Monotone Classification with Relative Approximations*

YUFEI TAO, The Chinese University of Hong Kong, China

In monotone classification, the input is a multi-set P of points in $\mathbb{R}^d$, each associated with a hidden label from $\{-1, 1\}$. The goal is to identify a monotone function h, which acts as a classifier, mapping from $\mathbb{R}^d$ to $\{-1, 1\}$ with a small *error*, measured as the number of points $p \in P$ whose labels differ from the function values $h(p)$. The *cost* of an algorithm is defined by the number of points having their labels revealed. This article studies the lowest cost required to find a monotone classifier whose error is at most $(1 + \epsilon) \cdot k^*$ where k^* is the minimum error achieved by an optimal monotone classifier and $\epsilon \geq 0$ is a given real value. Nearly matching upper and lower bounds are presented for the full range of ϵ.

CCS Concepts: • **Theory of computation → Approximation algorithms analysis**; **Active learning**.

Additional Key Words and Phrases: Active Learning; Monotone Classification; Entity Matching

ACM Reference Format:
Yufei Tao. . Monotone Classification with Relative Approximations. *J. ACM* , , Article 1 (January), 40 pages.

1 Introduction

This article undertakes a systematic study of monotone classification, aiming to determine the minimum overhead of label discovery to guarantee a classification error that is higher than the optimal error by at most a multiplicative factor. Next, we will begin by defining the problem, followed by an explanation of its practical motivations. After that, we will present our findings and discuss their significance in relation to previous results. Finally, we will provide an overview of our main techniques.

1.1 Problem Definitions

Math Conventions. For any integer $x \geq 1$, the notation $[x]$ represents the set $\{1, 2, ..., x\}$. Given two non-negative real values x and y, we use notation $x \leq_\epsilon y$ 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 multi-set P of n points in $\mathbb{R}^d$ for some integer $d \geq 1$; note that P may contain distinct elements p and q satisfying $p = q$ (i.e., they are at the same location). Each element $p \in P$ carries a label from $\{-1, 1\}$, which is represented as $label(p)$.

*Preliminary versions of this article appeared in PODS'18 [42] and PODS'21 [43]. This work was supported in part by GRF projects 14203421, and 14222822 from HKRGC.

Author's Contact Information: Yufei Tao, The Chinese University of Hong Kong, Hong Kong, China.

ACM 1557-735X//1-ART1
https://doi.org/

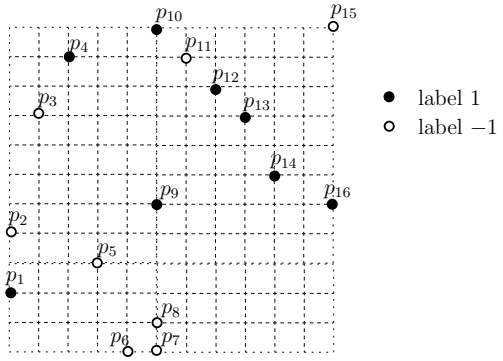


Fig. 1. An input set P for Problem 1

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

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

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

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

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

We will call k^* the *optimal error* of P as this is the lowest error that a monotone classifier can achieve. A classifier $h \in \mathbb{H}_{mon}$ is

- *optimal* on P if $err_P(h) = k^*$;
- *c-approximate* on P if $err_P(h) \leq c \cdot k^*$ 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, $err_P(h) = 3$. No other monotone classifier has a smaller error on P and, hence, $k^* = 3$. Consider the "all-positive" classifier h^{pos} that maps the entire P to 1. Observe that $err_P(h^{pos}) = 7$, because of which h^{pos} is $(7/3)$-approximate on P.*

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

In one extreme, by probing the whole P, an algorithm pays cost n, after which it can find an optimal monotone classifier on P with polynomial CPU computation[1]. In the other extreme, by probing nothing,

[1]Once all the point labels are available, an optimal monotone classifier can be found in time polynomial in n and d; see [2, 41].

an algorithm pays cost 0 but will have 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 for guaranteeing the optimal error k^* or an error higher than k^* by a small multiplicative factor. This gives rise to the following problem.

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.*

When $k^* = 0$, 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 described 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 set to 0 or 1 with probability 1/2. Each 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, $\mathcal{A}_{ran}$ degenerates 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 by $\mathcal{A}_{ran}$. Both X and h are random variables and thus so is $err_P(h)$. 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 error k *with high probability* (w.h.p.) if $\mathbf{Pr}[err_P(h) \leq k] \geq 1 - 1/n^c$ where c can be an arbitrarily large constant specified before running the algorithm.

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

1.2 Practical Motivations

An important application of monotone classification is "entity matching" (also known as "record linkage" or "duplicate detection" in specific contexts). Given two sets of entities E_1 and E_2, the goal of entity matching 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.`prod-name` = "MS Word" vs. e_2.`prod-name` = "Microsoft Word Processor"). Nevertheless, it would be reasonable to expect e_1.`year` = e_2.`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, such as reading the advertisements in detail. 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.

Towards the above purpose, a dominant methodology behind the existing approaches (e.g., [4, 6, 14, 15, 21, 26, 33, 36, 39, 44]) 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 the pairs that apparently cannot be 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 $e_1.\texttt{year} = e_2.\texttt{year}$.

(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 $sim_1, sim_2, ..., sim_d$, 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 $sim_i(e_1, e_2)$: 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 $-|e_1.\texttt{price} - e_2.\texttt{price}|$, 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 $e_1.\texttt{prod-name}$ and $e_2.\texttt{prod-name}$ 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.

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 distribution). Demanding monotonicty 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.

1.3 Related Work: Active Classification and Monotonicity Testing

Classification is a fundamental topic in machine learning. In the standard settings, 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 $h : \mathcal{P} \to \{-1, 1\}$, whose *error rate* with respect to $\mathcal{D}$ is calculated as

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

Let $\mathbb{H}$ be the set of classifiers of interest. The optimal error rate among the classifiers in $\mathbb{H}$ is given by:

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

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 $err\text{-}rate_{\mathcal{D}}(h) \leq \nu + \xi$ 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 until it is ready to produce a classifier meeting the PAC guarantee. In *passive* classification, in each sample (p, l) from $\mathcal{D}$, both p and l are revealed directly. The performance of $\mathcal{A}$ is measured by the *sample cost*, defined as the number of samples drawn.

In many practical applications, producing the point field of a sample from $\mathcal{D}$ requires negligible cost, but determining its label is expensive. This motivates *active* classification, which studies how 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 $\mathcal{A}$ can request l from an *oracle*, thereby doing a *probe*. If $\mathcal{A}$ does not consider the knowledge of l necessary, it may skip the probe. The performance of $\mathcal{A}$ is now measured by the *label cost*, defined as the number of probes performed. The sample cost is no longer a major concern.

Active classification has been extensively studied; see [29, 40] for two excellent surveys. Our subsequent discussion will concentrate on *agnostic* learning where $\nu > 0$ (i.e., even the best classifier in $\mathbb{H}$ has a positive, unknown, error rate); this is the branch most relevant to our work. The modern theory of agnostic active classification is built on two intrinsic parameters

- λ: the VC-dimension of $\mathbb{H}$ on $\mathcal{P}$;
- θ: the disagreement coefficient of $\mathbb{H}$ under $\mathcal{D}$.

We will defer the concrete definitions of λ and θ to Appendix B; for now it suffices to remember that they are both real values at least 1.

The dominant solution to agnostic active classification is an algorithm named $\mathtt{A}^2$. Its initial ideas were developed by Balcan et al. [5] and were subsequently improved in [18, 29]. As shown in [29], the algorithm achieves the PAC guarantee with a label cost of

$$\tilde{O}\left(\theta \cdot \lambda \cdot \frac{\nu^2}{\xi^2}\right) \tag{5}$$

where $\tilde{O}(.)$ hides a factor polylogarithmic to $\theta, 1/\xi$, and $1/\delta$. On the lower bound side, extending an earlier result of Kaariainen [32], Beygelzimer et al. [7] proved that the label cost needs to be

$$\Omega\left(\frac{\nu^2}{\xi^2} \cdot \left(\lambda + \log\frac{1}{\delta}\right)\right). \tag{6}$$

Note that there is a multiplicative gap of θ between the upper and lower bounds. When this parameter is at most a constant, the two bounds match up to polylogarithmic factors. Indeed, most success stories in active classification concern situations where the value of θ is very small; see, e.g., [17, 19, 25, 29, 45].

Unfortunately, this is not true when $\mathbb{H} = \mathbb{H}_{mon}$, the class of monotone classifiers. We will prove that the disagreement coefficient θ of $\mathbb{H}_{mon}$ can be huge even if $\mathcal{P}$ has a finite size. This has two consequences. First, the θ-gap between (5) and (6) becomes significant. This suggests that agnostic active classification has not been well understood on monotone classifiers. Second, when θ is large, the $\mathtt{A}^2$ algorithm incurs an expensive label cost, giving the hope that it could be considerably improved.

A special scenario where monotone classification (i.e., $\mathbb{H} = \mathbb{H}_{mon}$) has received exceptional attention is the so-called "hypercube learning" where $\mathcal{P} = \{0, 1\}^d$, every point $p \in \mathcal{P}$ carries a label $label(p)$, and $\mathcal{D}$ is the uniform distribution over $\{(p, label(p)) \mid p \in P\}$. Improving over previous results [9, 34], Lange and Vasilyan [35] showed that in passive classification the PAC guarantee can be achieved with a sample cost of $2^{\tilde{O}(\sqrt{d}/\xi)}$, where $\tilde{O}(.)$ hides a factor polylogarithmic in d and $1/\xi$.

In Problem 1, we aim at a relative precision guarantee. If, on the other hand, an *additive* precision guarantee is desired, the problem can be cast as an instance of active classification. Specifically, let us set $\mathbb{H}$ to $\mathbb{H}_{mon}$, set $\mathcal{P}$ to the input P of Problem 1, and define $\mathcal{D}$ as the uniform distribution over $\{(p, label(p)) \mid p \in P\}$. In this case, every monotone classifier h satisfies $err\text{-}rate_{\mathcal{D}}(h) = err_P(h)/n$ where $n = |P|$ and $err_P(h)$ is given in (1). It follows that $\nu = k^*/n$, where k^* is defined in (3). Thus, given a specific value of ξ, the $\mathtt{A}^2$ algorithm can find w.h.p. a monotone classifier h on P satisfying $err_P(h) \leq k^* + \xi n$.

Given a multi-set P of n labeled points as defined in Problem 1, *monotonicity testing* is the problem of deciding whether P is monotone (i.e., having optimal 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 $\mathcal{A}$ can interact 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. [24] gave an algorithm to perform monotonicty testing in $O(\sqrt{n/\xi})$ probes. The problem has also been explored in other settings that are less relevant to this article; the interested readers may refer to [10, 11, 13, 28] and the references therein.

1.4 Our Results

Under $\epsilon = 0$, the goal of Problem 1 is to find an optimal monotone classifier. We prove that any algorithm achieving the purpose with a nontrivial probability needs to probe $\Omega(n)$ points in expectation (Theorem 11 in Section 4). Consequently, the naive strategy of probing all points can no longer be improved by more than a constant factor. Our lower bound holds even when the dimensionality d is 1 and the optimal error k^* defined in (3) is given to the algorithm "for free".

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

$$w = \text{the size of the largest } S \subseteq P \text{ such that } \nexists \text{ distinct } p, q \in S \text{ satisfying } p \succeq q. \tag{7}$$

(a) A point set of width 1 (b) A point set of width n

Fig. 2. Illustration of the dominance width

Any one-dimensional P has $w = 1$. Once the dimensionality d reaches 2, the width w can be anywhere between 1 and n; see Figure 2 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* (7)*. 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 now enhance our understanding of the A^2 algorithm (reviewed in Section 1.3 as the state of the art in active classification) in the context of Problem 1. As explained in Section 1.3, the algorithm finds w.h.p. a classifier whose error on P is at most $k^* + \xi n$. Let us "favor" the algorithm by telling it the value of k^* for free and considering only the non-realizable case $k^* > 0$. Consequently, the algorithm can set ξ to $\epsilon k^*/n$ and solve Problem 1 w.h.p. at a cost given in (5). As shown in Appendix B, both λ (the VC-dimension) and θ (the disagreement coefficient) can be bounded by $O(w)$; furthermore, they can be $\Omega(w)$ simultaneously. Applying $\nu = k^*/n$, the best bound that can be derived from (5) is

$$\tilde{O}(w^2/\epsilon^2) \tag{8}$$

where $\tilde{O}(.)$ hides a factor polylogarithmic in $n = |P|$ and $1/\epsilon$.

This article will prove that the "correct" complexity of Problem 1 turns out to be lower than (8) by roughly a factor of w. We achieve the purpose by establishing nearly-matching upper and lower bounds. Regarding upper bounds:

- We design in the next subsection an algorithm that, as analyzed in Section 2, guarantees an expected error at most $2k^*$ with an expected cost of $O(w \operatorname{Log} \frac{n}{w})$. When $k^* = 0$ — i.e., Problem 1 is realizable — the algorithm always finds an optimal monotone classifier.
- We present in Section 3 an algorithm that guarantees an error of $(1 + \epsilon)k^*$ w.h.p. at a cost of $O(\frac{w}{\epsilon^2} \cdot \operatorname{Log} \frac{n}{w} \cdot \log n)$. When $k^* = 0$, the algorithm finds an optimal monotone classifier w.h.p..

