon|P_{mid}|}{64} + \frac{\epsilon|P_{rest}|}{64} \leq \frac{\epsilon|P|}{64}. \quad (44)$$

Proof of (21). We first present a useful lemma.

Lemma 8. $F(h)$ satisfies (21) whenever $G_1(h) \geq |P|(\frac{1}{4} - \frac{\epsilon}{64})$.

Proof. From (26) and $G_1(h) \geq |P|(\frac{1}{4} - \frac{\epsilon}{64})$, we have

$$\text{err}_P(h) \geq G_1(h) - \frac{\epsilon|P|}{64} \geq \frac{|P|}{4} - \frac{\epsilon|P|}{64} - \frac{\epsilon|P|}{64} \geq \frac{14|P|}{64} \quad (45)$$

where the last step used $\epsilon \leq 1$. Combining the above with (44) yields

$$\frac{\epsilon \cdot \text{err}_P(h)}{4} + \Delta \geq \frac{\epsilon}{4} \cdot \frac{14|P|}{64} - \frac{\epsilon|P|}{64} = \frac{10\epsilon|P|}{256}.$$

As proved earlier, F satisfies (20); hence:

$$F(h) - \text{err}_P(h) \leq \frac{\epsilon|P|}{64} < \frac{10\epsilon|P|}{256} \leq \frac{\epsilon \cdot \text{err}_P(h)}{4} + \Delta. \quad (46)$$

Similarly, from (44) and (45), we know

$$\frac{\epsilon \cdot \text{err}_P(h)}{4} - \Delta \geq \frac{\epsilon}{4} \cdot \frac{14|P|}{64} - \frac{\epsilon|P|}{64} = \frac{10\epsilon|P|}{256}.$$

Hence, (20) tells us

$$\text{err}_P(h) - F(h) \leq \frac{\epsilon|P|}{64} < \frac{10\epsilon|P|}{256} \leq \frac{\epsilon \cdot \text{err}_P(h)}{4} - \Delta. \quad (47)$$

The correctness of (21) now follows from (46) and (47). $\square$

As a corollary, when $\alpha = \beta = \text{null}$ (namely, $G_1(h) \geq |P|(\frac{1}{4} - \frac{\epsilon}{64})$ for all $h \in \mathbb{H}_{mon}$), $F(h)$ satisfies (21) for all $h \in \mathbb{H}_{mon}$.

Now, we consider that α and β exist. By property **G1-2** and the definitions of α and β , we know $G_1(h^\tau) \geq |P|(\frac{1}{4} - \frac{\epsilon}{64})$ for any τ satisfying $\tau < \alpha$ and $\tau \geq \beta_{next}$ (the reader may want to revisit the definitions of α, β , and β_{next} in (27), (28), and (33)). By Lemma 8, $F(h^\tau)$ must satisfy (21) for those τ values. It remains to prove that this is also true for $\tau \in [\alpha, \beta_{next})$, as we show next.

Lemma 9. *If α and β exist, $F(h^\tau)$ satisfies (21) for $\tau \in [\alpha, \beta_{next})$.*

Proof. For any $\tau \in [\alpha, \beta_{next})$, we have

$$err_{P_{rest}}(h^\tau) = err_{P_{rest}}(h^\beta) \quad (48)$$

because P_{rest} has no element in $[\alpha, \beta_{next})$. By property **G2-2**, $G_2(h^\tau) = G_2(h^\beta)$ for all $\tau \in [\alpha, \beta_{next})$. This, together with (41), yields

$$F(h^\tau) = G_2(h^\beta) + F_\alpha(h^\tau) + F_{mid}(h^\tau). \quad (49)$$

We can thus derive

$$\begin{aligned} & err_P(h^\tau)(1 + \epsilon/4) + \Delta \\ \text{(by (43) and (48))} &= (err_{P_\alpha}(h^\tau) + err_{P_{mid}}(h^\tau) + err_{P_{rest}}(h^\beta))(1 + \epsilon/4) + \Delta \\ &\geq (err_{P_\alpha}(h^\tau) + err_{P_{mid}}(h^\tau))(1 + \epsilon/4) + err_{P_{rest}}(h^\beta) + \Delta \\ \text{(by (42))} &= (err_{P_\alpha}(h^\tau) + err_{P_{mid}}(h^\tau))(1 + \epsilon/4) + \Delta_\alpha + \Delta_{mid} + G_2(h^\beta) \\ \text{(by (37) and (38))} &\geq F_\alpha(h^\tau) + F_{mid}(h^\tau) + G_2(h^\beta) \\ \text{(by (49))} &= F(h^\tau). \end{aligned}$$

Similarly,

$$\begin{aligned} & err_P(h^\tau)(1 - \epsilon/4) + \Delta \\ \text{(by (43) and (48))} &= (err_{P_\alpha}(h^\tau) + err_{P_{mid}}(h^\tau) + err_{P_{rest}}(h^\beta))(1 - \epsilon/4) + \Delta \\ &\leq (err_{P_\alpha}(h^\tau) + err_{P_{mid}}(h^\tau))(1 - \epsilon/4) + err_{P_{rest}}(h^\beta) + \Delta \\ \text{(by (42))} &= (err_{P_\alpha}(h^\tau) + err_{P_{mid}}(h^\tau))(1 - \epsilon/4) + \Delta_\alpha + \Delta_{mid} + G_2(h^\beta) \\ \text{(by (37) and (38))} &\leq F_\alpha(h^\tau) + F_{mid}(h^\tau) + G_2(h^\beta) \\ &= F(h^\tau). \end{aligned}$$

This completes the proof. $\square$

3.4 A Concrete Algorithm for Problem 2

Instantiating our framework in Section 3.2 requires constructing the function F in Section 3.1 and the functions G_1 and G_2 in Section 3.2. We explain how to do so in this subsection, by factoring in the consideration that the framework needs to succeed with probability at least $1 - \delta$. Denote by ℓ the number of recursion levels; the value ℓ is $O(\log n)$ due to Proposition 4.

Constructing G_1 and G_2 . Both G_1 and G_2 map $\mathbb{H}_{mon}$ to $\mathbb{R}$. Although $\mathbb{H}_{mon}$ has an infinite size, there exists a finite set of “effective” classifiers:

$$\mathbb{H}_{mon}^P = \{h^\tau \mid \tau \in P \text{ or } \tau = -\infty\}.$$

Every monotone classifier has the same error on P as a classifier in $\mathbb{H}_{mon}^P$.

To build G_1 , we uniformly sample with replacement a set S_1 of $O(\frac{1}{\epsilon^2} \log \frac{|P|^\ell}{\delta})$ elements from P . For each $h \in \mathbb{H}_{mon}$, define

$$G_1(h) = \frac{|P|}{|S_1|} \cdot \text{err}_{S_1}(h). \quad (50)$$

As discussed in Appendix A, $G_1(h)$ satisfies (26) with probability at least $1 - \frac{\delta}{3\ell \cdot (|P|+1)}$ for each $h \in \mathbb{H}_{mon}^P$. As $|\mathbb{H}_{mon}^P| \leq |P| + 1$, G_1 satisfies (26) for all $h \in \mathbb{H}_{mon}^P$ — thus for all $h \in \mathbb{H}_{mon}$ — with probability at least $1 - \delta/(3\ell)$. It is clear that G_1 has property **G1-2**.

If P_{rest} — defined in (31) — is empty, we set $G_2(h) = 0$ for all $h \in \mathbb{H}_{mon}$. Otherwise, we uniformly sample with replacement a set S_2 of $O(\frac{1}{\epsilon^2} \log \frac{|P|^\ell}{\delta})$ elements from P_{rest} . For each $h \in \mathbb{H}_{mon}$, define

$$G_2(h) = \frac{|P_{rest}|}{|S_2|} \cdot \text{err}_{S_2}(h). \quad (51)$$

An argument analogous to the one used earlier for G_1 shows that G_2 obeys (32) for all $h \in \mathbb{H}_{mon}$ with probability at least $1 - \delta/(3\ell)$. G_2 has property **G2-2** because S_2 has no elements in $[\alpha, \beta_{next})$, where α and β_{next} are defined in (27) and (33), respectively.

Constructing the Function F in Section 3.1. Our framework applies the method in Section 3.1 to solve Problem 2 on a non-empty P_α — defined in (29) — whose goal is to obtain a function F_α satisfying (35). To build F_α (a.k.a. the function F in Section 3.1 when $P = P_\alpha$), we uniformly sample with replacement a set S_α of $O(\frac{1}{\epsilon^2} \log \frac{|P|^\ell}{\delta})$ elements from P_α . For each $h \in \mathbb{H}_{mon}$, define

$$F_\alpha(h) = \frac{|P_\alpha|}{|S_\alpha|} \cdot \text{err}_{S_\alpha}(h). \quad (52)$$

It satisfies (35) for all $h \in \mathbb{H}_{mon}$ with probability at least $1 - \delta/(3\ell)$.

Putting All Levels Together. In summary, at each recursion level, by probing $O(\frac{1}{\epsilon^2} \log \frac{|P|^\ell}{\delta})$ elements we can construct the desired functions G_1 , G_2 , and F_α with probability at least $1 - \delta/\ell$. As there are ℓ levels, the overall cost is $O(\frac{\ell}{\epsilon^2} \log \frac{n}{\delta}) = O(\frac{\log n}{\epsilon^2} \cdot \log \frac{n}{\delta})$, and we solve Problem 2 with probability at least $1 - \delta$.

3.5 A One-Dimensional Relative-Comparison Coreset

We are ready to prove Theorem 6 for $d = 1$. Let us re-examine our algorithm (combining Sections 3.1, 3.2, and 3.4) and construct a coreset Z along the way.

- If $|P| = 1$, our algorithm probes the only element $p \in P$, sets $Z = P$, and assigns $\text{weight}(p) = 1$.
- If $|P| \geq 2$, our algorithm first produces the sets P_α , P_{mid} , and P_{rest} in (29)-(31) using the function G_1 built in Section 3.4. If $P_{rest} \neq \emptyset$, it adds to Z the sample set S_2 described in Section 3.4 and assigns $\text{weight}(p) = |P_{rest}|/|S_2|$ for each $p \in S_2$. If $P_\alpha \neq \emptyset$, it adds to Z the sample set S_α described in Section 3.4 and assigns $\text{weight}(p) = |P_\alpha|/|S_\alpha|$ for each $p \in S_\alpha$. The recursion on P_{mid} returns a coreset $Z_{mid} \subseteq P_{mid}$, which is also included in Z .

The following pseudocode summarizes the above steps.

algorithm BuildCoreset (P)

1. $Z = \emptyset$
2. **if** $|P| = 1$ **then** probe the (only) element $p \in P$, and set $Z = P$ with $weight(p) = 1$
else
3. probe the sample set S_1 described in Section 3.4
/* this defines G_1 (see (50)), which in turn defines $P_\alpha, P_{mid}, P_{rest}$ (see (29)-(31)) */
4. **if** $P_{rest} \neq \emptyset$ **then**
5. probe the sample set S_2 described in Section 3.4 /* this defines G_2 (see (51)) */
6. add S_2 to Z with $weight(p) = |P_{rest}|/|S_2|$ for each $p \in S_2$
7. **if** $P_\alpha \neq \emptyset$ **then**
8. probe the sample set S_α described in Section 3.4 /* this defines F_α (see (52)) */
9. add S_α to Z with $weight(p) = |P_\alpha|/|S_\alpha|$ for each $p \in S_\alpha$
10. $Z_{mid} = \text{Build-Coreset}(P_{mid})$; add Z_{mid} to Z
11. **return** Z

By the discussion in Section 3.4, BuildCoreset returns a set Z of $O(\frac{\log n}{\epsilon^2} \cdot \log \frac{n}{\delta})$ elements. It is rudimentary to verify that for every $h \in \mathbb{H}_{mon}$, we have

$$w\text{-err}_Z(h) = F(h)$$

where F is given in (24) if $|P| = 1$ or in (41) otherwise. This establishes Theorem 6 for $d = 1$.

3.6 Arbitrary Dimensions

This subsection proves Theorem 6 for any $d \geq 2$. As before, denote by n and w the size and width of the input P , respectively. We start by computing a chain decomposition of P with w chains: $C_1, C_2, \dots, C_w$. Such a decomposition can be computed in time polynomial in d and n (see [49]) without any probing.

For every $i \in [w]$, we compute a subset $Z_i \subseteq C_i$ where every element $p \in Z_i$ has its label revealed and carries a weight $weight(p) > 0$. The set Z_i ensures

$$err_{C_i}(h) \cdot \left(1 - \frac{\epsilon}{4}\right) - \Delta_i \leq w\text{-err}_{Z_i}(h) \leq err_{C_i}(h) \cdot \left(1 - \frac{\epsilon}{4}\right) + \Delta_i \quad (53)$$

for every $h \in \mathbb{H}_{mon}$, where Δ_i is a possibly unknown value. Then, we obtain

$$Z = \bigcup_{i=1}^w Z_i. \quad (54)$$

For every $h \in \mathbb{H}_{mon}$, it holds that

$$err_P(h) \cdot \left(1 - \frac{\epsilon}{4}\right) - \Delta \leq w\text{-err}_Z(h) \leq err_P(h) \cdot \left(1 - \frac{\epsilon}{4}\right) + \Delta \quad (55)$$

where

$$\Delta = \sum_{i=1}^w \Delta_i.$$

Finding Z_i for an $i \in [w]$ is a 1D problem. To explain, let us sort the elements of C_i in “ascending” order (i.e., if p precedes q in the ordering, then $p \preceq q$). A monotone classifier h maps a prefix of the

ordering to -1 ; hence, as far as C_i is concerned, we can regard h as a 1D classifier of the form (25). As Theorem 6 holds for $d = 1$, we can apply it to produce the desired Z_i with probability at least $1 - \frac{\delta}{w}$ by probing $O(\frac{1}{\epsilon^2} \cdot \text{Log } |C_i| \cdot \log(wn/\delta)) = O(\frac{\log(n/\delta)}{\epsilon^2} \text{Log } |C_i|)$ elements from C_i . This Z_i has size $O(\frac{\log(n/\delta)}{\epsilon^2} \text{Log } |C_i|)$.

Therefore, with probability at least $1 - 1/\delta$, we can produce $Z_1, \dots, Z_w$ with

$$O\left(\frac{\log(n/\delta)}{\epsilon^2} \sum_{i=1}^w \text{Log } |C_i|\right) = O\left(\frac{\log(n/\delta)}{\epsilon^2} \cdot w \text{Log } \frac{n}{w}\right)$$

probes in total. The same bound also applies to the size of Z in (54).

4 Optimal Monotone Classification Needs $\Omega(n)$ Probes

This section focuses on Problem 1 under $\epsilon = 0$; namely, the objective is to find an optimal monotone classifier. A naive solution is to probe everything in the input P . We prove that this is already the best approach up to a constant factor:

Theorem 10. *For Problem 1, any algorithm promising to find an optimal classifier with probability greater than $2/3$ must probe $\Omega(n)$ elements in expectation, where n is the size of the input P . This is true even if the dimensionality d is 1, and the algorithm knows the optimal monotone error k^* of P .*

The rest of the section serves as a proof of the theorem. Assume, w.l.o.g., that n is an even number. We construct a family $\mathbb{P}$ of n one-dimensional inputs. Every input of $\mathbb{P}$ has n elements positioned at values $1, 2, \dots, n$. Various inputs differ in their label assignments. Specifically, each $i \in [n/2]$ defines two inputs in $\mathbb{P}$:

- $P_{-1}(i)$, where every odd (resp., even) element has label 1 (resp., -1), with $2i - 1$ being the only exception, which is assigned label -1 instead. We refer to $P_{-1}(i)$ as a (-1) -input.
- $P_1(i)$, where every odd (resp., even) element has label 1 (resp., -1), with $2i$ being the only exception, which is assigned label 1 instead. We refer to $P_1(i)$ as a 1 -input.

The constructed family $\mathbb{P} = \{P_{-1}(1), P_{-1}(2), \dots, P_{-1}(n/2), P_1(1), P_1(2), \dots, P_1(n/2)\}$ can be understood in an alternative manner. Group the elements $1, 2, \dots, n$ into $n/2$ pairs $(1, 2), (3, 4), \dots, (n-1, n)$. In a *normal pair* $(x-1, x)$, elements $x-1$ and x carry labels 1 and -1 , respectively. Each input $P \in \mathbb{P}$ contains one *anomaly pair* $(x-1, x)$: if P is a (-1) -input, both $x-1$ and x have label -1 ; otherwise, they have label 1.

The optimal monotone error k^* is $n/2 - 1$ for each input $P \in \mathbb{P}$. Indeed, every monotone classifier must misclassify at least one element in each normal pair. On the other hand, the error $n/2 - 1$ is attainable by mapping all the elements to 1 for a 1-input or -1 for a (-1) -input.

An algorithm $\mathcal{A}$ for Problem 1 *errs* on an input $P \in \mathbb{P}$ if it fails to find an optimal classifier for P . Denote by $\text{cost}_P(\mathcal{A})$ the number of probes performed by $\mathcal{A}$ when executed on P ; this is a random variable if $\mathcal{A}$ is randomized. Define

$$\begin{aligned} \text{family-err}(\mathcal{A}) &= \sum_{P \in \mathbb{P}} \mathbf{Pr}[\mathcal{A} \text{ errs on } P] \\ \text{family-cost}(\mathcal{A}) &= \sum_{P \in \mathbb{P}} \text{cost}_P(\mathcal{A}). \end{aligned}$$

If $\mathcal{A}$ deterministic, $\mathbf{Pr}[\mathcal{A} \text{ errs on } P]$ is either 0 or 1 for each $P \in \mathbb{P}$. In Section 4.1, we prove:

Lemma 11. Fix any deterministic algorithm $\mathcal{A}_{det}$ and an arbitrary non-negative constant $c < 1$. For $n \geq 4$, if $\text{family-err}(\mathcal{A}_{det}) \leq cn/2$, then $\text{family-cost}(\mathcal{A}_{det}) \geq \frac{n^2}{4}(1 - c^2)$.

We now utilize the lemma to prove a hardness result for randomized algorithms.

Corollary 12. When $n \geq 4$, the following holds for any randomized algorithm $\mathcal{A}$: if $\text{family-err}(\mathcal{A}) < n/3$, then $\mathbf{E}[\text{family-cost}(\mathcal{A})] = \Omega(n^2)$.

Proof. A randomized algorithm becomes deterministic when all the random bits are fixed. Hence, we can treat $\mathcal{A}$ as a distribution over a family $\mathbb{A}$ of deterministic algorithms, each sampled possibly with a different probability. We say that an algorithm $\mathcal{A}_{det} \in \mathbb{A}$ is *accurate* if $\text{family-err}(\mathcal{A}_{det}) \leq (2/5)n$. Define $\mathbb{A}_{acc}$ as the set of accurate algorithms in $\mathbb{A}$.

We argue that $\Pr[\mathcal{A} \in \mathbb{A}_{acc}] > 1/6$. If $\Pr[\mathcal{A} \notin \mathbb{A}_{acc}] \geq 5/6$, then

$$\begin{aligned} \text{family-err}(\mathcal{A}) &= \sum_{\mathcal{A}_{det} \in \mathbb{A}} \text{family-err}(\mathcal{A}_{det}) \cdot \Pr[\mathcal{A} = \mathcal{A}_{det}] \\ &\geq \sum_{\mathcal{A}_{det} \notin \mathbb{A}_{acc}} \text{family-err}(\mathcal{A}_{det}) \cdot \Pr[\mathcal{A} = \mathcal{A}_{det}] \\ &\geq \frac{2n}{5} \sum_{\mathcal{A}_{det} \notin \mathbb{A}_{acc}} \Pr[\mathcal{A} = \mathcal{A}_{det}] \\ &= \frac{2n}{5} \cdot \Pr[\mathcal{A} \notin \mathbb{A}_{acc}] \geq \frac{2n}{5} \cdot \frac{5}{6} = n/3 \end{aligned}$$

contradicting the condition $\text{family-err}(\mathcal{A}) < n/3$.

By Lemma 11, every accurate $\mathcal{A}_{det} \in \mathbb{A}_{acc}$ satisfies $\text{family-cost}(\mathcal{A}_{det}) = \Omega(n^2)$. We thus have $\mathbf{E}[\text{family-cost}(\mathcal{A})] \geq \Omega(n^2) \cdot \Pr[\mathcal{A} \in \mathbb{A}_{acc}] = \Omega(n^2)$. $\square$

The corollary implies Theorem 10. Indeed, if $\mathcal{A}$ guarantees finding an optimal classifier with probability greater than $2/3$ on any input, then $\text{family-err}(\mathcal{A}) < |\mathbb{P}|/3 = n/3$. By Corollary 12, $\mathbf{E}[\text{family-cost}(\mathcal{A})] = \Omega(n^2)$ when $n \geq 4$. This means that the expected cost of $\mathcal{A}$ is $\Omega(n)$ on at least one input in $\mathbb{P}$ since $\mathbb{P}$ has n inputs.

4.1 Proof of Lemma 11

We start with a property of the family $\mathbb{P}$.

Proposition 5. For any $i \in [n/2]$, no monotone classifier can be optimal for both $P_{-1}(i)$ and $P_1(i)$.

Proof. As mentioned, the optimal monotone error is $n/2 - 1$ for each input of $\mathbb{P}$. A 1D monotone classifier h has the form in (25), which is parameterized by a value τ ; next, we denote the classifier as h^τ . To argue that no h^τ is optimal for both $P_{-1}(i)$ and $P_1(i)$, we examine all possible scenarios.