We now turn to our lower bounds:

- For any constant $c > 1$, we prove in Section 5 that any algorithm ensuring an expected error at most ck^* must probe $\Omega(w \operatorname{Log} \frac{n}{(k^*+1)w})$ points in expectation (Theorem 14). Hence, the cost of our first algorithm (with expected error at most $2k^*$) is asymptotically optimal when $k^* \leq (n/w)^{1-\delta}$ where $\delta > 0$ is an arbitrarily small constant.
- For any $\epsilon > 0$, we prove in Section 6 that any algorithm ensuring an expected error at most $(1 + \epsilon)k^*$ must probe $\Omega(w/\epsilon^2)$ points in expectation (Theorem 15). Hence, the cost of our second algorithm is asymptotically optimal up to a factor of $O(\operatorname{Log} \frac{n}{w} \cdot \log n)$.

Table 1 summarizes all the results mentioned earlier.

ϵ	probing cost	ref	remarks
any	$\tilde{O}(w^2/\epsilon^2)$	[5, 18, 29]	k^* known, w.h.p. error
1	$O(w \operatorname{Log} \frac{n}{w})$ expected	Thm. 1	expected error
any	$O(\frac{w}{\epsilon^2} \cdot \operatorname{Log} \frac{n}{w} \cdot \log n)$	Thm. 6 + Sec. 1.5	w.h.p. error
0	$\Omega(n)$	Thm. 11	success probability $> 2/3$
constant > 0	$\Omega(w \operatorname{Log} \frac{n}{(k^*+1)w})$ expected	Thm. 14	expected error
any	$\Omega(w/\epsilon^2)$ expected	Thm. 15	expected error

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

As a side product, our findings also imply a new result for monotonicity testing. Recall from Section 1.3 that the input to monotonicity testing involves a multi-set 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 $O(w \operatorname{Log} \frac{n}{w}) + 2/\xi$, where w is the width of P. This provides a meaningful alternative to the solution of [27], which probes $O(\sqrt{n/\xi})$ points as mentioned earlier.

1.5 Our Techniques

This subsection will highlight the new techniques developed in this work. Our discussion will focus on the proposed algorithms for solving Problem 1.

A Simple Algorithm. Our first algorithm — named RPE (random probes with elimination) — is remarkably elegant:

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** $label(z) = 1$ **then** remove from P every $p \in P$ satisfying $p \succeq z$ (note: this removes z)
5. **else** discard from P every $p \in P$ satisfying $z \succeq p$ (note: this removes z)
6. **return** Z

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

$$h_{\text{RPE}}(p) = \begin{cases} 1 & \text{if } \exists z \in Z \text{ such that } label(z) = 1 \text{ and } p \succeq z \\ -1 & \text{otherwise} \end{cases} \quad (9)$$

The classifier h_{RPE} 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 $label(z) = 1$ and $q \succeq z$ (such z must exist by the definition in (9)). It follows that $p \succ z$, in which case $h_{\text{RPE}}(p)$ should be 1 according to (9), giving a contradiction.

Another way to understand RPE is to regard it as the following labeling process. After acquiring the label of a random element $z \in P$ from the oracle, we will make sure that h_{RPE} maps z to $label(z)$. Accordingly, this requires us to finalize many other mappings because of monotonicity. Specifically, if $label(z) = 1$, then h_{RPE} assigns label 1 to all the elements $p \succeq z$; otherwise, h_{RPE} assigns label -1 to all

the elements p satisfying $z \succeq p$. The elements that have their labels assigned are removed from P. The process is then repeated recursively on the remaining elements.

Example 1.3. *Let us illustrate the algorithm on the input P in Figure 1. Assume that* RPE *randomly probes (at Step 3) p_1 first. Acquiring 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 label$(p_8) = -1$, it removes all the remaining points in P. With $Z = \{p_1, p_8\}$, the classifier h_{RPE} in (9) maps all the points to label 1 except p_6, p_7, and p_8. Its error on P is $err_P(h_{\text{RPE}}) = 5$.*

While RPE is procedurally simple, proving its theoretical guarantees requires nontrivial arguments. It is unclear why the classifier in (9) ensures an expected error at most $2k^*$ — notably, probing a "wrong" element of P may immediately force h_{RPE} to misclassify a large number of elements (imagine, e.g., the consequence if p_{15} is probed first). Equally intriguing is why RPE ends up probing $O(w \operatorname{Log} \frac{n}{w})$ elements in expectation, particularly since the algorithm never computes the value of w. In Section 2, we will unveil the answers to these questions through a novel analysis. The RPE algorithm also serves as the main step of our proposed method for monotonicity testing.

Relative-Comparison Coresets. Let us now turn our attention to guaranteeing an approximation ratio of $1 + \epsilon$ where $0 < \epsilon \leq 1$. To that aim, we will produce a function $F : \mathbb{H}_{mon} \to \mathbb{R}$ that ensures:

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

For every $h \in \mathbb{H}_{mon}$, the function value $F(h)$ can be precisely computed. Given F, we can then focus on identifying a monotone classifier $h^{\circledast}$ with the lowest $F(h^{\circledast})$. This classifier must fulfill the condition $err_P(h^{\circledast}) \leq (1+\epsilon)k^*$ (noticing that $F(h^{\circledast}) \leq F(h^*)$ 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.

An inherent barrier in finding such a function F is that we cannot hope to estimate $err_P(h)$ of every monotone classifier h up to a finite relative ratio without $\Omega(n)$ probes. This is true even if the dimensionality d of P is 1. To see why, consider the classifier h^{pos} that maps the entire P to 1. It has error 0 if all the 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.

Our method to produce a desired F is to ensure the following inequality for every $h \in \mathbb{H}_{mon}$:

$$err_P(h) \cdot \left(1 - \frac{\epsilon}{4}\right) + \Delta \leq F(h) \leq err_P(h) \cdot \left(1 + \frac{\epsilon}{4}\right) + \Delta \tag{10}$$

where Δ is an **unknown** real value common to all h. Indeed, 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 success 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 (10) because

$$err_P(h) \quad \leq \quad \frac{1}{1 - \epsilon/4} \cdot (F(h) - \Delta)$$

$$\begin{aligned}(\text{by } F(h) \le F(h')) \quad &\le \quad \frac{1}{1-\epsilon/4} \cdot (F(h') - \Delta)\\ (\text{applying (10)}) \quad &\le \quad \frac{1+\epsilon/4}{1-\epsilon/4} \cdot err_P(h')\\ &\le \quad (1+\epsilon) \cdot err_P(h')\end{aligned}$$

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

We will show that a function F meeting the unusual condition in (10) can be obtained using a coreset. Specifically, a *coreset* of P 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 $h \in \mathbb{H}_{mon}$, define its *weighted error* on Z as:

$$w\text{-}err_Z(h) \quad = \quad \sum_{p \in Z} weight(p) \cdot \mathbb{1}_{h(p) \neq label(p)}. \tag{11}$$

In Section 3, we show how to perform $O(\frac{w}{\epsilon^2} \operatorname{Log} \frac{n}{w} \cdot \log n)$ probes to obtain w.h.p. a coreset Z of size $O(\frac{w}{\epsilon^2} \operatorname{Log} \frac{n}{w} \cdot \log n)$, such that every $h \in \mathbb{H}_{mon}$ satisfies

$$err_P(h) \cdot \left(1 - \frac{\epsilon}{4}\right) + \Delta \le w\text{-}err_Z(h) \le err_P(h) \cdot \left(1 + \frac{\epsilon}{4}\right) + \Delta \tag{12}$$

for some unknown Δ common to all h. The function $w\text{-}err_Z$ readily serves as the desired function F and can be evaluated for any $h \in \mathbb{H}_{mon}$ based purely on Z. We will refer to Z as a *relative-comparison coreset* because its purpose is to enable the relative ϵ-comparison property. The "unknown-Δ" technique outlined above is different from all the existing coreset-building methods (see representative works [1, 8, 12, 16, 22, 23, 30, 31, 38] and their references) to our knowledge.

2 Random Probes with Elimination

This section will prove the theorem below regarding the RPE algorithm in Section 1.5:

Theorem 1. *For Problem 1, the* RPE *algorithm probes $O(w \operatorname{Log} \frac{n}{w})$ elements in expectation and the monotone classifier h_{RPE} in (9) has an expected error at most $2k^*$, where n is the number of elements in the input P, w is the width of P, and k^* is the optimal error of P.*

The expected error guarantee implies that when $k^* = 0$ (Problem 1 is realizable), the classifier h_{RPE} is always optimal (otherwise, the expected error would be strictly positive). 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 an important property of the classifier h_{RPE}, which establishes symmetry between labels -1 and 1.

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

Proof. We first prove that Z is a monotone set, or specifically:

- Z contains no two (distinct) elements p and q with $p = q$;

- for any $p, q \in Z$, if $p \succ q$, then $label(p) \geq label(q)$.

The first bullet holds because once an element $z \in P$ is probed at Line 3 of the pseudocode in Section 1.5, then 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 $p \succ q$, $label(p) = -1$, and $label(q) = 1$. But 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 $p \in Z$.

We are ready to prove that, for any $p \in P$, the condition $h_{\text{RPE}}(p) = -1$ holds if and only if $\exists z \in Z$ satisfying $label(z) = -1$ 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 $label(p) = -1$ (indeed, if $label(p) = 1$, then $h_{\text{RPE}}(p) = 1$ because we can $z = P$ in (9)). As $p \in Z$ and $label(p) = -1$, 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 (9) that no element $z' \in Z$ satisfies $label(z') = 1$ and $p \succeq z'$. Thus, RPE must have removed p after probing an element z satisfying $label(z) = -1$ and $z \succeq p$. This z completes the only-if direction.

Finally, we consider the "if direction" (i.e., $\Leftarrow$). The designated element z rules out the existence of any element $z' \in Z$ satisfying $label(z') = 1$ 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 (9). □

2.1 The Expected Error of RPE

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

Lemma 2. *The number of h-good elements misclassified by the classifier h_{RPE} from (9) is at most $err_P(h)$ in expectation.*

The lemma implies $\mathbf{E}[err_P(h_{\text{RPE}})] \leq 2k^*$. 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 $err_P(h^*) = k^*$ 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.

The rest of this subsection serves as a proof of Lemma 2. Our proof 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 will establish its correctness for $n = m$. Define

$$X \quad = \quad \text{the number of } h\text{-good elements in } P \text{ misclassified by } h_{\text{RPE}}, \tag{13}$$

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

For each h-bad element p, we define its *influence set* $I_{bad}(p)$ 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 $label(q) = -1$.

For each h-good element $p \in P$, define its *influence set* $I_{good}(p)$ as follows:

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

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

Lemma 3. *Both of the statements below are true:*

(1) *If $p \in P$ is h-good, then $label(p) \neq label(q)$ for every $q \in I_{good}(p)$.*
(2) *For any h-good $p \in P$ and any h-bad $q \in P$, we have $p \in I_{good}(q) \Leftrightarrow q \in I_{bad}(p)$.*

Proof. Let us start with statement (1). Suppose that $label(p) = -1$. Because p is h-good, we know $h(p) = -1$. As $q \in I_{good}(p)$, we must have $p \succeq q$. By monotonicity, $h(q)$ must be -1 as well. Since q is h-bad, it follows that $label(q) = 1$. A symmetric argument proves the statement in the case where $label(p) = 1$. Statement (2) follows directly from statement (1) and the influence set definitions. □

Statement (2) of Lemma 3 leads to

$$\sum_{h\text{-good } p \in P} |I_{good}(p)| = \sum_{h\text{-bad } p \in P} |I_{bad}(p)|. \tag{14}$$

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.*

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

Recall from its pseudocode in Section 1.5 that RPE is an iterative algorithm. We will refer to Lines 3-5 as an *iteration*. Let z be the point probed (at Line 3) in the first iteration. The revelation of $label(z)$ instructs the algorithm to remove z and possibly some other elements from P (at Line 4 or 5). Define

$$P_z = \text{the set of remaining elements at the end of the first iteration.} \tag{15}$$

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 $err_{P_z}(h) \leq err_P(h) - 1$ follows trivially from the fact that P_z has lost at least one h-bad element (i.e., z) compared to P.

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

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

$$X \quad = \quad Y_z + \text{the number of } h\text{-good elements in } P \setminus P_z \text{ misclassified by } h_{\text{RPE}} \tag{16}$$

where X is defined in (13).

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

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

Proof. Let us first discuss the scenario where $label(z) = 1$. The set $P \setminus P_z$ consists of every point $p \in P$ satisfying $p \succeq z$. The classifier h_{RPE} maps every point $p \in P \setminus P_z$ to 1. Hence, the number of h-good points in $P \setminus P_z$ misclassified by h_{RPE} is exactly the number — let it be x — of h-good points in $P \setminus P_z$ whose labels are -1.

Next, we analyze the number x. If z is h-bad, then x is exactly $|I_{bad}(z)|$ by definition of $I_{bad}(z)$. Consider the case where z is h-good. By monotonicity of h, we have $h(p) = 1$ for all $p \succeq z$ (recall that $label(z) = 1$). Hence, if p is h-good, then $label(p) = h(p) = 1$. The number of x is thus 0 in this case.