- Case $\tau < 2i - 1$: on $P_{-1}(i)$, h^τ misclassifies both $2i - 1$ and $2i$ and has error $n/2 + 1$.
- Case $\tau = 2i - 1$: on $P_{-1}(i)$, h^τ misclassifies $2i$ and has error $n/2$.
- Case $\tau \geq 2i$: on $P_1(i)$, h^τ misclassifies both $2i - 1$ and $2i$ and has error $n/2 + 1$.

This completes the proof. $\square$

To prove Lemma 11, we strengthen $\mathcal{A}_{det}$ by giving it certain “free” labels. Every time $\mathcal{A}_{det}$ probes an element of some pair $(2i - 1, 2i)$, where $i \in [n/2]$, we reveal the label for the other element (of the pair) voluntarily. Henceforth, $\mathcal{A}_{det}$ is said to “probe pair i ” if $\mathcal{A}_{det}$ probes either $2i - 1$ or $2i$. If Lemma 11 holds even on such an “empowered” $\mathcal{A}_{det}$, it must hold on the original $\mathcal{A}_{det}$ because an empowered algorithm can choose to ignore the free information.

We consider, w.l.o.g., that $\mathcal{A}_{det}$ terminates immediately after probing an anomaly pair — identifying the pair enables $\mathcal{A}_{det}$ to output an optimal classifier because the labels in normal pairs are fixed. Thus, we can model $\mathcal{A}_{det}$ as a procedure probing a fixed sequence: pair x_1 , pair x_2 , ..., pair x_t up to some $t \in [0, n/2]$. For each $j \in [t - 1]$, if x_j is an anomaly, $\mathcal{A}_{det}$ terminates; otherwise, it moves on to probe x_{j+1} . If all the t pairs are probed but no anomaly is found, $\mathcal{A}_{det}$ outputs a fixed classifier, denoted as h_{det} .

As $\mathcal{A}_{det}$ never probes pair i for

$$i \in \{1, 2, \dots, n/2\} \setminus \{x_1, x_2, \dots, x_t\} \quad (56)$$

its output must be h_{det} on both $P_{-1}(i)$ and $P_1(i)$. By Proposition 5, h_{det} cannot be optimal for both $P_{-1}(i)$ and $P_1(i)$, which gives

$$\text{family-err}(\mathcal{A}_{det}) \geq n/2 - t. \quad (57)$$

To analyze its cost, note that $\mathcal{A}_{det}$ performs t probes for $P_{-1}(i)$ and $P_1(i)$ of every i satisfying (56), but $j \in [t]$ probes for $P_{-1}(x_j)$ and $P_1(x_j)$. Hence

$$\text{family-cost}(\mathcal{A}_{det}) = 2t \cdot (n/2 - t) + 2 \sum_{j=1}^t j = nt - t^2 + t. \quad (58)$$

If $\text{family-err}(\mathcal{A}_{det})$ needs to be at most $cn/2$, then $t \geq \frac{n}{2}(1 - c)$ by (57). On the other hand, for $t \in [\frac{n}{2}(1 - c), \frac{n}{2}]$, we have (58) $\geq nt - t^2 \geq \frac{n^2}{4}(1 - c^2)$. This completes the proof of Lemma 11.

5 A Lower Bound for Constant Approximation Ratios

We now proceed to study the hardness of approximation. The main result of this section is:

Theorem 13. *Let n' , w' , k , and c be arbitrary integers satisfying $n' \geq 2, w' \geq 1, k \geq 0, c \geq 1$, and n' is a multiple of w' . Set*

$$n = n' + 2k + 2ckn' \quad (59)$$

$$w = w' + \mathbb{1}_{k \geq 1}. \quad (60)$$

For Problem 1, there is a family $\mathbb{P}$ of inputs with size n , width w , and optimal monotone error $k^* = k$ such that any randomized algorithm, which guarantees an expected error at most $c \cdot k^*$, must entail an expected cost of $\Omega(w' \log \frac{n'}{w'})$ on at least one input of $\mathbb{P}$, where $\Omega(\cdot)$ hides a constant that is not dependent on n' , w' , k , and c . The claim holds even if the algorithm knows k^* .

The theorem is particularly useful when the approximation ratio c is a constant. When $k = 0$ (realizable), the theorem gives a lower bound $\Omega(w \log \frac{n}{w})$. For $k \geq 1$ (non-realizable), we always have $n' + 2k \leq n/2$, because of which

$$\frac{n'}{w'} = \frac{n - (n' + 2k)}{2ckw'} \geq \frac{n}{4ckw'}.$$

Theorem 13 thus implies a lower bound $\Omega(w \log \frac{n}{k^*w})$ when w is sufficiently large. We first prove the theorem for $k = 0$ in Section 5.1 and then for $k \geq 1$ in Section 5.2.

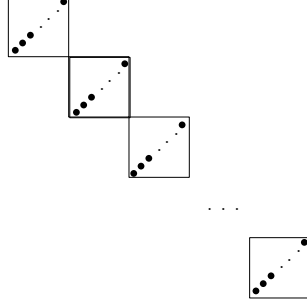


Figure 5: A hard realizable input for Problem 1: w boxes each with n/w points

5.1 The Realizable Case

We use the term *box* to refer to an axis-parallel rectangle with a positive area in $\mathbb{R}^2$. The *main diagonal* of the box is the segment connecting its bottom-left and top-right corners. We say that two boxes B_1 and B_2 are *independent* if no point in B_1 dominates any point in B_2 and vice versa.

For $k^* = 0$, an algorithm that guarantees an expected error at most $ck^* = 0$ must always find an optimal classifier. To prove Theorem 13 for $k = 0$, we construct hard inputs as follows. Let $B_1, B_2, \dots, B_{w'}$ be arbitrary mutually independent boxes. For each $i \in [w']$, place n'/w' points on the main diagonal of B_i , making sure that they are at distinct locations and no point lies at a corner of B_i . This yields a set P of n' points with width w' ; see Figure 5. Label assignment is done for each box independently, subject to the constraint that P is monotone. In each box, there are $1 + \frac{n'}{w'}$ assignments: for each $i \in [0, n'/w']$, assign label -1 to the i lowest points in the box and 1 to the rest. This gives a family $\mathbb{P}_{n',w'}$ of $(1 + n'/w')^{w'}$ labeled point sets.

A deterministic algorithm $\mathcal{A}_{det}$ is a binary decision tree $\mathcal{T}$. If $\mathcal{A}_{det}$ is always correct for $k = 0$, it must distinguish all the inputs in $\mathbb{P}_{n',w'}$ by returning a different classifier for each input (no classifier is optimal for two inputs in $\mathbb{P}_{n',w'}$). The number of leaves in $\mathcal{T}$ is thus at least $(1 + n'/w')^{w'}$. The *average cost* of $\mathcal{A}_{det}$ — where the average is over the inputs of $\mathbb{P}_{n',w'}$ — equals the average depth of the leaves in $\mathcal{T}$. A binary tree with at least $(1 + n'/w')^{w'}$ leaves must have an average depth of $\Omega(w' \log \frac{n'}{w'})$. Hence, $\mathcal{A}_{det}$ must have average cost $\Omega(w \log \frac{n}{w})$.

By Yao's minimax theorem [41], any randomized algorithm that is always correct for $k = 0$ must entail an expected cost of $\Omega(w' \log \frac{n'}{w'})$ on at least one input of $\mathbb{P}_{n',w'}$.

5.2 The Non-Realizable Case

This subsection serves as a proof of Theorem 13 for $k \geq 1$.

Algorithms with Guessing Power. We first strengthen the power of deterministic algorithms. As before, such an algorithm $\mathcal{A}_{det}$ is described by a binary decision tree $\mathcal{T}$ determined by the point locations in the input P . Different from the decision tree in Section 1.1, however, we allow two types of internal nodes:

- *Probe node.* This is the (only) type of internal nodes allowed in Section 1.1.
- *Guess node.* At such a node, $\mathcal{A}_{det}$ proposes a monotone classifier h and asks an oracle whether $err_P(h)$ is at most a certain value t . Both h and t are fixed at this node. On a “yes” answer from the oracle, $\mathcal{A}_{det}$ descends to the left child, which must be a leaf returning h . On a “no” answer, $\mathcal{A}_{det}$ branches right and continues.

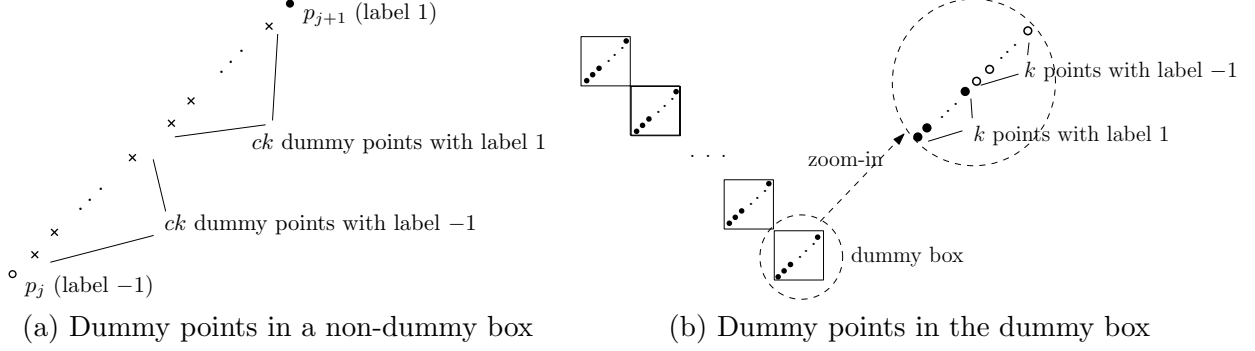


Figure 6: Adding dummy points for a Las Vegas lower bound

We charge one unit of cost for every node: probe or guess. A randomized algorithm is still modeled as a function that maps a random-bit sequence to a deterministic algorithm. A lower bound on such empowered algorithms must also hold on algorithms that use probe nodes only.

The argument in Section 5.1 has proved that any deterministic algorithm in the form of a binary decision tree must have an average cost of $\Omega(w' \text{Log } \frac{n'}{w'})$ over the inputs of $\mathbb{P}_{n',w'}$. Hence, the lower bound applies to deterministic algorithms with guess nodes. By Yao's minimax theorem, any randomized algorithm with guess nodes must incur $\Omega(w' \text{Log } \frac{n'}{w'})$ expected cost on at least one input of $\mathbb{P}(n', w')$ if it always returns an optimal classifier.

A Las Vegas Lower Bound. Let n', k', k, c, n , and w be as stated in Theorem 13. Denote by $\mathcal{A}$ a randomized algorithm (with guessing power) that, when executed on an input having size n , width w , and optimal monotone error $k^* = k$, guarantees

- returning a monotone classifier whose error on the input is at most ck^* , and
- an expected cost at most J_{LV} .

We show that $J_{LV} = \Omega(w' \text{Log } \frac{n'}{w'})$ by using $\mathcal{A}$ to process the inputs from $\mathbb{P}(n', w')$.

Given an input $P' \in \mathbb{P}_{n',w'}$, we construct a set P of labeled points as follows. First, set $P = P'$. Recall that the points of P' are inside w' boxes $B_1, B_2, \dots, B_{w'}$, each having n'/w' points. For each $i \in [w']$, let $p_1, p_2, \dots, p_{n'/w'}$ be the points of P in B_i , sorted in ascending order of y-coordinate. For each $j \in [n'/w' - 1]$, place $2ck$ dummy points on the main diagonal of B_i between p_j and p_{j+1} . Assign labels to these dummy points as follows:

- If $\text{label}(p_j) = \text{label}(p_{j+1})$, set the labels of all $2ck$ dummy points to $\text{label}(p_j)$.
- Otherwise, we must have $\text{label}(p_j) = -1$ and $\text{label}(p_{j+1}) = 1$ (because P' is monotone); set the labels of the ck lowest dummy points to -1 and the labels of the other dummy points to 1 ; see Figure 6a.

Furthermore, add ck dummy points to P between the bottom-left corner of B_i and p_1 . Set their labels to -1 if $\text{label}(p_1) = -1$, or 1 otherwise. Symmetrically, add to P another ck dummy points between $p_{n'/w'}$ and the top-right corner of B_i . Set their labels to 1 if $\text{label}(p_{n'/w'}) = 1$, or -1 otherwise. Finally, create a dummy box that is independent from all of $B_1, \dots, B_{n'/w'}$. Add $2k$ points to P on the main diagonal of this box, setting the labels of the k lowest (resp., highest) points to 1 (resp., -1); see Figure 6b. This finishes the construction of P .

The set P has $n = n' + 2k + 2ckn'$ points (all at distinct locations) and width $w = w' + 1$. Furthermore, the optimal monotone error k^* of P is k because (i) any monotone classifier must misclassify at least k points in the dummy box, and (ii) error k is attainable by the classifier that classifies all the points in $B_1, \dots, B_{n'/w'}$ correctly and maps all the points in the dummy box to 1.

Let us apply the given algorithm $\mathcal{A}$ on P and obtain its output classifier h . We argue that h must correctly classify every non-dummy point $p \in P$ (every such p originated from P'). Otherwise, suppose that $\text{label}(p) = -1$ but $h(p) = 1$ for some non-dummy $p \in P$. By our construction, p is dominated by at least ck points of label -1 . As h maps all those ck points to 1, we know $\text{err}_P(h) \geq ck + 1$, contradicting the fact that $\mathcal{A}$ guarantees an error at most ck^* . A symmetric argument rules out the possibility that $\text{label}(p) = 1$ but $h(p) = -1$. We can thus return h as an optimal classifier for P' . It follows from our earlier lower bound on $\mathbb{P}(n', w')$ that J_{LV} must be $\Omega(w' \text{Log } \frac{n'}{w'})$.

A Monte-Carlo Lower Bound. Again, let n', k', k, c, n , and w be as stated in Theorem 13. Let $\mathcal{A}$ be a randomized algorithm (with guessing power) that, when executed on an input P with size n , width w , and optimal monotone error $k^* = k$, always guarantees

- an expected error at most ck on P and
- an expected cost at most J_{MC} .

Next, we show $J_{MC} = \Omega(w' \text{Log } \frac{n'}{w'})$, which will complete the proof of Theorem 13.

With probability at least $1/2$, the algorithm $\mathcal{A}$ must (i) return a monotone classifier whose error on P is at most $4ck$ and (ii) probe at most $4J_{MC}$ points. Otherwise, one of the following two events must occur with probability at least $1/4$:

- $\mathcal{A}$ outputs a classifier whose error on P is over $4ck^*$, or
- $\mathcal{A}$ probes more than $4J_{MC}$ points of P .

However, this means that $\mathcal{A}$ either has an expected error higher than ck^* or incurs an expected cost higher than J_{MC} , contradicting its guarantees.

We deploy $\mathcal{A}$ as a black box to design a randomized algorithm that probes $O(J_{MC})$ points in expectation and *always* returns a monotone classifier with an error at most $4ck^*$ on P . For this purpose, run $\mathcal{A}$ until either it returns a monotone classifier h or has probed $4J_{MC}$ points. In the former situation, we ask the oracle whether $\text{err}_P(h) \leq 4ck^*$. If so, return h . In all other situations (i.e., the oracle answers “no” or $\mathcal{A}$ does not terminate after $4J_{MC}$ probes), we declare “failure” and start all over again. After having failed $\lceil \log_2 n \rceil$ times, we simply probe the entire P and return an optimal monotone classifier. Since each time we fail with probability at most $1/2$, the expected probing cost is bounded by

$$\left(\sum_{i=0}^{\lceil \log_2 n \rceil} 4J_{MC} \cdot (i+1) \left(\frac{1}{2}\right)^i \right) + \frac{1}{n} \cdot n = O(J_{MC})$$

noticing that the probability of failing $\lceil \log_2 n \rceil$ times is at most $1/n$. By our earlier Las Vegas lower bound, we conclude that $J_{MC} = \Omega(w' \text{Log } \frac{n'}{w'})$.

6 A Lower Bound for Arbitrary Approximation Ratios

We now continue our study on the hardness of approximation in the regime where ϵ can be arbitrarily small. The main result in this section is:

Theorem 14. *Let ϵ be an arbitrary value satisfying $0 < \epsilon \leq 1/10$. Fix any integers $n \geq 2$ and $w \geq 1$ such that n is a multiple of w , and $n \geq \frac{w}{\epsilon^2} \ln n$. Suppose that $\mathcal{A}$ is an algorithm for Problem 1 under $d = 2$ that guarantees an expected error of at most $(1 + \epsilon)k^*$ on every input of size n , where k^* is the optimal monotone error. Then, the expected cost of $\mathcal{A}$ must be $\Omega(w/\epsilon^2)$ on at least one input with width w , where $\Omega(\cdot)$ hides a constant that does not depend on ϵ , n , w , and k^* .*

The rest of the section serves as a proof of the theorem. Define

$$\gamma = 9\epsilon \quad (61)$$

$$\mu_1 = (1 - \gamma)/2 \quad (62)$$

$$\mu_2 = (1 + \gamma)/2 \quad (63)$$

$$M = (1 - \gamma^2)/(9\gamma^2). \quad (64)$$

Because $\gamma = 9\epsilon$ and $\epsilon \leq 1/10$, we have

$$\frac{wM}{8} = \frac{1 - 81\epsilon^2}{5832} \cdot \frac{w}{\epsilon^2} > \frac{1}{31000} \frac{w}{\epsilon^2}. \quad (65)$$

Suppose, for contradiction, that $\mathcal{A}$ has expected cost at most $wM/8$ on every input of size n and width w :

$\mathcal{A}$ = an algorithm with expected cost at most $wM/8$ on every input of size n and width w . (66)

Let $x_1, x_2, \dots, x_w$ be w distinct locations in $\mathbb{R}^2$ such that no location dominates another. Henceforth, we fix P to be a multiset of n points where

- exactly n/w points are placed at location x_i for each $i \in [w]$;
- every element of P has a distinct ID.

The elements in P do not carry labels yet. The random process below stochastically generates their labels and measures the cost and error of $\mathcal{A}$ over the resulting P :

RP-1

1. μ = a w -dimensional vector sampled from $\{\mu_1, \mu_2\}^w$ uniformly at random
2. **for** every element $p \in P$ **do** /* suppose that p is at location x_i */
3. assign p label 1 with probability $\mu[i]$ or -1 with probability $1 - \mu[i]$
4. $h_{\mathcal{A}}$ = the monotone classifier output by algorithm $\mathcal{A}$ when executed on P ;
 Λ_1 = the number of probes $\mathcal{A}$ performed
5. $R_1 = \text{err}_P(h_{\mathcal{A}})$
6. **return** (Λ_1, R_1)

The optimal monotone error k^* is a random variable determined by the labeled multiset P . By (66), we have $\mathbf{E}[\Lambda_1] \leq wM/8$. Our objective is to show that the expected error of $\mathcal{A}$ is too large. We do so by relating **RP-1** to another random process:

RP-2

1. μ = a w -dimensional vector sampled from $\{\mu_1, \mu_2\}^w$ uniformly at random
 /* now run $\mathcal{A}$ on P */
2. **while** algorithm $\mathcal{A}$ still needs to perform a probe **do**
3. p = the element probed by $\mathcal{A}$ (identified by ID) /* suppose that p is at location x_i */

4. assign p label 1 with probability $\mu[i]$ or -1 with probability $1 - \mu[i]$
5. $h_{\mathcal{A}}$ = the monotone classifier output by $\mathcal{A}$;
 Λ_2 = the number of probes $\mathcal{A}$ performed
6. **for** every element $q \in P$ that has not been probed by $\mathcal{A}$ **do** /* suppose that q is at x_i */
7. assign q label 1 with probability $\mu[i]$ or -1 with probability $1 - \mu[i]$
8. $R_2 = \text{err}_P(h_{\mathcal{A}})$
9. **return** (Λ_2, R_2)

Lemma 15. $\mathbf{E}[\Lambda_1] = \mathbf{E}[\Lambda_2]$ and $\mathbf{E}[R_1] = \mathbf{E}[R_2]$.

Proof. We prove only $\mathbf{E}[R_1] = \mathbf{E}[R_2]$ because an analogous (and simpler) argument shows $\mathbf{E}[\Lambda_1] = \mathbf{E}[\Lambda_2]$. Let us first consider that $\mathcal{A}$ is a deterministic algorithm, i.e., a binary decision tree $\mathcal{T}$. Recall that each internal node of $\mathcal{T}$ is associated with an element (identified by ID) in P that should be probed when $\mathcal{A}$ is at this node. Each leaf of $\mathcal{T}$ is associated with a classifier that should be returned when $\mathcal{A}$ is at this node. For each leaf node v , denote by π_v the path from the root of $\mathcal{T}$ to v .

We have for each $j \in \{1, 2\}$:

$$\mathbf{E}[R_j] = \sum_{\text{leaf } v \text{ of } \mathcal{T}} \Pr[\mathcal{A} \text{ finishes at } v \text{ in RP-}j] \cdot \mathbf{E}[R_j \mid \mathcal{A} \text{ finishes at } v \text{ in RP-}j].$$

Let us concentrate on an arbitrary leaf v . Let $p_1, p_2, \dots, p_t$ be the elements associated with the internal nodes of π_v in the top-down order. The algorithm $\mathcal{A}$ arrives at v if and only if each element p_i ($i \in [t]$) takes a specific label, denoted as l_i . The probability of the event “ $\text{label}(p_i) = l_i$ for all $i \in [t]$ ” is identical in RP-1 and RP-2. Hence, $\mathcal{A}$ has the same probability of reaching v in each random process. Conditioned on the aforementioned event, the classifier $h_{\mathcal{A}}$ is fixed, while the labels of the elements in $P \setminus \{p_1, \dots, p_t\}$ have the same distribution in the two random processes. Therefore, the conditional expectations of R_1 and R_2 are identical. It follows that $\mathbf{E}[R_1] = \mathbf{E}[R_2]$.