A symmetric argument applies to the scenario where $label(z) = -1$, using Proposition 1. □

Combining (16) and Proposition 3 yields

$$X \quad = \quad \begin{cases} Y_z + |I_{bad}(z)| & \text{if } z \text{ is } h\text{-bad} \\ Y_z & \text{if } z \text{ is } h\text{-good} \end{cases} \tag{17}$$

under the event that z is the first point probed.

The subsequent execution of RPE on P_z can be regarded as invoking RPE directly on P_z. We utilize this recursive view to proceed in the analysis. As P_z has at least one less element than P (because z is removed), the inductive assumption tells us:

$$\mathbf{E}[Y_z] \quad \leq \quad err_{P_z}(h). \tag{18}$$

We can now derive

$$\begin{aligned}
\mathbf{E}[X] &= \sum_{z \in P} \mathbf{E}[X \mid z \text{ is probed first}] \cdot \mathbf{Pr}[z \text{ is probed first}] \\
&= \frac{1}{m} \sum_{z \in P} \mathbf{E}[X \mid z \text{ is probed first}] \\
&= \frac{1}{m} \Big(\sum_{h\text{-good } z \in P} \mathbf{E}[X \mid z \text{ is probed first}] + \sum_{h\text{-bad } z \in P} \mathbf{E}[X \mid z \text{ is probed first}] \Big) \\
\text{(by (17))} \quad &= \Big(\frac{1}{m} \sum_{h\text{-good } z \in P} \mathbf{E}[Y_z] \Big) + \Big(\frac{1}{m} \sum_{h\text{-bad } z \in P} \mathbf{E}[Y_z] + |I_{bad}(z)| \Big) \\
\text{(by (18))} \quad &\leq \Big(\frac{1}{m} \sum_{h\text{-good } z \in P} err_{P_z}(h) \Big) + \Big(\frac{1}{m} \sum_{h\text{-bad } z \in P} err_{P_z}(h) + |I_{bad}(z)| \Big) \\
\text{(Proposition 2)} \quad &\leq \Big(\frac{1}{m} \sum_{h\text{-good } z \in P} err_P(h) - |I_{good}(z)| \Big) + \Big(\frac{1}{m} \sum_{h\text{-bad } z \in P} err_P(h) - 1 + |I_{bad}(z)| \Big)
\end{aligned}$$

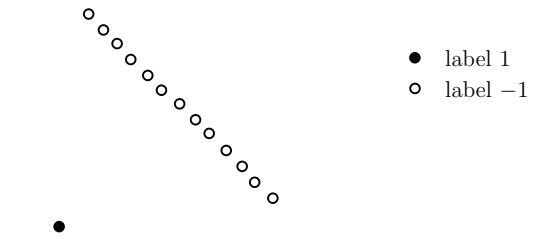


Fig. 3. Approximation ratio 2 is tight for RPE

$$\begin{aligned} &\leq \Big(\frac{1}{m}\sum_{h\text{-good } z \in P} err_P(h) - |I_{good}(z)|\Big) + \Big(\frac{1}{m}\sum_{h\text{-bad } z \in P} err_P(h) + |I_{bad}(z)|\Big) \\ &= err_P(h) + \frac{1}{m}\Big(\sum_{h\text{-bad } z \in P} |I_{bad}(z)| - \sum_{h\text{-good } z \in P} |I_{good}(z)|\Big) \\ \text{(by (14))} \quad &= err_P(h). \end{aligned}$$

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

Remark. As explained, Lemma 2 implies that RPE guarantees an expected error at most $2k^*$. The approximation ratio 2 is the best possible for this algorithm. To see this, consider the input P in Figure 3, where $n-1$ white points have label -1 and the only black point has label 1. The optimal 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 $n-1$. On the other hand, if the first point probed is white, then h_{RPE} misclassies only the black point and incurs an error 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 review a fundamental property of the width w of P defined in (7). Call a subset $S \subseteq P$ a *chain* if it is possible to linearize the points of S into a sequence $p_1 \preceq p_2 \preceq ... \preceq p_{|S|}$. A *chain decomposition* of P is a collection of disjoint chains $C_1, C_2, ..., C_t$ (for some $t \geq 1$) whose union is P. Dilworth's Theorem [20] 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. This serves as evidence that the width w of P is 6.*

Attrition and Elimination. Before discussing the cost of RPE, let us take a detour to discuss a relevant problem. Consider the following *attrition-and-elimination (A&E) game* between Alice and Bob. The input to the game 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.

Lemma 4. *Regardless of Bob's strategy, the game has $O(\text{Log}\, m)$ rounds in expectation.*

Proof. Let X be the number of elements left in C after the first round. We will show $\mathbf{Pr}[X \leq m/2] > 1/2$. 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 $p_1 \preceq p_2 \preceq ... \preceq p_{m'}$. If Alice picks $p = p_i$ ($i \in [m']$), then at most $i - 1$ points are left after her elimination because such points must be among $p_1, p_2, ..., p_{i-1}$. Hence, $X \leq m'/2$ as long as $i \leq 1 + m'/2$, which occurs with probability greater than $1/2$. The fact $\mathbf{Pr}[X \leq m/2] > 1/2$ 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 $\mathbf{E}[Y] \leq 2\log_2(1 + m) + m \cdot \mathbf{Pr}[\text{at least } 2\log_2(1 + m) \text{ rounds}] = O(\text{Log}\, m)$. □

Cost Analysis of RPE. Returning to RPE, let $\{C_1, C_2, ..., C_w\}$ 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:

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

Lemma 5. RPE *probes $O(\text{Log}\, |C_i^{pos}|)$ 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 $C = C_i^{pos}$, 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} that have been 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 $z \in C_i^{pos}$, he passes the turn to Alice.

From Alice's perspective, conditioned on $z \in C_i^{pos}$, 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 $O(\text{Log}\, |C_i^{pos}|)$ rounds in expectation regardless of Bob's strategy. Hence, RPE probes $O(\text{Log}\, |C_i^{pos}|)$ elements from C_i^{pos} in expectation. □

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 the fact $\sum_{i=1}^{w} |C_i| = w$. This completes the proof of Theorem 1.

2.3 Application to Monotonicity Testing

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

- always "yes" if P is monotone;
- "no" with probability at least $2/3$ if $k^* \geq \xi n$ where k^* is the optimal 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 (9)
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 will 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 $err_P(h_{\text{RPE}}) \geq k^*$, 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 will solve Problem 1 up to an approximation ratio $1+\epsilon$ 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 $p \in Z$ has its label revealed and is associated with a positive weight such that the weighted error of every monotone classifier h on Z — namely, $w\text{-}err_Z(h)$ defined in (11) — satisfies the condition in (12), where Δ is some unknown value common to all $h \in \mathbb{H}_{mon}$. Formally, we will establish:

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

As explained in Section 1.5, given the coreset Z in Theorem 6, the remaining task to solve Problem 1 is to find a monotone classifier $h^{\circledast}$ minimizing $w\text{-}err_Z(h^{\circledast})$. The task requires no more probing and can be done in CPU time polynomial in $|Z|$ and d (see [2, 41]). This leads us to:

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

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

Problem 2. *Let* P *be a multi-set of 1D labeled points as defined in Problem 1 (under* $d = 1$*), and* ϵ *be a value in* $(0, 1]$*. Find a function* $F : \mathbb{H}_{mon} \to \mathbb{R}$ *such that every* $h \in \mathbb{H}_{mon}$ *satisfies* (19) *and* (20)*:*

$$|F(h) - err_P(h)| \leq \epsilon|P|/64 \tag{19}$$

$$err_P(h)\cdot\left(1-\frac{\epsilon}{4}\right)+\Delta \leq F(h) \leq err_P(h)\cdot\left(1+\frac{\epsilon}{4}\right)+\Delta \tag{20}$$

where Δ *is a possibly unknown value with*

$$|\Delta| \leq \epsilon|P|/64. \tag{21}$$

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

Two remarks are in order:

- The value Δ in (20) is the same for all $h \in \mathbb{H}_{mon}$.
- Before returning a function F, we must make sure that $F(h)$ is computable for every $h \in \mathbb{H}_{mon}$.

Sections 3.1-3.4 will settle Problem 2 with probability at least $1-\delta$ by performing $O(\frac{w}{\epsilon^2}\operatorname{Log}\frac{w}{n}\cdot\log\frac{n}{\delta})$ probes. Section 3.5 will then utilize our solution to build a relative-comparison coreset that meets the requirements of Theorem 6 for $d = 1$. Section 3.6 will extend the argument to $d > 1$.

3.1 Warm Up: A Special Case

Solving Problem 2 requires demonstrating the existence of Δ even when its actual value remains undetermined. This subsection will illustrate this principle in the specific case where all elements of P have identical values (i.e., all the points in P are located at the same position). In this case, any monotone classifier h maps the entire P to 1 or -1. We denote by h^{pos} (resp., h^{neg}) the monotone classifier that always outputs 1 (resp., -1). It suffices to construct a function $F : \{h^{pos}, h^{neg}\} \to \mathbb{R}$ such that (19)-(21) hold for $h \in \{h^{pos}, h^{neg}\}$.

As it turns out, in this special instance, we can simply return an arbitrary function $F : \{h^{pos}, h^{neg}\} \to \mathbb{R}$ satisfying (19). It is standard to build such a function with random sampling, the details of which will be given in Section 3.4. What is less standard, however, is to argue for the availability of a Δ value to meet the requirements in (20) and (21). W.l.o.g., let us assume that the optimal error k^* is achieved by h^{pos}, namely, $k^* = err_P(h^{pos})$. Our argument distinguishes two scenarios depending on whether k^* is large.

When $k^* \geq |P|/16$. This is a simple scenario and can be dealt with by setting $\Delta = 0$. This choice of Δ trivially satisfies (21). To see why (20) holds as well, recall that the value $F(h)$ estimates $err_P(h)$ up to an absolute offset of $\epsilon|P|/64$ for $h \in \{h^{pos}, h^{neg}\}$. As $err_P(h) \geq |P|/16$, the offset is at most $\frac{\epsilon}{4}err_P(h)$. This yields $err_P(h)\cdot(1-\frac{\epsilon}{4}) \leq F(h) \leq err_P(h)\cdot(1+\frac{\epsilon}{4})$.

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

$$\Delta = F(h^{pos}) - k^*. \tag{22}$$

Even though we cannot compute Δ (because k^* is unknown), we are certain that it exists. Furthermore, as F satisfies (19) for $h = h^{pos}$, we have $|\Delta| \leq \epsilon|P|/64$; hence, (21) holds.

Next, we will explain why (20) holds for $h \in \{h^{pos}, h^{neg}\}$. This is easy for $h = h^{pos}$, in which case (20) becomes $k^*(1 - \epsilon/4) \leq k^* \leq k^*(1 + \epsilon/4)$, which is true. Regarding h^{neg}, first note that as $k^* < |P|/16$, we have $err_P(h^{neg}) = |P| - err_P(h^{pos}) \geq \frac{15}{16}|P|$. Thus, from (19):

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

On the other hand, as explained earlier,

$$|\Delta| \leq \frac{\epsilon|P|}{64} \leq \frac{\epsilon}{60} \cdot err_P(h^{neg})$$

Hence

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

We thus conclude that (20) holds for $h = h^{neg}$.

3.2 A Recursive Framework for Problem 2

This subsection discusses Problem 2 in its generic settings. If $|P| = 1$, we probe the only element in P and return $F(h) = err_P(h)$ for any $h \in \mathbb{H}_{mon}$. Our subsequent discussion assumes $|P| \geq 2$.

When $d = 1$, a monotone classifier h has the form

$$h(p) = \begin{cases} 1 & \text{if } p > \tau \\ -1 & \text{otherwise} \end{cases} \tag{23}$$

which is parameterized by a value τ; we will sometimes make the parameter explicit by representing the classifier in (23) as h^τ.

We will construct (in Section 3.4) a function $G_1 : \mathbb{H}_{mon} \to \mathbb{R}$ that approximates err_P up to absolute error $\epsilon|P|/64$, namely,

$$|G_1(h) - err_P(h)| \leq \epsilon|P|/64 \tag{24}$$

for all $h \in \mathbb{H}_{mon}$. The function is said to be *consistently large* if

$$G_1(h^\tau) \geq |P| \left(\frac{1}{4} - \frac{\epsilon}{64}\right) \text{ for all } \tau \in \mathbb{R}. \tag{25}$$

Our subsequent development depends on whether (25) is true.

3.2.1 When G_1 Is Consistently Large. In this case, we decide the target function F to be

$$F(h) = G_1(h). \tag{26}$$

3.2.2 When G_1 Is Not Consistently Large. Define:

$$\alpha = \text{the smallest } \tau \in \mathbb{R} \text{ with } G_1(h^\tau) < |P| \cdot \left(\frac{1}{4} - \frac{\epsilon}{64}\right)$$
$$\beta = \text{the largest } \tau \in \mathbb{R} \text{ with } G_1(h^\tau) < |P| \cdot \left(\frac{1}{4} - \frac{\epsilon}{64}\right).$$

Our construction of G_1 in Section 3.4 makes α and β well defined when G_1 is not consistently large. Now, break P into:

$$P_\alpha = \{p \in P \mid p = \alpha\} \tag{27}$$
$$P_{mid} = \{p \in P \mid \alpha < p \leq \beta\} \tag{28}$$
$$P_{rest} = (P \setminus P_\alpha) \setminus P_{mid} \tag{29}$$

Note that these are multi-sets where each (1D) point inherits its label in P.