As a randomized algorithm is a distribution over a family of deterministic algorithms, the equality also holds for a randomized $\mathcal{A}$. $\square$

Because $\mathbf{E}[\Lambda_1] \leq wM/8$, Lemma 15 gives

$$\mathbf{E}[\Lambda_2] \leq wM/8. \tag{67}$$

The labeled multiset P , and hence k^* , has the same distribution in RP-1 and RP-2. The next subsection proves:

Lemma 16. When (67) holds, $n \geq 2$, $\epsilon \leq 1/10$, and $n \geq \frac{w}{\epsilon^2} \ln n$, we must have

$$\mathbf{E}[R_2] > (1 + \epsilon) \mathbf{E}[k^*].$$

We can now complete the proof of Theorem 14. If $\mathcal{A}$ guarantees an expected error of at most $(1 + \epsilon)k^*$ on every labeled input, then, after conditioning on the labeled multiset P generated by RP-1, we have

$$\mathbf{E}[R_1 \mid P] \leq (1 + \epsilon)k^*.$$

Taking expectation over P gives $\mathbf{E}[R_1] \leq (1 + \epsilon) \mathbf{E}[k^*]$. This contradicts Lemma 15 and Lemma 16. Hence, $\mathcal{A}$ must have expected cost greater than $wM/8$ on at least one input. The theorem now follows from (65).

6.1 Proof of Lemma 16

Our proof is inspired by a lower bound argument in [7] for non-realizable active learning. We need the following expected-sample testing bound, whose proof is deferred to the end of this subsection.

Lemma 17. *Define μ to be a random variable that equals μ_1 or μ_2 — see (62) and (63) — each with probability $1/2$. Consider an algorithm that adaptively draws i.i.d. samples from $\{-1, 1\}$, where each sample equals 1 with probability μ , and eventually outputs a guess from $\{\mu_1, \mu_2\}$. Let X be its number of samples. If $\mathbf{E}[X] < M$, where the expectation is over μ , the samples, and the internal randomness of the algorithm, then its guess is wrong with probability greater than $1/3$.*

The subsequent discussion is carried out under the conditions stated in Lemma 16. For each $i \in [w]$, let M_i be the number of points at x_i probed by $\mathcal{A}$ in RP-2. We say that i is *light* if $\mathbf{E}[M_i] < M$ or *heavy* otherwise.

Lemma 18. *At least $7w/8$ values of i are light.*

Proof. Every heavy i satisfies $\mathbf{E}[M_i] \geq M$. Because

$$\sum_{i=1}^w \mathbf{E}[M_i] = \mathbf{E}[\Lambda_2] \leq \frac{wM}{8},$$

the number of heavy values is at most $w/8$. The lemma follows. $\square$

Consider the classifier $h_{\mathcal{A}}$ output by $\mathcal{A}$ at Line 5 of RP-2. For each $i \in [w]$, set $K_i = 1$ if one of the following events occurs:

- $h_{\mathcal{A}}(x_i) = 1$ and $\mu[i] = \mu_1$;
- $h_{\mathcal{A}}(x_i) = -1$ and $\mu[i] = \mu_2$.

Otherwise, set $K_i = 0$.

Lemma 19. *If $i \in [w]$ is light, then $\mathbf{E}[K_i] > 1/3$.*

Proof. We interpret $h_{\mathcal{A}}(x_i)$ as a guess about $\mu[i]$: the value 1 represents the guess $\mu[i] = \mu_2$, while the value -1 represents the guess $\mu[i] = \mu_1$. Thus, $K_i = 1$ if and only if the guess is wrong.

Conditional on $\mu[i]$, the labels revealed by the probes at x_i are i.i.d. samples of the form considered in Lemma 17. Because i is light, $\mathbf{E}[M_i] < M$. Lemma 17 gives $\Pr[K_i = 1] > 1/3$, proving the lemma. $\square$

Recall that RP-2 randomly chooses a vector μ at Line 1. For each $i \in [w]$, define its *good i -label* to be

- 1 if $\mu[i] = \mu_2$;
- -1 otherwise.

The label in $\{-1, 1\}$ different from the good i -label is called the *bad i -label*. At x_i , the expected number of points receiving the good i -label is $\frac{n}{w}(\frac{1}{2} + \frac{\gamma}{2})$. We say that the labeled multiset P created by RP-2 is *intended* if, for every $i \in [w]$, at least

$$\frac{n}{w} \left(\frac{1}{2} + \frac{\gamma}{4} \right)$$

points at location x_i receive the good i -label.

Lemma 20. *The probability for P to be intended is at least $1 - 1/n^4$.*

Proof. Focus on an arbitrary $i \in [w]$, and let B_i be the number of points at x_i receiving the bad i -label. Then

$$\mathbf{E}[B_i] = \frac{n}{w} \left(\frac{1}{2} - \frac{\gamma}{2} \right).$$

Set $t = \frac{\gamma}{2(1-\gamma)}$, for which

$$(1+t) \mathbf{E}[B_i] = \frac{n}{w} \left(\frac{1}{2} - \frac{\gamma}{4} \right).$$

By Chernoff bound (74),

$$\begin{aligned} \Pr \left[B_i \geq \frac{n}{w} \left(\frac{1}{2} - \frac{\gamma}{4} \right) \right] &\leq \exp \left(-\frac{t^2}{2+t} \mathbf{E}[B_i] \right) \\ &\leq \exp \left(-\frac{\gamma^2 n}{16w} \right) \\ (\text{by (61)}) &= \exp \left(-\frac{81\epsilon^2 n}{16w} \right). \end{aligned}$$

This probability is at most $1/n^5$ because $n \geq \frac{w}{\epsilon^2} \ln n$. A union bound over the $w \leq n$ locations shows that P is intended with probability at least $1 - 1/n^4$. $\square$

Lemma 21.

$$\mathbf{E} \left[\mathbb{1}_{P \text{ intended}} \sum_{i=1}^w K_i \right] > w \left(\frac{7}{24} - \frac{1}{n^4} \right).$$

Proof. Lemmas 18 and 19 imply

$$\mathbf{E} \left[\sum_{i=1}^w K_i \right] > \frac{7w}{8} \cdot \frac{1}{3} = \frac{7w}{24}.$$

On the other hand, $\sum_{i=1}^w K_i \leq w$, while Lemma 20 gives $\Pr[P \text{ not intended}] \leq 1/n^4$. Therefore,

$$\begin{aligned} \mathbf{E} \left[\mathbb{1}_{P \text{ intended}} \sum_{i=1}^w K_i \right] &\geq \mathbf{E} \left[\sum_{i=1}^w K_i \right] - w \cdot \Pr[P \text{ not intended}] \\ &> w \cdot \left(\frac{7}{24} - \frac{1}{n^4} \right) \end{aligned}$$

as claimed. $\square$

For every labeled multiset P supported at $x_1, \dots, x_w$, a monotone classifier can choose the majority label independently at every location because the locations are mutually incomparable. Hence,

$$k^* \leq n/2. \tag{68}$$

Suppose that P is intended. For each $i \in [w]$, let I_i be the number of points at x_i receiving the good i -label. An optimal classifier h^* maps x_i to the good i -label, and

$$I_i \geq \frac{n}{w} \left(\frac{1}{2} + \frac{\gamma}{4} \right).$$

By the definition of K_i , the classifiers h^* and $h_{\mathcal{A}}$ disagree at x_i if and only if $K_i = 1$. Such a disagreement causes $h_{\mathcal{A}}$ to make $2I_i - n/w$ more errors than h^* at x_i . Consequently, whenever P is intended,

$$R_2 - k^* = \text{err}_P(h_{\mathcal{A}}) - k^* = \sum_{i=1}^w K_i \left(2I_i - \frac{n}{w} \right) \geq \frac{n\gamma}{2w} \sum_{i=1}^w K_i. \quad (69)$$

Because $R_2 \geq k^*$ always, Lemma 21 and (69) yield

$$\begin{aligned} \mathbf{E}[R_2] - \mathbf{E}[k^*] &= \mathbf{E}[R_2 - k^*] \\ &\geq \mathbf{E}[\mathbb{1}_{P \text{ intended}}(R_2 - k^*)] \\ &\geq \frac{n\gamma}{2w} \mathbf{E} \left[\mathbb{1}_{P \text{ intended}} \sum_{i=1}^w K_i \right] \\ &> \frac{n\gamma}{2} \left(\frac{7}{24} - \frac{1}{n^4} \right) \\ (\text{by } n \geq 2 \text{ and (61)}) &\geq \frac{99\epsilon n}{96} \\ &> \frac{\epsilon n}{2} \\ (\text{by (68)}) &\geq \epsilon \mathbf{E}[k^*]. \end{aligned}$$

It follows that $\mathbf{E}[R_2] > (1 + \epsilon) \mathbf{E}[k^*]$, completing the proof of Lemma 16.

Proof of Lemma 17. For any $p \in [0, 1]$, let $\text{Bern}(p)$ denote the Bernoulli distribution over $\{-1, 1\}$ that assigns probability p to 1 and probability $1 - p$ to -1 . Define

$$d_{\mu_1, \mu_2} = D_{KL}(\text{Bern}(\mu_1) \parallel \text{Bern}(\mu_2)) = \gamma \ln \frac{1 + \gamma}{1 - \gamma}$$

where D_{KL} represents KL divergence. It is easy to verify

$$d_{\mu_1, \mu_2} = D_{KL}(\text{Bern}(\mu_2) \parallel \text{Bern}(\mu_1)) = \gamma \ln \frac{1 + \gamma}{1 - \gamma}. \quad (70)$$

Define a *transcript* as the random bits consumed by the algorithm and the sequence of samples drawn. We will view a transcript as a sequence: $\Sigma_0, (\text{sample } 1, \Sigma_1), (\text{sample } 2, \Sigma_2), \dots, (\text{sample } X, \Sigma_X)$ where, for each $0 \leq i \leq X - 1$, Σ_i is the random bit sequence consumed by the algorithm between samples i and $i + 1$, and Σ_X is the random bit sequence consumed after the last sample. The output of the algorithm is a function of the transcript.