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

Proof. We argue that P_{mid} has 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 (24) tells us $G_1(h^\beta) \geq |P|(\frac{1}{4} - \frac{\epsilon}{64})$, contradicting the definition of β.

Similarly, we argue that P_{mid} has 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 (24) tells us $G_1(h^\alpha) \geq |P|(\frac{1}{4} - \frac{\epsilon}{64})$, contradicting the definition of α. □

We will construct (again in Section 3.4) another function $G_2 : \mathbb{H}_{mon} \to \mathbb{R}$ fulfilling two requirements:

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

$$|G_2(h) - err_{P_{rest}}(h)| \leq \epsilon|P_{rest}|/64; \tag{30}$$

- **G2-2:** for any $\tau \in [\alpha, \beta]$, it holds that

$$G_2(h^\tau) = G_2(h^\beta). \tag{31}$$

By solving Problem 2 on P_α using the solution in Section 3.1 and on P_{mid} recursively, we obtain functions $F_\alpha : \mathbb{H}_{mon} \to \mathbb{R}$ and $F_{mid} : \mathbb{H}_{mon} \to \mathbb{R}$ such that every $h \in \mathbb{H}_{mon}$ satisfies (32)-(35):

$$|F_\alpha(h) - err_{P_\alpha}(h)| \leq \epsilon|P_\alpha|/64 \tag{32}$$
$$|F_{mid}(h) - err_{P_{mid}}(h)| \leq \epsilon|P_{mid}|/64 \tag{33}$$
$$err_{P_\alpha}(h) \cdot \left(1 - \frac{\epsilon}{4}\right) + \Delta_\alpha \leq F_\alpha(h) \leq err_{P_\alpha}(h) \cdot \left(1 + \frac{\epsilon}{4}\right) + \Delta_\alpha \tag{34}$$
$$err_{P_{mid}}(h) \cdot \left(1 - \frac{\epsilon}{4}\right) + \Delta_{mid} \leq F_{mid}(h) \leq err_{P_{mid}}(h) \cdot \left(1 + \frac{\epsilon}{4}\right) + \Delta_{mid} \tag{35}$$

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

$$|\Delta_\alpha| \leq \epsilon|P_\alpha|/64 \tag{36}$$
$$|\Delta_{mid}| \leq \epsilon|P_{mid}|/64 \tag{37}$$

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

$$F(h) = G_2(h) + F_\alpha(h) + F_{mid}(h). \tag{38}$$

3.3 Correctness of the Framework

Next, we prove that the above framework always produces a function F obeying (19)-(21).

3.3.1 When G_1 Is Consistently Large. In this scenario, the constructed F is given in (26). We will show that F satisfies (19) and (20) with $\Delta = 0$. Note that this choice of Δ trivially fulfills (21).

Proof of (19)**.** This directly follows from (24) and (26).

Proof of (20)**.** As G_1 is consistently large, we can first apply (24) and then (25) to derive

$$err_P(h) \geq G_1(h) - \frac{\epsilon|P|}{64} \geq \frac{|P|}{4} - \frac{\epsilon|P|}{64} - \frac{\epsilon|P|}{64} \geq \frac{14|P|}{64} \tag{39}$$

where the last step used $\epsilon \leq 1$. Applying (24) and then (39) yields

$$|G_1(h) - err_P(h)| \leq \frac{\epsilon|P|}{64} \leq \frac{\epsilon}{64} \cdot \frac{64 \cdot err_P(h)}{14} < \frac{\epsilon \cdot err_P(h)}{4}$$

which leads to

$$err_P(h) \cdot (1 - \epsilon/4) \leq F(h) \leq err_P(h) \cdot (1 + \epsilon/4)\,.$$

Thus, (20) holds with $\Delta = 0$.

3.3.2 When G_1 Is Not Consistently Large. Inductively, assuming that (33), (35), and (37) hold on P_{mid}, we will show that the function F produced in (38) satisfies (19)-(21) with

$$\Delta \quad = \quad \Delta_\alpha + \Delta_{mid} + G_2(h^\beta) - err_{P_{rest}}(h^\beta). \tag{40}$$

Proof of (19)**.** For any $h \in \mathbb{H}_{mon}$, it holds that

$$err_P(h) \quad = \quad err_{P_\alpha}(h) + err_{P_{mid}}(h) + err_{P_{rest}}(h). \tag{41}$$

Combining the above with (38) gives:

$$\begin{aligned} |F(h) - err_P(h)| \quad &\leq \quad |F_\alpha(h) - err_{P_\alpha}(h)| + |F_{mid}(h) - err_{P_{mid}}(h)| + |G_2(h) - err_{P_{rest}}(h)| \\ \text{(by (30), (32), and (33))} \quad &\leq \quad \frac{\epsilon|P_\alpha|}{64} + \frac{\epsilon|P_{mid}|}{64} + \frac{\epsilon|P_{rest}|}{64} \leq \frac{\epsilon|P|}{64} \end{aligned}$$

where the last step used the fact that P_α, P_{mid}, and P_{rest} decompose P; see (27)-(29).

Proof of (21)**.** From (30), (36), (37), and (40), we know

$$|\Delta| = |\Delta_\alpha + \Delta_{mid} + G_2(h^\beta) - err_{P_{rest}}(h^\beta)| \leq \frac{\epsilon|P_\alpha|}{64} + \frac{\epsilon|P_{mid}|}{64} + \frac{\epsilon|P_{rest}|}{64} \leq \frac{\epsilon|P|}{64}. \tag{42}$$

Proof of (20)**.** We will prove that $F(h^\tau)$ satisfies (20) for all $\tau \in \mathbb{R}$ in two separate lemmas.

Lemma 8. *The requirement* (20) *is fulfilled when $\tau < \alpha$ or $\tau > \beta$.*

Proof. By the definitions of α and β, we have $G_1(h^\tau) \geq |P|(\frac{1}{4} - \frac{\epsilon}{64})$ when $\tau < \alpha$ or $\tau > \beta$. Hence, by the same derivation in (39), we obtain $err_P(h^\tau) \geq \frac{14}{64}|P|$. Combining this with (42) yields

$$\frac{\epsilon \cdot err_P(h^\tau)}{4} + \Delta \geq \frac{\epsilon}{4} \cdot \frac{14|P|}{64} - \frac{\epsilon|P|}{64} = \frac{10\epsilon|P|}{256}.$$

As proved earlier, F satisfies (19), which tells us

$$F(h^\tau) - err_P(h^\tau) \leq \frac{\epsilon|P|}{64} < \frac{10\epsilon|P|}{256} \leq \frac{\epsilon \cdot err_P(h^\tau)}{4} + \Delta. \tag{43}$$

Similarly, from $err_P(h^\tau) \geq \frac{14}{64}|P|$ and (42), we know

$$\frac{\epsilon \cdot err_P(h^\tau)}{4} - \Delta \geq \frac{\epsilon}{4} \cdot \frac{14|P|}{64} - \frac{\epsilon|P|}{64} = \frac{10\epsilon|P|}{256}.$$

Hence, (19) tells us

$$err_P(h^\tau) - F(h^\tau) \leq \frac{\epsilon|P|}{64} < \frac{10\epsilon|P|}{256} \leq \frac{\epsilon \cdot err_P(h^\tau)}{4} - \Delta. \tag{44}$$

The correctness of (20) now follows from (43) and (44). □

Lemma 9. (20) *holds when* $\alpha \leq \tau \leq \beta$.

Proof. For any $\tau \in [\alpha, \beta]$, we have

$$err_{P_{rest}}(h^\tau) \quad = \quad err_{P_{rest}}(h^\beta) \tag{45}$$

because P_{rest} has no element in $[\alpha, \beta]$. By requirement **G2-2** (see Section 3.2.2), $G_2(h^\tau) = G_2(h^\beta)$ for all $\tau \in [\alpha, \beta]$. This, together with (38), yields

$$F(h^\tau) = F_\alpha(h^\tau) + F_{mid}(h^\tau) + G_2(h^\beta). \tag{46}$$

We can thus derive

$$\begin{aligned}
& err_P(h^\tau)(1+\epsilon/4) + \Delta \\
\text{(by (41) and (45))} \quad = \quad & (err_{P_\alpha}(h^\tau) + err_{P_{mid}}(h^\tau) + err_{P_{rest}}(h^\beta))(1+\epsilon/4) + \Delta \\
\geq \quad & (err_{P_\alpha}(h^\tau) + err_{P_{mid}}(h^\tau))(1+\epsilon/4) + err_{P_{rest}}(h^\beta) + \Delta \\
\text{(by (40))} \quad = \quad & (err_{P_\alpha}(h^\tau) + err_{P_{mid}}(h^\tau))(1+\epsilon/4) + \Delta_\alpha + \Delta_{mid} + G_2(h^\beta) \\
\text{(by (34) and (35))} \quad \geq \quad & F_\alpha(h^\tau) + F_{mid}(h^\tau) + G_2(h^\beta) \\
\text{(by (46))} \quad = \quad & F(h^\tau).
\end{aligned}$$

Similarly,

$$\begin{aligned}
& err_P(h^\tau)(1-\epsilon/4) + \Delta \\
\text{(by (41) and (45))} \quad = \quad & (err_{P_\alpha}(h^\tau) + err_{P_{mid}}(h^\tau) + err_{P_{rest}}(h^\beta))(1-\epsilon/4) + \Delta \\
\leq \quad & (err_{P_\alpha}(h^\tau) + err_{P_{mid}}(h^\tau))(1-\epsilon/4) + err_{P_{rest}}(h^\beta) + \Delta \\
\text{(by (40))} \quad = \quad & (err_{P_\alpha}(h^\tau) + err_{P_{mid}}(h^\tau))(1-\epsilon/4) + \Delta_\alpha + \Delta_{mid} + G_2(h^\beta) \\
\text{(by (34) and (35))} \quad \leq \quad & F_\alpha(h^\tau) + F_{mid}(h^\tau) + G_2(h^\beta) = F(h^\tau).
\end{aligned}$$

This completes the proof. □

3.4 A Concrete Algorithm for Problem 2

Instantiating our framework in Section 3.2 into an actual algorithm requires constructing the function F in Section 3.1, the function G_1 in Section 3.2, and the function G_2 in Section 3.2.2. We will explain how to achieve those purposes in this subsection. In doing so, we will factor in the consideration that the whole framework needs to succeed with probability at least $1 - \delta$. Denote by ℓ the number of recursion levels in our framework; the value of ℓ is $O(\log n)$ due to Proposition 4.

Constructing G_1 and G_2. Recall that 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:

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

The size of $|\mathbb{H}_{mon}(P)|$ is at most $|P| + 1$. Every monotone classifier has the same error on P as one of the classifiers in $\mathbb{H}_{mon}(P)$.

To build G_1, 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 $G_1(h)$ as

$$G_1(h) \quad = \quad \frac{|P|}{|S_1|} \cdot err_{S_1}(h). \tag{47}$$

By the discussion in Appendix A, for each $h \in \mathbb{H}_{mon}(P)$, the value $G_1(h)$ satisfies (24) with probability at least $1 - \frac{\delta}{3\ell \cdot (|P|+1)}$. As $|\mathbb{H}_{mon}(P)| = |P| + 1$, the function G_1 given in (47) satisfies (24) for all $h \in \mathbb{H}_{mon}(P)$ — also for all $h \in \mathbb{H}_{mon}$ — with probability at least $1 - \delta/(3\ell)$.

To build G_2, uniformly sample with replacement a set S_2 of $O(\frac{1}{\epsilon^2} \log \frac{|P|\ell}{\delta})$ elements from P_{rest}, where P_{rest} is given in (29). For each $h \in \mathbb{H}_{mon}$, define

$$G_2(h) \quad = \quad \frac{|P_{rest}|}{|S_2|} \cdot err_{S_2}(h). \tag{48}$$

An argument analogous to the one used earlier for G_1 shows that G_2 obeys (30) for all $h \in \mathbb{H}_{mon}$ with probability at least $1 - \delta/(3\ell)$. G_2 fulfills requirement **G2-2** because S_2 has no element in $[\alpha, \beta]$.

Constructing the Function F in Section 3.1. In our framework, the method in Section 3.1 is applied is to solve Problem 2 on P_α — defined in (27) — whose goal is to obtain a function F_α satisfying (32). To build F_α (i.e., the function F in Section 3.1 when $P = P_\alpha$), 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

$$F_\alpha(h) \quad = \quad \frac{|P_\alpha|}{|S_\alpha|} \cdot err_{S_\alpha}(h). \tag{49}$$

It satisfies (32) 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 build 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 now ready to prove Theorem 6 for $d = 1$. Let us examine our algorithm (combining Sections 3.1, 3.2, and 3.4) again and construct a coreset Z along the way.

If $|P| = 1$, our algorithm probes the only element $p \in P$. Accordingly, we set $Z = P$ and define $weight(p) = 1$. When $|P| \geq 2$, our algorithm acts differently in two scenarios.

- If function G_1 is consistently large, the target function F is decided in (26). In this case, we add to Z the entire sample set S_1 described in Section 3.4; for each element $p \in S_1$, define $weight(p) = |P|/|S_1|$.
- Otherwise, the function F is decided in (38), where function F_α is obtained from P_α and function F_{mid} is recursively obtained from P_{mid}. In this case, we first add to Z the entire sample set S_2 described in Section 3.4; for each element $p \in S_2$, define $weight(p) = |P_{rest}|/|S_2|$. Then, we add to Z the entire sample set S_α described in Section 3.4; for each element $p \in S_\alpha$, define $weight(p) = |P_\alpha|/|S_\alpha|$. The recursion on P_{mid} returns a coreset $Z_{mid} \subseteq P_{mid}$. We finish by adding Z_{mid} to Z.

The following pseudocode summarizes the above steps.

algorithm `Build-Coreset` (P)
1. **if** $|P| = 1$ **then** probe the only element $p \in P$ and set $Z = P$ with $weight(p) = 1$
2. **else** probe the sample set S_1 described in Section 3.4 /* this defines G_1; see (47) */
3. **if** G_1 is consistently large **then**
4. set $Z = S_1$ with $weight(p) = |P|/|S_1|$ for each $p \in S_1$
   else
5. probe the sample set S_2 described in Section 3.4 /* this defines G_2; see (48) */
6. set $Z = S_2$ with $weight(p) = |P_{rest}|/|S_2|$ for each $p \in S_2$
7. probe the sample set S_α described in Section 3.4 /* this defines F_α; see (49) */
8. set $weight(p) = |P_\alpha|/|S_\alpha|$ for each $p \in S_\alpha$ and then add S_α to Z
9. $Z_{mid} =$ `Build-Coreset` (P_{mid}) and add Z_{mid} to Z
10. **return** Z

The discussion in Section 3.4 asserts that `Build-Coreset` returns a set Z of $O(\frac{\log n}{\epsilon^2} \cdot \log \frac{n}{\delta})$ elements.

Lemma 10. *The function F we return for Problem 2 satisfies $F(h) = w\text{-}err_Z(h)$ for every $h \in \mathbb{H}_{mon}$.*

Proof. The claim obviously holds when $|P| = 1$. Assuming that the claim is true for all P with at most $m \geq 1$ elements, next we prove the correctness for $|P| = m + 1$. When G_1 is consistently large, $F(h) = G_1(h)$ and Z is simply the set S_1 in Section 3.4. Hence:

$$F(h) = G_1(h) = \frac{|P|}{|S_1|} \cdot err_{S_1}(h) = w\text{-}err_{S_1}(h) = w\text{-}err_Z(h).$$

When G_1 is not consistently large, the algorithm recursively processes P_{mid} whose size is strictly smaller than $|P|$ (Proposition 4). Suppose that the recursion returns a function F_{mid} and a coreset Z_{mid}. By the inductive assumption, $F_{mid}(h) = w\text{-}err_{Z_{mid}}(h)$. Recall that $F(h) = G_2(h) + F_\alpha(h) + F_{mid}(h)$ (see (38)) and the coreset constructed for P is $Z = S_\alpha \cup S_2 \cup Z_{mid}$ (review S_α and S_2 from Section 3.4). Hence:

$$F(h) \quad = \quad G_2(h) + F_\alpha(h) + F_{mid}(h) = \frac{|P_{rest}|}{|S_2|} \cdot err_{S_2}(h) + \frac{|P_\alpha|}{|S_\alpha|} \cdot err_{S_\alpha}(h) + F_{mid}(h)$$

$$= \quad w\text{-}err_{S_2}(h) + w\text{-}err_{S_\alpha}(h) + w\text{-}err_{Z_{mid}}(h) = w\text{-}err_Z(h).$$

This completes the proof. □

We therefore conclude that Theorem 6 holds for $d = 1$.

3.6 Arbitrary Dimensionalities

This subsection will prove Theorem 6 for any $d \geq 2$. As before, denote by n and w the size and the width of the input P, respectively. We start by computing a chain decomposition of P with w chains: $C_1, C_2, ..., C_w$; as explained in Section 2.2, such a chain decomposition definitely exists. It can be computed in time polynomial in d and n (see [43]) without any probing.

For every $i \in [w]$, we will 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

$$err_{C_i}(h) \cdot \left(1 - \frac{\epsilon}{4}\right) - \Delta_i \leq w\text{-}err_{Z_i}(h) \leq err_{C_i}(h) \cdot \left(1 - \frac{\epsilon}{4}\right) + \Delta_i \tag{50}$$

for every $h \in \mathbb{H}_{mon}$, where Δ_i is some unknown real value common to all $h \in \mathbb{H}_{mon}$. Once this is done, we can obtain

$$Z \quad = \quad \bigcup_{i=1}^{w} Z_i. \tag{51}$$

It must hold for every $h \in \mathbb{H}_{mon}$ that

$$err_P(h) \cdot \left(1 - \frac{\epsilon}{4}\right) - \Delta \leq w\text{-}err_Z(h) \leq err_P(h) \cdot \left(1 - \frac{\epsilon}{4}\right) + \Delta \tag{52}$$

where

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

remains unknown.

Finding Z_i for an arbitrary $i \in [w]$ 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 only 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 (23). Earlier, we have proved that Theorem 6 is correct for $d = 1$. We can thus apply the theorem 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 $1 - 1/\delta$, the aforementioned $Z_1, ..., Z_w$ can be produced with a total probing cost of

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

where the derivation used the fact $\sum_{i=1}^{w} |C_i| = n$. The same bound also applies to the size of Z in (51). This completes the whole proof of Theorem 6.

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

This section will focus on Problem 1 under $\epsilon = 0$, namely, the objective is to find an optimal monotone classifier. Naively, we can achieve the objective by probing all the elements in the input P. We will prove that this is already the best approach up to a constant factor, as stated in the theorem below.

Theorem 11. *For Problem 1, any algorithm promising to find an optimal classifier with probability over 2/3 must probe $\Omega(n)$ labels in expectation, where n is the number of elements in the input P. This is true even if the dimensionality d is 1, and the algorithm knows the optimal error k^* of P.*

The rest of the section serves as a proof of the theorem. Assuming n to be an even number, we construct a family $\mathbb{P}$ of n one-dimensional inputs. Every input of $\mathbb{P}$ has n elements, which are 1, 2, ..., n (every number represents a 1D point). The inputs of $\mathbb{P}$, however, differ in their label assignments. Specifically, every integer $i \in [n/2]$ defines two inputs in $\mathbb{P}$:

- $P_{-1}(i)$, where every odd (resp., even) number in $[n]$ carries label 1 (resp., -1). The only exception is the number $2i - 1$, which is assigned label -1;
- $P_1(i)$, where every odd (resp., even) number in $[n]$ carries label 1 (resp., -1). The only exception is the number $2i$, which is assigned label 1.

We will refer to $P_{-1}(i)$ and $P_1(i)$ as a *(-1)-input* and a *1-input*, respectively.

The constructed family $\mathbb{P} = \{P_{-1}(1), P_{-1}(2), ..., P_{-1}(n/2), P_1(1), P_1(2), ..., P_1(n/2)\}$ can also be understood in an alternative manner. Chop the elements $1, 2, ..., n$ into $n/2$ pairs $(1, 2), (3, 4), ..., (n-1, n)$. 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 exactly one *anomaly pair* $(x-1, x)$. If P is a (-1)-input, both $x-1$ and x are assigned label -1; otherwise, they are assigned label 1.

For each input $P \in \mathbb{P}$, an optimal monotone classifier has error $k^* = n/2 - 1$. Indeed, every monotone classifier has to misclassify at least one element in each normal pair of P. 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.

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

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

If $\mathcal{A}$ is a deterministic algorithm, then $\mathbf{Pr}[\mathcal{A} \text{ errs on } P]$ is either 0 or 1 for each $P \in \mathbb{P}$. Section 4.1 will prove the following lemma for such algorithms.

Lemma 12. *Fix any non-negative constant $c < 1$. When $n \geq \max\{4, 2/c\}$, the following holds for any deterministic algorithm $\mathcal{A}_{det}$: if $\textit{family-err}(\mathcal{A}_{det}) \leq cn/2$, then $\textit{family-cost}(\mathcal{A}_{det}) = \Omega(n^2)$.*

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

Corollary 13. *When $n \geq 4$, the following holds for any randomized algorithm $\mathcal{A}$: if $\textit{family-err}(\mathcal{A}) < n/3$, then $\mathbf{E}[\textit{family-cost}(\mathcal{A})] = \Omega(n^2)$.*

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

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

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

contradicting the definition of $\mathcal{A}$.

By Lemma 12, every accurate $\mathbb{A}_{det}$ must satisfy $\textit{family-cost}(A_{det}) = \Omega(n^2)$. Thus, $\mathbf{E}[\textit{family-cost}(\mathcal{A})] \geq \Omega(n^2) \cdot \mathbf{Pr}[\mathcal{A} \in \mathbb{A}_{acc}] = \Omega(n^2)$. □

The corollary implies Theorem 11. Indeed, if $\mathcal{A}$ guarantees finding an optimal classifier with probability over $2/3$ on any input, then $\textit{family-err}(\mathcal{A}) < |\mathbb{P}|/3 = n/3$. Thus, Corollary 13 tells us that $\mathbf{E}[\textit{family-cost}(\mathcal{A})] = \Omega(n^2)$ when $n \geq 4$. This means that the expected cost of $\mathcal{A}$ is $\Omega(n)$ on at least one input in $\mathbb{P}$ (recall that $\mathbb{P}$ has n inputs).

4.1 Proof of Lemma 12

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

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

Proof. As mentioned, the optimal error is $n/2 - 1$ for each input of $\mathbb{P}$. Recall that a 1D monotone classifier h has the form (23), which is parameterized by a value τ; next, we will denote the classifier as h^τ. We argue that no h^τ is optimal for both $P_{-1}(i)$ and $P_1(i)$. For this purpose, 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 $n/2 + 1$.

Thus, regardless of τ, the error of h^τ is non-optimal on either $P_{-1}(i)$ or $P_1(i)$. □

To simplify the proof of Lemma 12, we strengthen the power of $\mathcal{A}_{det}$ by giving it certain "free" labels. Specifically, 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 12 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, i.e., catching the only pair where the two elements share the same label. Once the anomaly pair is found, $\mathcal{A}_{det}$ can output an optimal classifier immediately because the labels in all the normal pairs are fixed. As a result, we can model $\mathcal{A}_{det}$ as a procedure that performs probing according to a pre-determined sequence: pair x_1, pair x_2, ..., pair x_t up to some integer $t \in [0, n/2]$. Specifically, for each $j \in [t-1]$, if pair x_j is an anomaly, the algorithm terminates after the probing of x_j; otherwise, it moves on to pair x_{j+1}. If all the t pairs have been probed but no anomaly is found, $\mathcal{A}_{det}$ always outputs a fixed classifier, denoted as h_{det}.

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

$$i \in \{1, 2, ..., n/2\} \setminus \{x_1, x_2, ..., x_t\} \tag{53}$$

the output of $\mathcal{A}_{det}$ must be h_{det} on both $P_{-1}(i)$ and $P_1(i)$. Proposition 5 asserts that h_{det} cannot be optimal for both $P_{-1}(i)$ and $P_1(i)$, meaning that $\mathcal{A}_{det}$ has to err on either $P_{-1}(i)$ or $P_1(i)$, which gives

$$\textit{family-err}(\mathcal{A}_{det}) \quad \geq \quad n/2 - t. \tag{54}$$

Regarding its cost, observe that $\mathcal{A}_{det}$ performs t probes for $P_{-1}(i)$ and $P_1(i)$ of every i satisfying (53), but $j \in [t]$ probes for $P_{-1}(x_j)$ and $P_1(x_j)$. Hence

$$\textit{family-cost}(\mathcal{A}_{det}) = 2t \cdot (n/2 - t) + 2\sum_{j=1}^{t} j = nt - t^2 - t. \tag{55}$$

If $\textit{family-err}(\mathcal{A}_{det})$ needs to be at most $cn/2$ (as demanded in Lemma 12), then by (54) t must be at least $\frac{n}{2}(1-c)$. On the other hand, for $t \geq \frac{n}{2}(1-c)$ and $n \geq 2/c$, we have

$$(55) \quad \geq \quad \frac{n^2}{4}(1-c^2) - \frac{n}{2}(1-c)$$

which is at least $n^2(1-c^2)/8$ for $n \geq 4$. This completes the proof of Lemma 12.

5 A Lower Bound for Constant Approximation Ratios

We now proceed to study the hardness of approximation for Problem 1. This section's main result is:

Theorem 14. *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*

$$n \quad = \quad n' + 2k + 2ckn' \tag{56}$$

$$w \quad = \quad w' + \mathbb{1}_{k \geq 1}. \tag{57}$$

For Problem 1, we can construct a family $\mathbb{P}$ of inputs where each input has size n, width w, and optimal error $k^* = k$, such that any randomized algorithm, which guarantees an expected error at most ck^*, must entail expected cost $\Omega(w' \operatorname{Log} \frac{n'}{w'})$ on at least one input of $\mathbb{P}$. Here, $\Omega(.)$ hides a constant that does not depend on n', w', k, and c. The claim holds true even if the algorithm knows the value of k^*.