Let $\mathcal{T}$ be the space of all possible transcripts. For $j \in \{1, 2\}$, define the probability distribution $\mathcal{Q}_j$ over $\mathcal{T}$ by

$$\mathcal{Q}_j(\mathcal{T}) = \mathbf{Pr}[\text{the transcript belongs to } \mathcal{T} \mid \mu = \mu_j] \quad \text{for every } \mathcal{T} \subseteq \mathcal{T}.$$

For each $T \in \mathcal{T}$, we abbreviate $\mathcal{Q}_j(\{T\})$ as $\mathcal{Q}_j(T)$. The total-variation distance of $\mathcal{Q}_1$ and $\mathcal{Q}_2$ is

$$d_{TV}(\mathcal{Q}_1, \mathcal{Q}_2) = \sup_{\text{events } \mathcal{T} \subseteq \mathcal{T}} |\mathcal{Q}_1(\mathcal{T}) - \mathcal{Q}_2(\mathcal{T})|. \quad (71)$$

Lemma 22. $D_{KL}(\mathcal{Q}_1\|\mathcal{Q}_2) = d_{\mu_1, \mu_2} \cdot \mathbf{E}[X \mid \mu = \mu_1]$.

Proof. Conceptually, generate an infinite sequence $Y_1, Y_2, \dots$ of i.i.d. samples before running the algorithm. The algorithm observes only the first X samples. For $j \in \{1, 2\}$, define

$$\varphi_j(1) = \mu_j \quad \text{and} \quad \varphi_j(-1) = 1 - \mu_j.$$

Thus, $\varphi_j(y)$ is the probability of obtaining the sample value y when $\mu = \mu_j$.

Consider an arbitrary transcript $T \in \mathcal{T}$ written as

$$T = \Sigma_0, (y_1, \Sigma_1), \dots, (y_x, \Sigma_x),$$

where x is the number of samples in T and $y_i \in \{-1, 1\}$ is the i -th sample. Let $b(T)$ be the total number of random bits in $\Sigma_0, \dots, \Sigma_x$; each such bit is fair and independent of μ . Furthermore, after the random bits and sample values in T have been fixed, every decision made by the algorithm — including whether to draw another sample or to stop — is deterministic. It follows that

$$\mathcal{Q}_j(T) = 2^{-b(T)} \prod_{i=1}^x \varphi_j(y_i).$$

Hence,

$$\ln \frac{\mathcal{Q}_1(T)}{\mathcal{Q}_2(T)} = \sum_{i=1}^x \ln \frac{\varphi_1(y_i)}{\varphi_2(y_i)}.$$

For every fixed transcript $T \in \mathcal{T}$, let $x(T)$ be its number of samples and let $y_i(T)$ be its i -th sample. Let T^* denote the random transcript generated by the algorithm. By the definition of $\mathcal{Q}_1$,

$$\mathcal{Q}_1(T) = \mathbf{Pr}[T^* = T \mid \mu = \mu_1].$$

Furthermore, $x(T^*) = X$ and $y_i(T^*) = Y_i$. Using the definition of KL divergence, we obtain

$$\begin{aligned} D_{KL}(\mathcal{Q}_1\|\mathcal{Q}_2) &= \sum_{T \in \mathcal{T}} \mathcal{Q}_1(T) \ln \frac{\mathcal{Q}_1(T)}{\mathcal{Q}_2(T)} \\ &= \sum_{T \in \mathcal{T}} \mathbf{Pr}[T^* = T \mid \mu = \mu_1] \sum_{i=1}^{x(T)} \ln \frac{\varphi_1(y_i(T))}{\varphi_2(y_i(T))} \\ &= \mathbf{E} \left[\sum_{i=1}^{x(T^*)} \ln \frac{\varphi_1(y_i(T^*))}{\varphi_2(y_i(T^*))} \mid \mu = \mu_1 \right] \\ &= \mathbf{E} \left[\sum_{i=1}^X \ln \frac{\varphi_1(Y_i)}{\varphi_2(Y_i)} \mid \mu = \mu_1 \right]. \end{aligned} \tag{72}$$

For each $i \geq 1$, the event $X \geq i$ is determined by the internal randomness of the algorithm and the samples $Y_1, \dots, Y_{i-1}$. Hence, conditioned on $\mu = \mu_1$, this event is independent of Y_i , which has distribution $\text{Bern}(\mu_1)$. Therefore,

$$\mathbf{E} \left[\mathbb{1}_{X \geq i} \cdot \ln \frac{\varphi_1(Y_i)}{\varphi_2(Y_i)} \mid \mu = \mu_1 \right] = \mathbf{Pr}[X \geq i \mid \mu = \mu_1] \sum_{y \in \{-1, 1\}} \varphi_1(y) \ln \frac{\varphi_1(y)}{\varphi_2(y)}$$

$$= \Pr[X \geq i \mid \mu = \mu_1] \cdot d_{\mu_1, \mu_2}.$$

Under the premise of Lemma 17, $\mathbf{E}[X] < M$ and hence $\mathbf{E}[X \mid \mu = \mu_1] \leq 2\mathbf{E}[X] < \infty$. Furthermore, $\ln(\varphi_1(Y_i)/\varphi_2(Y_i))$ takes only two finite values. Fubini's theorem³ therefore permits exchanging the expectation and summation in (72). We can thus derive

$$\begin{aligned} D_{KL}(\mathcal{Q}_1 \parallel \mathcal{Q}_2) &= \sum_{i \geq 1} \mathbf{E} \left[\mathbb{1}_{X \geq i} \cdot \ln \frac{\varphi_1(Y_i)}{\varphi_2(Y_i)} \mid \mu = \mu_1 \right] \\ &= d_{\mu_1, \mu_2} \sum_{i \geq 1} \Pr[X \geq i \mid \mu = \mu_1] \\ &= d_{\mu_1, \mu_2} \cdot \mathbf{E}[X \mid \mu = \mu_1] \end{aligned}$$

where the last equality used the tail-sum identity. $\square$

The same argument with the roles of μ_1 and μ_2 exchanged gives

$$D_{KL}(\mathcal{Q}_2 \parallel \mathcal{Q}_1) = d_{\mu_1, \mu_2} \cdot \mathbf{E}[X \mid \mu = \mu_2].$$

Suppose that the algorithm is wrong with probability at most $1/3$. Let $\mathcal{E} \subseteq \mathcal{T}$ be the event consisting of the transcripts under which the algorithm outputs μ_1 , and let $\mathcal{E}^c = \mathcal{T} \setminus \mathcal{E}$. Under the uniform prior on $\{\mu_1, \mu_2\}$, the error probability of the algorithm is

$$\frac{1}{2} (\mathcal{Q}_1(\mathcal{E}^c) + \mathcal{Q}_2(\mathcal{E})) = \frac{1}{2} (1 - (\mathcal{Q}_1(\mathcal{E}) - \mathcal{Q}_2(\mathcal{E}))).$$

From (71), we know $\mathcal{Q}_1(\mathcal{E}) - \mathcal{Q}_2(\mathcal{E}) \leq d_{TV}(\mathcal{Q}_1, \mathcal{Q}_2)$. Hence, the error probability is at least $\frac{1}{2}(1 - d_{TV}(\mathcal{Q}_1, \mathcal{Q}_2))$. Since it is at most $1/3$, we must have $d_{TV}(\mathcal{Q}_1, \mathcal{Q}_2) \geq 1/3$. By Pinsker's inequality⁴,

$$d_{\mu_1, \mu_2} \cdot \mathbf{E}[X] = \frac{D_{KL}(\mathcal{Q}_1 \parallel \mathcal{Q}_2) + D_{KL}(\mathcal{Q}_2 \parallel \mathcal{Q}_1)}{2} \geq 2d_{TV}(\mathcal{Q}_1, \mathcal{Q}_2)^2 \geq \frac{2}{9}.$$

Finally,

$$\ln \frac{1+\gamma}{1-\gamma} = 2 \int_0^\gamma \frac{1}{1-z^2} dz \leq \frac{2\gamma}{1-\gamma^2},$$

and hence (70) gives $d_{\mu_1, \mu_2} \leq 2\gamma^2/(1-\gamma^2)$. We conclude that

$$\mathbf{E}[X] \geq \frac{2}{9d_{\mu_1, \mu_2}} \geq \frac{1-\gamma^2}{9\gamma^2} = M$$

where the last step used (64). This contradicts the fact that $\mathbf{E}[X] < M$.

³Fubini's theorem states that, for any sequence of random variables $Z_1, Z_2, \dots$ satisfying $\sum_{i \geq 1} \mathbf{E}[|Z_i|] < \infty$, we have $\mathbf{E}[\sum_{i \geq 1} Z_i] = \sum_{i \geq 1} \mathbf{E}[Z_i]$.

⁴Here, D_{KL} is defined using the natural logarithm. Under this convention, Pinsker's inequality states that $d_{TV}(\mathcal{P}, \mathcal{Q}) \leq \sqrt{D_{KL}(\mathcal{P} \parallel \mathcal{Q})/2}$ for any two probability distributions $\mathcal{P}$ and $\mathcal{Q}$.

7 Conclusions

This article has provided a comprehensive study of monotone classification in $\mathbb{R}^d$ with relative approximation guarantees, where the objective is to minimize the label-probing cost while finding a classifier whose error can be higher than the optimal monotone error k^* by at most a $1 + \epsilon$ multiplicative factor. Our findings delineate the complexity landscape across the spectrum of ϵ . For the exact case ($\epsilon = 0$), we established a lower bound of $\Omega(n)$ probes even in 1D space, where n is the size of the input P , underscoring the hardness of achieving optimality. In the approximate regime ($\epsilon > 0$), we introduced two algorithms: the simple RPE algorithm, which achieves an expected error of at most $2k^*$ with $O(w \log \frac{n}{w})$ probes where w is the width of P , and an algorithm powered by a new “relative-comparison coreset” technique, which ensures $(1 + \epsilon)k^*$ error w.h.p. at a cost of $O(\frac{w}{\epsilon^2} \log \frac{n}{w} \cdot \log n)$. These are complemented by lower bounds of $\Omega(w \log \frac{n}{(1+k^*)w})$ for constant $\epsilon \geq 1$ and $\Omega(w/\epsilon^2)$ for arbitrary $\epsilon > 0$, demonstrating that our algorithms are near-optimal asymptotically.

For future work, it would be an intriguing challenge to shave off an $O(\log n)$ factor in the cost of our coreset-based algorithm. Equally challenging would be the task of proving a lower bound that grows strictly faster than $\Omega(w/\epsilon^2)$ for arbitrary $\epsilon > 0$.