The theorem is particularly useful when the approximation ratio c is a constant. To see this, first note that when $k = 0$ (the realizable case), the theorem gives a lower bound of $\Omega(w \operatorname{Log} \frac{n}{w})$ on the expected cost. For $k \geq 1$ (the non-realizable case), we always have $n' + 2k \leq n/2$, because of which

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

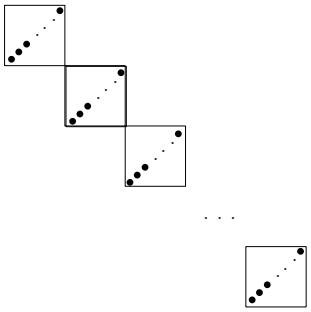


Fig. 4. A hard realizable input for Problem 1: w boxes each with n/w points

In this case, Theorem 14 implies a lower bound of $\Omega(w \operatorname{Log} \frac{n}{k^* w})$ on the expected cost when w is sufficiently large.

We will first prove the theorem for $k = 0$ in Section 5.1 and then for $k \geq 1$ in Section 5.2.

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* from each other if no point in B_1 dominates any point in B_2 and vice versa.

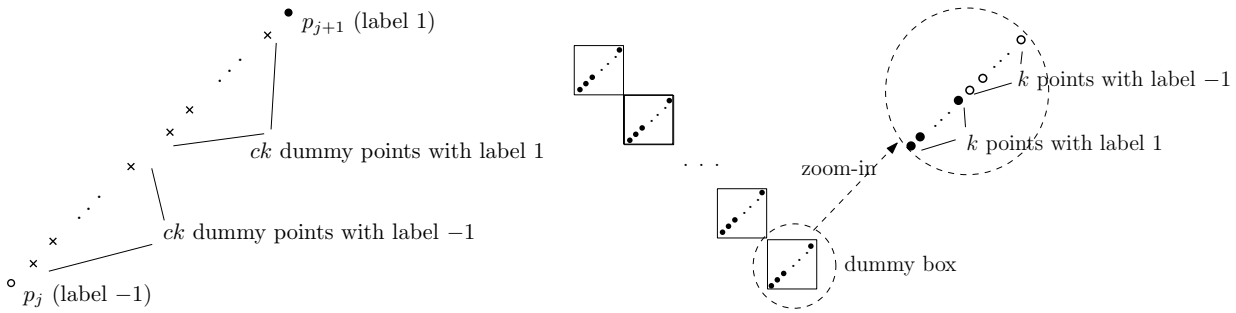
When $k = 0$, an algorithm that guarantees an expected error at most $ck^* = 0$ must always find an optimal classifier. To prove Theorem 14 in this case, we construct hard inputs as follows. Let $n' \geq 2$ and $w' \geq 1$ be integers such that n' is a multiple of w'. Let $B_1, B_2, ..., B_{w'}$ be arbitrary 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 that has width w'; see Figure 4 for an illustration. Label assignment is done for each box independently, subject to the constraint that P is monotone. In each box, there are $1 + \frac{n'}{w'}$ ways to do the assignment: for each $i \in [0, \frac{n'}{w'}]$, 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 $(1 + \frac{n'}{w'})^{w'}$ labeled point sets, each of which serves as an input to Problem 1.

Recall from Section 1.1 that 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 be able to 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 $(1 + \frac{n'}{w'})^{w'}$. The *average cost* of $\mathcal{A}_{det}$ — defined as the average of its costs on all the inputs of $\mathbb{P}_{n',w'}$ — equals the average depth of the leaves in $\mathcal{T}$. A binary tree with at least $(1 + \frac{n'}{w'})^{w'}$ leaves must have an average depth of $\Omega(w' \operatorname{Log} \frac{n'}{w'})$. Hence, $\mathcal{A}_{det}$ must have average cost $\Omega(w \operatorname{Log} \frac{n}{w})$.

By Yao's minimax theorem [37], any randomized algorithm that is always correct for $k = 0$ must entail $\Omega(w' \operatorname{Log} \frac{n'}{w'})$ expected cost on at least one input of $\mathbb{P}_{n',w'}$, as claimed in Theorem 14 (for $k = 0$).

5.2 The Non-Realizable Case

This subsection serves as a proof of Theorem 14 for $k \geq 1$.



(a) Dummy points in a non-dummy box (b) Dummy points in the dummy box

Fig. 5. Adding dummy points for a Las Vegas lower bound

Algorithms with Guessing Power. We first strengthen the power of a deterministic algorithm $\mathcal{A}_{det}$. As before, $\mathcal{A}_{det}$ 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, there are 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 almighty guru whether $err_P(h)$ is at most a certain value fixed at this node. On a "yes" answer from the guru, $\mathcal{A}_{det}$ descends to the left child, which must be a leaf returning h. On a "no" answer, $\mathcal{A}_{det}$ branches right and continues.

We charge one unit of cost to every probe or guess node. A randomized algorithm is still modeled as a function that maps a random-bit sequence to a deterministic algorithm. Almighty gurus do not exist in reality. However, a lower bound on such empowered algorithms must also hold on algorithms that use probe nodes only.

Let $n' \geq 2$ and $w' \geq 1$ be arbitrary integers such that n' is a multiple of w'. The argument in Section 5.1 has essentially proved that any deterministic algorithm in the form of a binary decision tree must have an average cost of $\Omega(w' \operatorname{Log} \frac{n'}{w'})$ over the inputs of $\mathbb{P}_{n',w'}$. Hence, this is also true for a deterministic algorithm with guess nodes. By Yao's minimax theorem, any randomized algorithm with guess nodes must incur $\Omega(w' \operatorname{Log} \frac{n'}{w'})$ expected cost on at least one input of $\mathbb{P}(n', w')$ if it always returns an optimal classifier on the inputs of $\mathbb{P}(n', w')$.

A Las Vegas Lower Bound. Let n', k', k, c, n, and w be as described in the statement of Theorem 14. Denote by $\mathcal{A}$ a randomized algorithm (with guessing power) that, when executed on an input having size n, width w, and optimal 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 will show that $J_{LV} = \Omega(w' \operatorname{Log} \frac{n'}{w'})$.

Given an input $P' \in \mathbb{P}_{n',w'}$, we construct a set P of labeled points in several steps. As the first step, add all the points of P' to P. Recall that the points of P' are inside w' boxes $B_1, B_2, ..., B_{w'}$, each of which has n'/w' points; see Figure 4. For each $i \in [w']$, let $p_1, p_2, ..., p_{n'/w'}$ be the points already in B_i, sorted in

ascending order of y-coordinate. For each $j \in [\frac{n'}{w'} - 1]$, place $2ck$ dummy points on the main diagonal of B_i between p_j and p_{j+1}. Assign labels to those dummy points as follows:

- If $label(p_j) = label(p_{j+1})$, set the labels of all $2ck$ dummy points to $label(p_j)$.
- Otherwise, we must have $label(p_j) = -1$ and $label(p_{j+1}) = 1$ (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 5a.

Furthermore, add ck dummy points to P between the bottom-left corner of B_i and p_1. Set their labels to -1 if $label(p_1) = -1$, 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 $label(p_{n'/w'}) = 1$, or -1 otherwise. Finally, create a dummy box that is independent from all of $B_1, ..., B_{n'/w'}$. 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 5b. 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 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 $B_1, ..., B_{n'/w'}$ 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 $p \in P$ (remember that such p originated from P'). Otherwise, suppose that $label(p) = -1$ 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 $err_P(h) \geq ck + 1$, contradicting the fact that $\mathcal{A}$ guarantees an error at most ck^*. A symmetric argument rules out the possibility that $label(p) = 1$ 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 $\Omega(w' \operatorname{Log} \frac{n'}{w'})$.

A Monte-Carlo Lower Bound. Again, let n', k', k, c, n, and w be as described in Theorem 14. Let $\mathcal{A}$ be a randomized algorithm (with guessing power) that, when executed on an input P with size n, width w, and optimal error $k^* = k$, always guarantees

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

We will show that $J_{MC} = \Omega(w' \operatorname{Log} \frac{n'}{w'})$, which will complete the proof of Theorem 14.

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 can 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 almighty guru whether $err_P(h) \leq 4ck^*$. If so, return h. In all other situations (i.e., the guru

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 $J_{MC} = \Omega(w' \operatorname{Log} \frac{n'}{w'})$.

6 A Lower Bound for Arbitrary Approximation Ratios

We now continue our study on the approximation hardness of Problem 1 in the regime where ϵ can be arbitrarily small. Our main result in this section is:

Theorem 15. *Let ϵ be an arbitrary value satisfying $0 < \epsilon \leq 1/10$. Fix any integers $n \geq 1$ and $w \geq 1$ such that n is a multiple of w, and $n \geq \max\{90, \frac{w}{\epsilon^2} \ln n\}$. Suppose that $\mathcal{A}$ is an algorithm for Problem 1 under $d = 2$ that guarantees an expected error of $(1+\epsilon)k^*$ on any input P of size n where k^* is the optimal error of P. Then, the expected cost of $\mathcal{A}$ must be $\Omega(w/\epsilon^2)$ on at least one input where w is the width of P, and $\Omega(.)$ hides a constant that does not depend on ϵ, n, w, and c.*

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

$$\gamma = 9\epsilon \tag{58}$$
$$\mu_1 = (1-\gamma)/2 \tag{59}$$
$$\mu_2 = (1+\gamma)/2 \tag{60}$$
$$M = \ln(9/8) \cdot (1-\gamma^2)/\gamma^2. \tag{61}$$

Let $\mathcal{A}$ be an algorithm ensuring an expected cost at most $wM/6$ when executed on an input P of size n and width k. Careful calculation shows that when $\epsilon \leq 1/10$,

$$\frac{wM}{6} > \frac{1}{4200} \frac{w}{\epsilon^2}. \tag{62}$$

Let $r(P)$ represent the approximation ratio of the classifier $h_{\mathcal{A}}$ output by $\mathcal{A}$, namely,

$$r(P) = err_P(h_{\mathcal{A}})/k^*. \tag{63}$$

Our goal is to prove that $\mathbf{E}[r(P)] > 1+\epsilon$, which along with (62) will then imply that $\Omega(w/\epsilon^2)$ probes are needed in expectation to guarantee an expected error of $(1+\epsilon)k^*$.

Let $x_1, x_2, ..., x_w$ be w distinct locations in $\mathbb{R}^2$ such that no location dominates another. Henceforth, we will fix P to be a multi-set 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 described below stochastically generates their labels and measures the cost and inaccuracy of $\mathcal{A}$ over the resulting P:

RP-1

1. $\boldsymbol{\mu}$ = a w-dimensional vector sampled from $\{\mu_1, \mu_2\}^m$ uniformly at random
2. **for** every element $p \in P$ **do** /* suppose that p is at x_i */
3. assign p label 1 with probability $\boldsymbol{\mu}[i]$ or -1 with probability $1 - \boldsymbol{\mu}[i]$
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. $R_1 = err_P(h_{\mathcal{A}})/k^*$ where k^* is the optimal error of P
6. **return** (Λ_1, R_1)

Recall that $\mathcal{A}$ ensures $\mathbf{E}[\Lambda_1] \leq wM/6$. Our objective is to argue that R_1 has a large expectation. We will achieve the purpose by relating RP-1 to another random process:

RP-2

1. $\boldsymbol{\mu}$ = a w-dimensional vector sampled from $\{\mu_1, \mu_2\}^m$ 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 x_i */
4. assign p label 1 with probability $\boldsymbol{\mu}[i]$ or -1 with probability $1 - \boldsymbol{\mu}[i]$
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 $\boldsymbol{\mu}[i]$ or -1 with probability $1 - \boldsymbol{\mu}[i]$
8. $R_2 = err_P(h_{\mathcal{A}})/k^*$ where k^* is the optimal error of P
9. **return** (Λ_2, R_2)

Lemma 16. $\mathbf{E}[\Lambda_1] = \mathbf{E}[\Lambda_2]$ *and* $\mathbf{E}[R_1] = \mathbf{E}[R_2]$.

Proof. We will 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 $j \in \{1, 2\}$:

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

Let us concentrate on an arbitrary leaf v. Let $p_1, p_2, ..., p_t$ 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 "$label(p_i) = l_i$ 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, $R_j = err_P(h_{\mathcal{A}})/k^*$ is determined by the labels of the elements in $P \setminus \{p_1, ..., p_t\}$, whose distributions are the same in RP-1 and RP-2. Hence, $\mathbf{E}[R_j \mid \mathcal{A}$ finishes at v in RP-$j]$ is identical for $j = 1$ and 2. It thus follows that $\mathbf{E}[R_1] = \mathbf{E}[R_2]$.

As a randomized algorithm is a distribution over a family of deterministic algorithms, our above deterministic analysis implies that $\mathbf{E}[R_1] = \mathbf{E}[R_2]$ holds for a randomized $\mathcal{A}$ as well. □

Because $\mathbf{E}[\Lambda_1] \leq wM/6$, we immediately have

$$\mathbf{E}[\Lambda_2] \leq wM/6. \tag{64}$$

The next subsection will utilize (64) to prove:

Lemma 17. $\mathbf{E}[R_2] > 1 + \epsilon$ *when* $\epsilon \leq 1/10$ *and* $n \geq \max\{90, \frac{w}{\epsilon^2} \ln n\}$.

The above lemma indicates that, when n meets the stated condition, the algorithm $\mathcal{A}$ cannot guarantee $\mathbf{E}[r(P)] \leq 1 + \epsilon$ on every input P, where $r(P)$ is defined in (63). Otherwise, R_1, defined at Line 5 of RP-1, must have an expectation at most $1 + \epsilon$, which by Lemma 16 tells us $\mathbf{E}[R_2] \leq 1 + \epsilon$, giving a contradiction. This concludes the proof of Theorem 15.

6.1 Proof of Lemma 17

Our proof was inspred by an argument in [7] for establishing the lower bound (6) on agnostic active learning. The main technicality arises from adapting the argument to the scenario where precision is measured on a finite number of points (in [7], precision is measured over a distribution). The following fundamental result from [3] will be useful later.

Lemma 18. *Define* μ *to be a random variable that equals* μ_1 *or* μ_2 *— see* (59) *and* (60) *— each with probability* 1/2. *Let* $\Sigma = (X_1, ..., X_m)$ *be a sequence of i.i.d. samples such that* $\mathbf{Pr}[X_i = 1] = \mu$ *and* $\mathbf{Pr}[X_i = -1] = 1 - \mu$. *If* $m < M$ *where* M *is given in* (61), *then no (deterministic or randomized) algorithm can correctly infer* μ *from* Σ *with probability over* 2/3.

Proof. Every algorithm that aims to infer μ from Σ can be regarded as a distribution over a family of functions mapping $\{-1, 1\}^m$ to $\{\mu_1, \mu_2\}$. By [3, Lemma 5.1], under the condition $m < M$ no function $\{-1, 1\}^m$ to $\{\mu_1, \mu_2\}$ can correctly output μ with a probability over 2/3. The lemma then follows from the law of total probability. □

For each $i \in [w]$, define a random variable according to RP-2:

$$L_i = \begin{cases} 1 & \text{if } \mathcal{A} \text{ probes less than } M \text{ points at location } x_i \text{ in RP-2} \\ 0 & \text{otherwise} \end{cases}$$

We say that the value i is *light* if $\mathbf{Pr}[L_i = 1] \geq 2/3$ or *heavy* otherwise.

Lemma 19. *There are more than* $w/2$ *light values of* i.

Proof. For each $i \in [w]$, let M_i be the number of points at x_i probed by $\mathcal{A}$ in RP-2. For every heavy $i \in [w]$, $\mathbf{Pr}[L_i = 0] > 1/3$ and hence $\mathbf{E}[M_i] \geq M_i \cdot \mathbf{Pr}[L_i = 0] > M/3$. If at least $w/2$ heavy values exist in $[w]$, then $\sum_{i=1}^{w} \mathbf{E}[M_i] > \frac{w}{2}\frac{M}{3} = wM/6$, meaning that in RP-2 the algorithm $\mathcal{A}$ probes over $wM/6$ elements in expectation, which contradicts (64). Hence, $[w]$ must have less than $w/2$ heavy values, thus establishing the claim. □

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

Otherwise, $K_i = 0$. Note that K_i is a random variable decided by RP-2.

Lemma 20. *If $i \in [w]$ is light, then $\mathbf{E}[K_i] > 1/3$.*

Proof. Recall that $\boldsymbol{\mu}[i]$ is taken from $\{u_1, u_2\}$ uniformly at random. We can view $h_{\mathcal{A}}(x_i)$ as the guess of algorithm $\mathcal{A}$ about $\boldsymbol{\mu}[i]$ based on the labels of the points probed at location x_i (the labels of points at other locations provides no information about $\boldsymbol{\mu}[i]$). Specifically, we consider that $\mathcal{A}$ guesses $\boldsymbol{\mu}[i] = \mu_2$ if $h_{\mathcal{A}}(x_i) = 1$ or $\boldsymbol{\mu}[i] = \mu_1$ if $h_{\mathcal{A}}(x_i) = -1$. Thus, $K_i = 1$ if and only if the guess of $\mathcal{A}$ is wrong. Lemma 18 indicates that, when $L_i = 1$ (i.e., $\mathcal{A}$ probes less than M points at x_i), the guess $\mathcal{A}$ is wrong with probability over $1/3$. Thus:

$$\mathbf{Pr}[K_i = 1] \geq \mathbf{Pr}[K_i = 1 \mid L_i = 1] \cdot \mathbf{Pr}[L_i = 1] > \frac{1}{3} \cdot \frac{2}{3} = \frac{4}{9}.$$

The claim in the lemma now follows. □

Recall that RP-2 randomly chooses a vector $\boldsymbol{\mu}$ at Line 1. Accordingly, for each $i \in [w]$, we define its *good i-label* to be

- 1 if $\boldsymbol{u}[i] = \mu_2$;
- -1 otherwise.

We use the term *bad i-label* to refer to the label in $\{-1, 1\}$ different from the good i-label. To each of the n/w points at location x_i, RP-2 assigns label 1 with probability $\boldsymbol{\mu}[i]$ and label -1 with probability $1 - \boldsymbol{\mu}[i]$. At location 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 multi-set 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 21. *Let c be a sufficiently large constant. When $n \geq \frac{w}{\epsilon^2} \ln n$, the probability for P to be intended is at least $1 - 1/n$.*

Proof. Let us focus on an arbitrary $i \in [w]$. Denote by $X_1, X_2, ..., X_{n/w}$ the labels of the n/w elements of P at location x_i. For each $j \in [n/w]$, $\mathbf{Pr}[X_j = \text{bad } i\text{-label}] = \frac{1}{2} - \frac{\gamma}{2}$. Let Y_i be how many elements of P at location x_i actually receive the bad i-label; hence, $\mathbf{E}[Y_i] = \frac{n}{w}(\frac{1}{2} - \frac{\gamma}{2})$. Set $t = \frac{\gamma}{2(1-\gamma)}$ such that $\mathbf{E}[Y_i] \cdot (1 + t) = \frac{1}{2} - \frac{\gamma}{4}$. By Chernoff bound (68),

$$\mathbf{Pr}\left[Y_i \geq \frac{1}{2} - \frac{\gamma}{4}\right] \leq \exp\left(-\frac{t^2}{2+t}\,\mathbf{E}[Y_i]\right) \leq \exp\left(-\frac{\gamma^2 n}{16w}\right) = \exp\left(-\frac{81\epsilon^2 n}{16w}\right)$$

which is at most $1/n^2$ when $n \geq \frac{w}{\epsilon^2} \ln n$. As $w \leq n$, the probability of "$Y_i \geq \frac{1}{2} - \frac{\gamma}{4}$ for at least one $i \in [w]$" is at most $1/n$. It thus follows that P is intended with probability at least $1 - 1/n$. □

Lemma 22. *When $n \geq 9$, $\mathbf{E}[\sum_{i=1}^{w} K_i \mid P \text{ intended}] > w/8$.*

Proof. For each light $i \in [w]$, we argue that $\mathbf{E}[K_i \mid P \text{ intended}] \geq 1/4$ when $n \geq 9$. Otherwise,

$$\begin{aligned}\mathbf{E}[K] &= \mathbf{E}[K_i \mid P \text{ intended}] \cdot \mathbf{Pr}[P \text{ intended}] + \mathbf{E}[K_i \mid P \text{ not intended}] \cdot \mathbf{Pr}[P \text{ not intended}] \\ &\leq (1/4) \cdot (1 - 1/n) + 1 \cdot (1/n)\end{aligned}$$

which is at most $1/3$ when $n \geq 9$. This, however, contradicts Lemma 20. The target claim then follows from the fact that $[w]$ has more than $w/2$ light values of $i \in [w]$ (see Lemma 19). □

Unless otherwise stated, the subsequent discussion assumes that P is intended. For each $i \in [w]$, define I_i as the number of points at location x_i receiving the good i-label; thus, $n/w - I_i$ points at x_i receive the bad i-label. The optimal classifier h^* for P should map x_i to the good label. As P is intended, we have $I_i \geq \frac{n}{w}(\frac{1}{2} + \frac{\gamma}{4})$ for all $i \in [w]$. Therefore:

$$k^* = \sum_{i=1}^{w} \left(\frac{n}{w} - I_i \right) \leq \sum_{i=1}^{w} \left(\frac{n}{w} - \frac{n}{w} \left(\frac{1}{2} + \frac{\gamma}{4} \right) \right) = n \left(\frac{1}{2} - \frac{\gamma}{4} \right). \tag{65}$$

For each $i \in [w]$, by how K_i and the good i-label are defined, $h^*(x_i) = h_{\mathcal{A}}(x_i)$ if and only if $K_i = 0$. Furthermore, if $h^*(x_i) \neq h(x_i)$, then $h(x_i)$ misclassifies I_i points, which is $2I_i - \frac{n}{w}$ more than the number $\frac{n}{w} - I_i$ of points misclassified by h^* at x_i. Hence:

$$err_P(h) = k^* + \sum_{i=1}^{w} K_i \cdot \left(2I_i - \frac{n}{w} \right).$$

Thus

$$err_P(h) - k^* = \sum_{i=1}^{w} K_i \cdot \left(2I_i - \frac{n}{w} \right) \geq \sum_{i=1}^{w} K_i \cdot \left(2 \cdot \frac{n}{w} \left(\frac{1}{2} + \frac{\gamma}{4} \right) - \frac{n}{w} \right) = \frac{n\gamma}{2w} \sum_{i=1}^{w} K_i \tag{66}$$

Therefore

$$\begin{aligned}
\mathbf{E}[R_2 \mid P \text{ intended}] &= \mathbf{E}[err_P(h)/k^* \mid P \text{ intended}] \\
&= 1 + \mathbf{E}\left[\frac{err_P(h) - k^*}{k^*} \,\middle|\, P \text{ intended} \right] \\
\text{(by (65) and (66))} \quad &\geq 1 + \mathbf{E}\left[\frac{n\gamma/(2w)}{n(1/2 - \gamma/4)} \sum_{i=1}^{w} K_i \,\middle|\, P \text{ intended} \right] \\
\text{(by Lemma 22)} \quad &> 1 + \frac{\gamma/16}{1/2 - \gamma/4} \\
&\geq 1 + \gamma/8 = 1 + (9/8)\epsilon.
\end{aligned}$$

We now conclude that $\mathbf{E}[R_2] \geq \mathbf{E}[R_2 \mid P \text{ intended}] \cdot \mathbf{Pr}[P \text{ intended}] \geq (1 + \frac{9}{8}\epsilon) \cdot (1 - 1/n)$ which is greater than $1 + \epsilon$ for $n \geq 90$ and $\epsilon \leq 1/10$. This completes the proof of Lemma 17.

7 Conclusions

This article has provided a comprehensive study of monotone classification in $\mathbb{R}^d$ with relative error guarantees, where the objective is to minimize the label-probing cost while finding a classifier whose error can be higher than the optimal error k^* by at most a $1 + \epsilon$ multiplicative factor. Our findings delinerate 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 $O(w \operatorname{Log} \frac{n}{w})$ 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 $O(\frac{w}{\epsilon^2} \operatorname{Log} \frac{n}{w} \cdot \log n)$. These are complemented by lower

bounds of $\Omega(w \operatorname{Log} \frac{n}{(1+k^*)w})$ 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 $\epsilon > 0$.

Appendix

A Concentration Bounds

Let $X_1, X_2, ..., X_t$ be t independent Bernoulli random variables with $\mathbf{Pr}[X_i = 1] = \mu$ for each $i \in [t]$ (and hence $\mathbf{Pr}[X_i = 0] = 1 - \mu$). The following are two standard forms of Chernoff bounds [37]:

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

$$\mathbf{Pr}\left[\left|\mu - \frac{1}{t}\sum_{i=1}^{t} X_i\right| \geq \gamma\mu\right] \quad \leq \quad 2\exp\left(-\frac{\gamma^2 t\mu}{3}\right); \tag{67}$$

- for any $\gamma \geq 0$:

$$\mathbf{Pr}\left[\frac{1}{t}\sum_{i=1}^{t} X_i \geq (1+\gamma)\mu\right] \quad \leq \quad \exp\left(-\frac{\gamma^2}{2+\gamma}t\mu\right). \tag{68}$$

In addition, we prove:

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

$$\mathbf{Pr}\left[\left|\mu - \frac{1}{t}\sum_{i=1}^{t} X_i\right| \geq \phi\right] \quad \leq \quad \delta \tag{69}$$

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

$$\begin{aligned}\mathbf{Pr}\left[\left|\mu - \frac{1}{t}\sum_{i=1}^{t} X_i\right| \geq \phi\right] \quad &= \quad \mathbf{Pr}\left[\left|\mu - \frac{1}{t}\sum_{i=1}^{t} X_i\right| \geq \frac{\phi}{\mu}\cdot\mu\right] \\ \text{(by (67))} \quad &\leq \quad 2\exp\left(-\frac{1}{3}\left(\frac{\phi}{\mu}\right)^2 t\mu\right)\end{aligned}$$

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

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

$$\begin{aligned}&\mathbf{Pr}\left[\left|\mu - \frac{1}{t}\sum_{i=1}^{t} X_i\right| \geq \phi\right] \\ = \quad &\mathbf{Pr}\left[\frac{1}{t}\sum_{i=1}^{t} X_i \geq \phi + \mu\right] = \mathbf{Pr}\left[\frac{1}{t}\sum_{i=1}^{t} X_i \geq \left(1 + \frac{\phi}{\mu}\right)\mu\right]\end{aligned}$$

$$\text{(by (68))} \quad \leq \quad \exp\left(-\frac{(\phi/\mu)^2}{2+\phi/\mu}\cdot t\mu\right) = \exp\left(\frac{t\phi^2}{2\mu+\phi}\right)$$

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

Consider the settings 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 further 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 and Disagreement Coefficient of Monotone Classifiers

This section will discuss the VC-dimension and disagree coefficient of the class $\mathbb{H}_{mon}$ of monotone classifiers. We will focus on the context of Problem 1, in which P is a multi-set of points in $\mathbb{R}^d$ where each element $p \in P$ is associated with a label from $\{-1, 1\}$. The VC-dimension of $\mathbb{H}_{mon}$ on P is defined as the size of the largest subset $S \subseteq P$ that can be *shattered* by $\mathbb{H}_{mon}$, meaning that for any function $f : S \to \{-1, 1\}$, there exists a classifier $h \in \mathbb{H}_{mon}$ such that $h(p) = f(p)$ for every $p \in S$.