References

- [1] M. Alishahi and J. M. Phillips. No dimensional sampling coresets for classification. In *Proceedings of International Conference on Machine Learning (ICML)*, 2024.
- [2] S. Angelov, B. Harb, S. Kannan, and L. Wang. Weighted isotonic regression under the L_1 norm. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 783–791. ACM Press, 2006.
- [3] A. Arasu, M. Götz, and R. Kaushik. On active learning of record matching packages. In *Proceedings of ACM Management of Data (SIGMOD)*, pages 783–794, 2010.
- [4] O. Bachem, M. Lucic, and A. Krause. Scalable k-means clustering via lightweight coresets. pages 1119–1127, 2018.
- [5] M. Balcan, A. Beygelzimer, and J. Langford. Agnostic active learning. *Journal of Computer and System Sciences (JCSS)*, 75(1):78–89, 2009.
- [6] K. Bellare, S. Iyengar, A. G. Parameswaran, and V. Rastogi. Active sampling for entity matching with guarantees. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 7(3):12:1–12:24, 2013.
- [7] A. Beygelzimer, S. Dasgupta, and J. Langford. Importance weighted active learning. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 49–56, 2009.
- [8] V. Braverman, D. Feldman, H. Lang, A. Statman, and S. Zhou. Efficient coreset constructions via sensitivity sampling. In *Asian Conference on Machine Learning (ACML)*, volume 157, pages 948–963, 2021.
- [9] N. H. Bshouty and C. Tamon. On the fourier spectrum of monotone functions. *Journal of the ACM (JACM)*, 43(4):747–770, 1996.

- [10] D. Chakrabarty and C. Seshadhri. Optimal bounds for monotonicity and lipschitz testing over hypercubes and hypergrids. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 419–428, 2013.
- [11] D. Chakrabarty and C. Seshadhri. An optimal lower bound for monotonicity testing over hypergrids. *Theory of Computing*, 10:453–464, 2014.
- [12] K. Chen. On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM Journal of Computing*, 39(3):923–947, 2009.
- [13] X. Chen, A. De, Y. Huang, Y. Li, S. Nadimpalli, R. A. Servedio, and T. Yang. Relative-error monotonicity testing. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 373–402, 2025.
- [14] P. Christen, D. Vatsalan, and Q. Wang. Efficient entity resolution with adaptive and interactive training data selection. In *Proceedings of International Conference on Management of Data (ICDM)*, pages 727–732, 2015.
- [15] X. Chu, I. F. Ilyas, and P. Koutris. Distributed data deduplication. *Proceedings of the VLDB Endowment (PVLDB)*, 9(11):864–875, 2016.
- [16] A. Dasgupta, P. Drineas, B. Harb, R. Kumar, and M. W. Mahoney. Sampling algorithms and coresets for ℓ_p regression. *SIAM Journal of Computing*, 38(5):2060–2078, 2009.
- [17] S. Dasgupta. Coarse sample complexity bounds for active learning. In *Proceedings of Neural Information Processing Systems (NIPS)*, pages 235–242, 2005.
- [18] S. Dasgupta, D. J. Hsu, and C. Monteleoni. A general agnostic active learning algorithm. In *Proceedings of Neural Information Processing Systems (NIPS)*, pages 353–360, 2007.
- [19] R. P. Dilworth. A decomposition theorem for partially ordered sets. *The Annals of Mathematics*, 51(1):161–166, 1950.
- [20] V. Efthymiou, G. Papadakis, G. Papastefanatos, K. Stefanidis, and T. Palpanas. Parallel meta-blocking for scaling entity resolution over big heterogeneous data. *Information Systems*, 65:137–157, 2017.
- [21] D. Feldman and M. Langberg. A unified framework for approximating and clustering data. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 569–578, 2011.
- [22] D. Feldman, M. Schmidt, and C. Sohler. Turning big data into tiny data: Constant-size coresets for k-means, pca, and projective clustering. *SIAM Journal of Computing*, 49(3):601–657, 2020.
- [23] E. Fischer, E. Lehman, I. Newman, S. Raskhodnikova, R. Rubinfeld, and A. Samorodnitsky. Monotonicity testing over general poset domains. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*, pages 474–483, 2002.
- [24] C. Gokhale, S. Das, A. Doan, J. F. Naughton, N. Rampalli, J. W. Shavlik, and X. Zhu. Corleone: hands-off crowdsourcing for entity matching. In *Proceedings of ACM Management of Data (SIGMOD)*, pages 601–612, 2014.
- [25] O. Goldreich, S. Goldwasser, E. Lehman, and D. Ron. Testing monotonicity. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 426–435, 1998.

- [26] O. Goldreich, S. Goldwasser, E. Lehman, D. Ron, and A. Samorodnitsky. Testing monotonicity. *Combinatorica*, 20(3):301–337, 2000.
- [27] S. Hanneke. A bound on the label complexity of agnostic active learning. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 353–360, 2007.
- [28] S. Hanneke. Teaching dimension and the complexity of active learning. In *Proceedings of Conference on Learning Theory (COLT)*, pages 66–81, 2007.
- [29] S. Hanneke. Theory of disagreement-based active learning. *Foundations and Trends in Machine Learning*, 7(2-3):131–309, 2014.
- [30] S. Hanneke. Agnostic active learning is always better than passive learning. In *Proceedings of Neural Information Processing Systems (NIPS)*, 2025.
- [31] S. Hanneke and L. Yang. Minimax analysis of active learning. *Journal of Machine Learning Research (JMLR)*, 16:3487–3602, 2015.
- [32] S. Har-Peled and A. Kushal. Smaller coresets for k-median and k-means clustering. *Discrete & Computational Geometry*, 37(1):3–19, 2007.
- [33] S. Har-Peled and M. Sharir. Relative (p, ϵ) -approximations in geometry. *Discrete & Computational Geometry*, 45(3):462–496, 2011.
- [34] M. Kääriäinen. Active learning in the non-realizable case. In *Proceedings of International Conference on Algorithmic Learning Theory (ALT)*, pages 63–77, 2006.
- [35] H. Köpcke, A. Thor, and E. Rahm. Evaluation of entity resolution approaches on real-world match problems. *Proceedings of the VLDB Endowment (PVLDB)*, 3(1):484–493, 2010.
- [36] J. Lange, R. Rubinfeld, and A. Vasilyan. Properly learning monotone functions via local correction. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 75–86, 2022.
- [37] J. Lange and A. Vasilyan. Agnostic proper learning of monotone functions: beyond the black-box correction barrier. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1149–1170, 2023.
- [38] T. Mai, C. Musco, and A. Rao. Coresets for classification - simplified and strengthened. In *Proceedings of Neural Information Processing Systems (NIPS)*, pages 11643–11654, 2021.
- [39] V. V. Meduri, L. Popa, P. Sen, and M. Sarwat. A comprehensive benchmark framework for active learning methods in entity matching. In *Proceedings of ACM Management of Data (SIGMOD)*, pages 1133–1147, 2020.
- [40] S. Meintrap, A. Munteanu, and D. Rohde. Random projections and sampling algorithms for clustering of high-dimensional polygonal curves. In *Proceedings of Neural Information Processing Systems (NIPS)*, pages 12807–12817, 2019.
- [41] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
- [42] A. Munteanu, C. Schwiegelshohn, C. Sohler, and D. P. Woodruff. On coresets for logistic regression. In *Proceedings of Neural Information Processing Systems (NIPS)*, pages 6562–6571, 2018.

- [43] J. M. Phillips. *Coresets and Sketches*, chapter 49. CRC Press, 2016.
- [44] A. Samadian, K. Pruhs, B. Moseley, S. Im, and R. R. Curtin. Unconditional coresets for regularized loss minimization. In *Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 482–492, 2020.
- [45] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *Proceedings of ACM Knowledge Discovery and Data Mining (SIGKDD)*, pages 269–278, 2002.
- [46] B. Settles. Active learning literature survey. *Technical Report, University of Wisconsin-Madison*, 2010.
- [47] Q. F. Stout. Isotonic regression via partitioning. *Algorithmica*, 66(1):93–112, 2013.
- [48] Y. Tao. Entity matching with active monotone classification. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 49–62, 2018.
- [49] Y. Tao and Y. Wang. New algorithms for monotone classification. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*, pages 260–272, 2021.
- [50] A. Thor and E. Rahm. MOMA - A mapping-based object matching system. In *Proceedings of Biennial Conference on Innovative Data Systems Research (CIDR)*, pages 247–258, 2007.
- [51] E. Tolochinsky, I. Jubran, and D. Feldman. Generic coreset for scalable learning of monotonic kernels: Logistic regression, sigmoid and more. In *Proceedings of International Conference on Machine Learning (ICML)*, volume 162, pages 21520–21547, 2022.
- [52] L. Wang. Smoothness, disagreement coefficient, and the label complexity of agnostic active learning. *Journal of Machine Learning Research (JMLR)*, 12:2269–2292, 2011.

Appendix

A Concentration Bounds

Let $X_1, X_2, \dots, X_t$ be t independent Bernoulli random variables with $\Pr[X_i = 1] = \mu$ for each $i \in [t]$ (and hence $\Pr[X_i = 0] = 1 - \mu$). The following are two standard forms of Chernoff bounds [41]:

- for any $\gamma \in (0, 1]$:

$$\Pr \left[\left| \mu - \frac{1}{t} \sum_{i=1}^t X_i \right| \geq \gamma \mu \right] \leq 2 \exp \left(-\frac{\gamma^2 t \mu}{3} \right); \quad (73)$$

- for any $\gamma \geq 0$:

$$\Pr \left[\frac{1}{t} \sum_{i=1}^t X_i \geq (1 + \gamma) \mu \right] \leq \exp \left(-\frac{\gamma^2}{2 + \gamma} t \mu \right). \quad (74)$$

Lemma 23. For any $\phi \in (0, 1]$ and $\delta \in (0, 1]$, it holds that

$$\Pr \left[\left| \mu - \frac{1}{t} \sum_{i=1}^t X_i \right| \geq \phi \right] \leq \delta \quad (75)$$

as long as $t \geq \lceil \max\{\frac{\mu}{\phi^2}, \frac{1}{\phi}\} \cdot 3 \ln \frac{2}{\delta} \rceil$.

Proof. If $\mu \geq \phi$, we can derive

$$\begin{aligned} \Pr \left[\left| \mu - \frac{1}{t} \sum_{i=1}^t X_i \right| \geq \phi \right] &= \Pr \left[\left| \mu - \frac{1}{t} \sum_{i=1}^t X_i \right| \geq \frac{\phi}{\mu} \cdot \mu \right] \\ &\text{(by (73))} \leq 2 \exp \left(-\frac{1}{3} \left(\frac{\phi}{\mu} \right)^2 t \mu \right) \end{aligned}$$

which is at most δ when $t = \lceil \frac{3\mu}{\phi^2} \ln \frac{2}{\delta} \rceil$.

If $\mu < \phi$, we can derive

$$\begin{aligned} &\Pr \left[\left| \mu - \frac{1}{t} \sum_{i=1}^t X_i \right| \geq \phi \right] \\ &= \Pr \left[\frac{1}{t} \sum_{i=1}^t X_i \geq \phi + \mu \right] = \Pr \left[\frac{1}{t} \sum_{i=1}^t X_i \geq \left(1 + \frac{\phi}{\mu} \right) \mu \right] \\ &\text{(by (74))} \leq \exp \left(-\frac{(\phi/\mu)^2}{2 + \phi/\mu} \cdot t \mu \right) = \exp \left(-\frac{t \phi^2}{2\mu + \phi} \right) \end{aligned}$$

which is at most δ when $t = \lceil \frac{2\mu + \phi}{\phi^2} \ln \frac{1}{\delta} \rceil \leq \lceil \frac{3}{\phi} \ln \frac{1}{\delta} \rceil$. $\square$

Consider the setting of Problem 1. Suppose that we want to estimate the number x of elements in the input P satisfying an arbitrary predicate Q . We can draw with replacement a set S of $t = O(\frac{1}{\phi^2} \log \frac{1}{\delta})$ elements from P uniformly at random. If y is the number of elements in S satisfying Q , Lemma 23 assures us that $(y/t) \cdot n$ approximates x up to absolute error ϕn with probability at least $1 - \delta$. As a corollary, given any $h \in \mathbb{H}_{mon}$, we can utilize the aforementioned S to estimate $err_P(h)$ — defined in (1) — up to absolute error ϕn by formulating Q as follows: an element $p \in P$ satisfies Q if and only if $label(p) \neq h(p)$.

B VC-Dimension, Disagreement Coefficient, Star Number, and Teaching Dimension of Monotone Classifiers

The input of Problem 1 is a multiset P of points in $\mathbb{R}^d$ where each element $p \in P$ has a label from $\{-1, 1\}$. This section discusses the VC-dimension λ , disagreement coefficient θ , star number $\mathfrak{s}$, and teaching dimension χ of $\mathbb{H}_{mon}$ (the set of monotone classifiers) on P . These quantities are needed to understand the limitations of the existing active learning algorithms. We will show that all four quantities are $\Omega(w)$ in the worst case, where w is the width of P .

VC-Dimension. The VC-dimension λ of $\mathbb{H}_{mon}$ on P is the size of the largest subset $S \subseteq P$ that can be *shattered* by $\mathbb{H}_{mon}$; that is, for any function $f : S \rightarrow \{-1, 1\}$, there exists a classifier $h \in \mathbb{H}_{mon}$ such that $h(p) = f(p)$ for every $p \in S$.

We show that λ is at least the width w of P . Identify w points $p_1, p_2, \dots, p_w$ of P such that no point dominates another (such points must exist due to Dilworth's Theorem [19]). The set of these points can be shattered by $\mathbb{H}_{mon}$. Specifically, for any function $f : S \rightarrow \{-1, 1\}$, we can find a classifier $h \in \mathbb{H}_{mon}$ such that $\forall i \in [w]$

- if $f(p_i) = 1$, then $h(q) = 1$ for any point $q \succeq p_i$;
- if $f(p_i) = -1$, then $h(q) = 1$ for any point $p_i \succeq q$.

The points $p_1, p_2, \dots, p_w$ serve as evidence that $\lambda \geq w$.

Disagreement Coefficient. Given a set $\mathcal{H} \subseteq \mathbb{H}_{\text{mon}}$ of monotone classifiers, the *disagreement region* of $\mathcal{H}$ — denoted as $\text{DIS}(\mathcal{H})$ — is the set of elements $p \in P$ such that $h_1(p) \neq h_2(p)$ for some $h_1, h_2 \in \mathcal{H}$. Fix h^* to be an optimal monotone classifier on P , namely, $\text{err}_P(h^*) = k^*$. Given a real value $\rho \in (0, 1]$, define the *ball* — denoted as $B(h^*, \rho)$ — as the set of classifiers $h \in \mathbb{H}_{\text{mon}}$ such that $|\{p \in P \mid h(p) \neq h^*(p)\}| \leq \rho n$, namely, h disagrees with h^* on at most ρn elements of P . The *disagreement coefficient* θ [29] of $\mathbb{H}_{\text{mon}}$ under the uniform distribution over P is defined as

$$\theta = \max \left\{ 1, \sup_{\rho \in (\frac{k^*}{n}, 1]} \frac{|\text{DIS}(B(h^*, \rho))|}{\rho \cdot n} \right\}. \quad (76)$$

Next, we construct a multiset P whose θ value is at least its width w . Choose arbitrary integers n and w such that $w \geq 2$, $n > w^2$, and n is a multiple of w . Identify w distinct locations $x_1, x_2, \dots, x_w$ in $\mathbb{R}^2$ where no location dominates another. Place n/w points at each x_i ($i \in [w]$), assign label 1 to all of them except for exactly one point, which is assigned label -1 . This yields n labeled points, which constitute P . It is clear that P has width w and optimal monotone error $k^* = w$. Define h^* to be the optimal monotone classifier that maps the entire $\mathbb{R}^2$ to 1.

Set $\rho = 1/w$, which is greater than $k^*/n = w/n$ because $n > w^2$. The ball $B(h^*, \rho)$ includes every classifier $h \in \mathbb{H}_{\text{mon}}$ that differs from h^* in at most n/w elements of P . For each $i \in [w]$, define h_i as the monotone classifier that maps the entire P to 1 except the n/w elements at x_i , which h_i maps to -1 . Thus, $h_i \in B(h^*, \rho)$ because h_i disagrees with h^* on $n/w = \rho n$ points. For every point $p \in P$, there exist distinct $j_1, j_2 \in [w]$ such that $h_{j_1}(p) \neq h_{j_2}(p)$. Specifically, suppose that p is at location x_{j_1} ; then $h_{j_1}(p) = -1$ but $h_{j_2}(p) = 1$ for any $j_2 \in [w] \setminus \{j_1\}$. It follows that $\text{DIS}(B(h^*, \rho)) = P$; and hence, $\theta \geq \frac{|\text{DIS}(B(h^*, \rho))|}{\rho \cdot n} = w$.

Star Number. The *star number* $\mathfrak{s}$ [31] of $\mathbb{H}_{\text{mon}}$ on P is the largest integer s such that there exist distinct points $p_1, p_2, \dots, p_s \in P$ and distinct classifiers $h_0, h_1, \dots, h_s \in \mathbb{H}_{\text{mon}}$ satisfying the requirement:

$$\forall i \in [s], h_0 \text{ and } h_i \text{ assign the same label to all } p_1, p_2, \dots, p_s \text{ except } p_i.$$

We show that $\mathfrak{s}$ is at least the width w of P . Identify w points $p_1, p_2, \dots, p_w$ of P such that no point dominates another (such points must exist due to Dilworth's Theorem). Define

- h_0 as the monotone classifier that maps $\{p_1, \dots, p_w\}$ to 1;
- for each $i \in [w]$, h_i as the monotone classifier that maps $\{p_1, \dots, p_w\} \setminus \{p_i\}$ to 1 but maps p_i to -1 .

The points $p_1, \dots, p_w$ and classifiers $h_0, h_1, \dots, h_w$ serve as evidence that $\mathfrak{s} \geq w$.

Teaching Dimension. Fix an integer m , and consider an arbitrary $\Sigma \in P^m$ (i.e., Σ is a sequence of m elements from P , some of which may be identical). W.l.o.g., let $\Sigma = (p_1, p_2, \dots, p_m)$. Define

$$\mathcal{L}(\Sigma) = \{(h(p_1), \dots, h(p_m)) \mid h \in \mathbb{H}_{\text{mon}}\}. \quad (77)$$

In other words, $\mathcal{L}(\Sigma)$ includes all possible ways to label Σ using monotone classifiers. Let f be any function from P to $\{-1, 1\}$. Define $\chi_0(f, \Sigma)$ as the smallest integer $s \in [m]$ such that we can find s elements in Σ to

- either reject $(f(p_1), \dots, f(p_m)) \in \mathcal{L}(\Sigma)$

- or identify a unique candidate $(l_1, \dots, l_m) \in \mathcal{L}(\Sigma)$ that may be $(f(p_1), \dots, f(p_m))$.

Formally, the integer s should satisfy the requirement that there is a subset $S \subseteq [m]$ with $|S| = s$ such that at most one $(l_1, \dots, l_m) \in \mathcal{L}(\Sigma)$ satisfies $l_i = f(p_i)$ on every $i \in S$.

Let $\mathcal{D}$ be the uniform distribution over P . Define $\chi_1(f, m, \delta)$ be the smallest integer $s \in [m]$ such that

$$\Pr_{\Sigma \sim \mathcal{D}^m}[\chi_0(f, \Sigma) > s] \leq \delta.$$

Then, the *extended teaching dimension growth function* [27] of $\mathbb{H}_{mon}$ on P is

$$\chi_2(m, \delta) = \max_f \chi_1(f, m, \delta)$$

where the maximization is over all possible functions $f : P \rightarrow \{-1, 1\}$. According to [27], the *teaching dimension* χ in the label complexity in (8) equals $\chi_2(n, \delta)$, where $n = |P|$ and $\delta = 1/\text{polylog } n$.⁵

Next, we construct a multiset P of size n and width w for which $\chi_2(m, 1/2) \geq w/2$. As χ_2 is non-descending in δ , it will then follow that $\chi = \Omega(w)$.

Choose arbitrary integers n and w such that $w \geq 2$, $n \geq 2w$, and n is a multiple of w . Identify w distinct locations $x_1, x_2, \dots, x_w$ in $\mathbb{R}^2$ where no location dominates another. Place n/w points at each x_i ($i \in [w]$). These points constitute our P , which has size n and width w . We fix f to be the function that maps the entire P to 1 and argue that $\chi_1(f, n, 1/2) \geq w/2$, which implies $\chi_2(n, 1/2) = w/2$.

Draw $\Sigma \in \mathcal{D}^n$ (recall that $\mathcal{D}$ is the uniform distribution over P). If Σ includes points from y different locations, we must have $\chi_0(f, \Sigma) \geq y$. W.l.o.g., assume that Σ includes points from locations $x_1, x_2, \dots, x_y$. Monotone classifiers can map each location independently to label -1 or 1 . We need at least y points to identify the unique member of $\mathcal{L}(\Sigma)$ that agrees with f on the entire Σ . Thus, to prove $\chi(f, n, 1/2) \geq w/2$, it suffices to show

$$\Pr_{\Sigma \sim \mathcal{D}^n}[y > w/2] > 1/2$$

or equivalently

$$\Pr_{\Sigma \sim \mathcal{D}^n}[w - y < w/2] > 1/2. \tag{78}$$

Note that $w - y$ is the number of locations having no point picked in Σ — call them the *deserted* locations. The probability that a specific location is deserted is $(1 - 1/w)^n \leq e^{-n/w} \leq 1/e^2 < 0.14$. Hence, $\mathbf{E}_{\Sigma \sim \mathcal{D}^n}[w - y] \leq 0.14w$. By Markov's inequality, $\Pr_{\Sigma \sim \mathcal{D}^n}[w - y \geq 0.3w] \leq 0.14/0.3 < 1/2$. This implies the correctness of (78).

⁵This is obtained from Theorem 3 of [27], which in our context sets m to $\min\{n, \Omega(\frac{1}{\nu+\xi})\}$, where ν and ξ are given in (6) and (7), respectively. When $k^* = 0$, the value of m equals n .