Lemma 24. *The VC-dimension of $\mathbb{H}_{mon}$ on P is the width w of P.*

Proof. By Dilworth's Theorem [20], there exists a subset $S \subseteq P$ such that $|S| = w$ and there are no two distinct elements $p, q \in S$ such that $p \succeq q$. It is clear that S can be shattered by $\mathbb{H}_{mon}$. On the other hand, Dilworth's Theorem also shows that, among any $w+1$ elements from P, we can always find two elements p, q such that $p \succeq q$. The subset of P comprising those $w+1$ elements cannot be shattered because no monotone classifier h can satisfy $h(p) = -1$ but $h(q) = 1$. It thus follows that the VC-dimension of $\mathbb{H}_{mon}$ on P is exactly w. □

The rest of the section will focus on disagreement coefficients. Given a set $\mathcal{H} \subseteq \mathbb{H}_{mon}$ 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, $err_P(h^*) = k^*$. Given a real value $\rho \in (0, 1]$, define the *ball* — denoted as $B(h^*, \rho)$ — as the set of classifiers $h \in \mathbb{H}_{mon}$ 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 $\mathbb{H}_{mon}$ under the uniform distribution over P is defined as:

$$\theta \quad = \quad \max\left\{1, \sup_{\rho\in(\frac{k^*}{n},1]} \frac{|\text{DIS}(B(h^*,\rho))|}{\rho\cdot n}\right\}. \tag{70}$$

Lemma 25. *The value of θ is at most $2w$.*

Proof. Consider any $\rho \in (k^*/n, 1]$. We will prove that $|\text{DIS}(B(h^*, \rho))| \leq 2w\rho n$. The claim then follows from the definition of θ in (70).

As defined in Section 2.2, a subset $S \subseteq P$ is a chain if we can arrange the elements of S into a sequence $p_1 \preceq p_2 \preceq ... \preceq p_{|S|}$. By Dilworth's Theorem [20], we can decompose P into w disjoint chains $C_1, C_2, ..., C_w$. For each $i \in [w]$, use the optimal classifier h^* to break the chain C_i into two disjoint subsets:

- $C_i^1 = \{p \in C_i \mid h^*(p) = 1\}$, and
- $C_i^{-1} = \{p \in C_i \mid h^*(p) = -1\}$.

We will prove that $\text{DIS}(B(h^*, \rho))$ can contain at most ρn elements from each of C_i^1 and C_i^{-1}. This will then establish the fact $|\text{DIS}(B(h^*, \rho))| \leq 2w\rho n$. Next, we will explain why $\text{DIS}(B(h^*, \rho)) \cap C_i^1$ has at most ρn elements; a symmetric argument works on C_i^{-1}.

Set $m = |C_i^1|$ and linearize the elements of C_i^1 as $p_1 \preceq p_2 \preceq ... \preceq p_m$. If a monotone classifier h maps p_i to -1 for some $i \in [m]$, then it must also map p_j to -1 for every $j \in [i]$. Hence, h and h^* can only differ on a prefix of the sequence $p_1, p_2, ..., p_m$. Furthermore, as $h \in B(h^*, \rho)$, the prefix must have a length at most ρn. It thus follows that every classifier in $B(h^*, \rho)$ must map the whose sequence to 1 except possibly for the first ρn elements. By definition of $\text{DIS}(B(h^*, \rho))$, we can assert that $|\text{DIS}(B(h^*, \rho)) \cap C_i^1| \leq \rho n$. □

We close the section by giving a multi-set 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 $x_1, x_2, ..., x_w$ 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 error $k^* = w$. Define h^* to be the optimal classifier that maps the entire $\mathbb{R}^2$ to 1.

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

References

[1] Meysam Alishahi and Jeff M. Phillips. 2024. No Dimensional Sampling Coresets for Classification. In *Proceedings of International Conference on Machine Learning (ICML)*.

[2] Stanislav Angelov, Boulos Harb, Sampath Kannan, and Li-San Wang. 2006. Weighted isotonic regression under the L_1 norm. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. ACM Press, 783–791.

[3] Martin Anthony and Peter L. Bartlett. 1999. *Neural Network Learning: Theoretical Foundations*. Cambridge University Press.

[4] Arvind Arasu, Michaela Götz, and Raghav Kaushik. 2010. On active learning of record matching packages. In *Proceedings of ACM Management of Data (SIGMOD)*. 783–794.

[5] Maria-Florina Balcan, Alina Beygelzimer, and John Langford. 2009. Agnostic active learning. *Journal of Computer and System Sciences (JCSS)* 75, 1 (2009), 78–89.

[6] Kedar Bellare, Suresh Iyengar, Aditya G. Parameswaran, and Vibhor Rastogi. 2013. Active Sampling for Entity Matching with Guarantees. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 7, 3 (2013), 12:1–12:24.

[7] Alina Beygelzimer, Sanjoy Dasgupta, and John Langford. 2009. Importance weighted active learning. In *Proceedings of International Conference on Machine Learning (ICML)*. 49–56.

[8] Vladimir Braverman, Dan Feldman, Harry Lang, Adiel Statman, and Samson Zhou. 2021. Efficient Coreset Constructions via Sensitivity Sampling. In *Asian Conference on Machine Learning (ACML)*, Vol. 157. 948–963.
[9] Nader H. Bshouty and Christino Tamon. 1996. On the Fourier Spectrum of Monotone Functions. *Journal of the ACM (JACM)* 43, 4 (1996), 747–770.
[10] Deeparnab Chakrabarty and C. Seshadhri. 2013. Optimal bounds for monotonicity and lipschitz testing over hypercubes and hypergrids. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*. 419–428.
[11] Deeparnab Chakrabarty and C. Seshadhri. 2014. An Optimal Lower Bound for Monotonicity Testing over Hypergrids. *Theory of Computing* 10 (2014), 453–464.
[12] Ke Chen. 2009. On Coresets for k-Median and k-Means Clustering in Metric and Euclidean Spaces and Their Applications. *SIAM Journal of Computing* 39, 3 (2009), 923–947.
[13] Xi Chen, Anindya De, Yizhi Huang, Yuhao Li, Shivam Nadimpalli, Rocco A. Servedio, and Tianqi Yang. 2025. Relative-error monotonicity testing. In *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. 373–402.
[14] Peter Christen, Dinusha Vatsalan, and Qing Wang. 2015. Efficient Entity Resolution with Adaptive and Interactive Training Data Selection. In *Proceedings of International Conference on Management of Data (ICDM)*. 727–732.
[15] Xu Chu, Ihab F. Ilyas, and Paraschos Koutris. 2016. Distributed Data Deduplication. *Proceedings of the VLDB Endowment (PVLDB)* 9, 11 (2016), 864–875.
[16] Anirban Dasgupta, Petros Drineas, Boulos Harb, Ravi Kumar, and Michael W. Mahoney. 2009. Sampling Algorithms and Coresets for ℓ_p Regression. *SIAM Journal of Computing* 38, 5 (2009), 2060–2078.
[17] Sanjoy Dasgupta. 2005. Coarse sample complexity bounds for active learning. In *Proceedings of Neural Information Processing Systems (NIPS)*. 235–242.
[18] Sanjoy Dasgupta, Daniel J. Hsu, and Claire Monteleoni. 2007. A general agnostic active learning algorithm. In *Proceedings of Neural Information Processing Systems (NIPS)*. 353–360.
[19] Sanjoy Dasgupta, Adam Tauman Kalai, and Claire Monteleoni. 2009. Analysis of Perceptron-Based Active Learning. *Journal of Machine Learning Research (JMLR)* 10 (2009), 281–299.
[20] Robert P. Dilworth. 1950. A Decomposition Theorem for Partially Ordered Sets. *The Annals of Mathematics* 51, 1 (1950), 161–166.
[21] Vasilis Efthymiou, George Papadakis, George Papastefanatos, Kostas Stefanidis, and Themis Palpanas. 2017. Parallel meta-blocking for scaling entity resolution over big heterogeneous data. *Information Systems* 65 (2017), 137–157.
[22] Dan Feldman and Michael Langberg. 2011. A unified framework for approximating and clustering data. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*. 569–578.
[23] Dan Feldman, Melanie Schmidt, and Christian Sohler. 2020. Turning Big Data Into Tiny Data: Constant-Size Coresets for k-Means, PCA, and Projective Clustering. *SIAM Journal of Computing* 49, 3 (2020), 601–657.
[24] Eldar Fischer, Eric Lehman, Ilan Newman, Sofya Raskhodnikova, Ronitt Rubinfeld, and Alex Samorodnitsky. 2002. Monotonicity testing over general poset domains. In *Proceedings of ACM Symposium on Theory of Computing (STOC)*. 474–483.
[25] Yoav Freund, H. Sebastian Seung, Eli Shamir, and Naftali Tishby. 1997. Selective Sampling Using the Query by Committee Algorithm. *Machine Learning* 28, 2-3 (1997), 133–168.
[26] Chaitanya Gokhale, Sanjib Das, AnHai Doan, Jeffrey F. Naughton, Narasimhan Rampalli, Jude W. Shavlik, and Xiaojin Zhu. 2014. Corleone: hands-off crowdsourcing for entity matching. In *Proceedings of ACM Management of Data (SIGMOD)*. 601–612.
[27] Oded Goldreich, Shafi Goldwasser, Eric Lehman, and Dana Ron. 1998. Testing Monotonicity. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 426–435.
[28] Oded Goldreich, Shafi Goldwasser, Eric Lehman, Dana Ron, and Alex Samorodnitsky. 2000. Testing Monotonicity. *Combinatorica* 20, 3 (2000), 301–337.
[29] Steve Hanneke. 2014. Theory of Disagreement-Based Active Learning. *Foundations and Trends in Machine Learning* 7, 2-3 (2014), 131–309.
[30] Sariel Har-Peled and Akash Kushal. 2007. Smaller Coresets for k-Median and k-Means Clustering. *Discrete & Computational Geometry* 37, 1 (2007), 3–19.
[31] Sariel Har-Peled and Micha Sharir. 2011. Relative (p, ϵ)-Approximations in Geometry. *Discrete & Computational Geometry* 45, 3 (2011), 462–496.

[32] Matti Kääriäinen. 2006. Active Learning in the Non-realizable Case. In *Proceedings of International Conference on Algorithmic Learning Theory (ALT)*. 63–77.
[33] Hanna Köpcke, Andreas Thor, and Erhard Rahm. 2010. Evaluation of entity resolution approaches on real-world match problems. *Proceedings of the VLDB Endowment (PVLDB)* 3, 1 (2010), 484–493.
[34] Jane Lange, Ronitt Rubinfeld, and Arsen Vasilyan. 2022. Properly learning monotone functions via local correction. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 75–86.
[35] Jane Lange and Arsen Vasilyan. 2023. Agnostic proper learning of monotone functions: beyond the black-box correction barrier. In *Proceedings of Annual IEEE Symposium on Foundations of Computer Science (FOCS)*. 1149–1170.
[36] Venkata Vamsikrishna Meduri, Lucian Popa, Prithviraj Sen, and Mohamed Sarwat. 2020. A Comprehensive Benchmark Framework for Active Learning Methods in Entity Matching. In *Proceedings of ACM Management of Data (SIGMOD)*. 1133–1147.
[37] Rajeev Motwani and Prabhakar Raghavan. 1995. *Randomized Algorithms*. Cambridge University Press.
[38] Jeff M. Phillips. 2016. *Coresets and Sketches*. CRC Press, Chapter 49.
[39] Sunita Sarawagi and Anuradha Bhamidipaty. 2002. Interactive deduplication using active learning. In *Proceedings of ACM Knowledge Discovery and Data Mining (SIGKDD)*. 269–278.
[40] Burr Settles. 2010. Active learning Literature Survey. *Technical Report, University of Wisconsin-Madison* (2010).
[41] Quentin F. Stout. 2013. Isotonic Regression via Partitioning. *Algorithmica* 66, 1 (2013), 93–112.
[42] Yufei Tao. 2018. Entity Matching with Active Monotone Classification. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*. 49–62.
[43] Yufei Tao and Yu Wang. 2021. New Algorithms for Monotone Classification. In *Proceedings of ACM Symposium on Principles of Database Systems (PODS)*. 260–272.
[44] Andreas Thor and Erhard Rahm. 2007. MOMA - A Mapping-based Object Matching System. In *Proceedings of Biennial Conference on Innovative Data Systems Research (CIDR)*. 247–258.
[45] Liwei Wang. 2011. Smoothness, Disagreement Coefficient, and the Label Complexity of Agnostic Active Learning. *Journal of Machine Learning Research (JMLR)* 12 (2011), 2269–2292.