Entity Matching with Active Monotone Classification

Yufei Tao*
Chinese University of Hong Kong
Hong Kong
taoyf@cse.cuhk.edu.hk

ABSTRACT

Given two sets of entities X and Y, *entity matching* aims to decide whether x and y represent the same entity for each pair $(x, y) \in X \times Y$. As the last resort, human experts can be called upon to inspect every (x, y), but this is expensive because the correct verdict could not be determined without investigation efforts dedicated specifically to the two entities x and y involved. It is therefore important to design an algorithm that asks humans to look at only some pairs, and renders the verdicts on the other pairs automatically with good accuracy.

At the core of most (if not all) existing approaches is the following classification problem. The input is a set P of points in $\mathbb{R}^d$, each of which carries a binary label: 0 or 1. A *classifier* $\mathcal{F}$ is a function from $\mathbb{R}^d$ to $\{0, 1\}$. The objective is to find a classifier that captures the labels of a large number of points in P.

In this paper, we cast the problem as an instance of active learning where the goal is to learn a *monotone* classifier $\mathcal{F}$, namely, $\mathcal{F}(p) \geq \mathcal{F}(q)$ holds whenever the coordinate of p is at least that of q on all dimensions. In our formulation, the labels of all points in P are hidden at the beginning. An algorithm A can invoke an *oracle*, which discloses the label of a point $p \in P$ chosen by A. The algorithm may do so repetitively, until it has garnered enough information to produce $\mathcal{F}$. The cost of A is the number of times that the oracle is called. The challenge is to strike a good balance between the *cost* and the *accuracy* of the classifier produced.

We describe algorithms with non-trivial guarantees on the cost and accuracy simultaneously. We also prove lower bounds that establish the asymptotic optimality of our solutions for a wide range of parameters.

CCS CONCEPTS

• **Theory of computation → Approximation algorithms analysis**; **Active learning**;

KEYWORDS

Active Learning, Monotone Classification, Entity Matching

*This work was partially supported by a direct grant (Project Number: 4055079) from CUHK and by a Faculty Research Award from Google.

PODS'18, June 10–15, 2018, Houston, TX, USA

ACM ISBN 978-1-4503-4706-8/18/06…$15.00
https://doi.org/10.1145/3196959.3196984

ACM Reference Format:
Yufei Tao. 2018. Entity Matching with Active Monotone Classification. In *PODS'18: 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, June 10–15, 2018, Houston, TX, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3196959.3196984

1 INTRODUCTION

Given two sets of entities X and Y, *entity matching* aims to decide whether x and y represent the same entity for each pair $(x, y) \in X \times Y$; if so, they are said to form a *match*. For example, X (or Y) can be a set of advertisements placed at *Amazon* (or *eBay*, resp.). Each advertisement has attributes like `prod-name`, `prod-description`, `year`, `price`, and so on. The goal is to decide whether advertisements x and y are about the same product, for all $(x, y) \in X \times Y$.

What makes the problem a challenge is the fact that the decision cannot be made through a simple comparison on the attributes of x and y, because even a pair of matching x and y may still disagree on the attribute values. This is obvious for `prod-description` and `price` since x and y can introduce the same product in different ways, and price it differently. In fact, x and y may not agree even on "supposedly standardized" attributes like `prod-name` (e.g., x.`prod-name` = "MS Word" vs. y.`prod-name` = "Microsoft Word Processor"), although it would be reasonable to expect x.`year` = y.`year` (advertisements are required to be correct).

As the bottomline resort, human experts are called upon to inspect each $(x, y) \in X \times Y$. This is expensive because the inspection demands manual efforts (e.g., reading both advertisements in detail). It is therefore important to design an algorithm that asks humans to look at only some pairs, and renders the verdicts on the other pairs automatically, perhaps at the expense of a small number of errors.

Towards the above purpose, a dominant methodology behind existing approaches (e.g., [1, 3, 6, 7, 12, 14, 22, 24, 26]) 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 $T \subseteq X \times Y$, by eliminating the pairs that obviously cannot be matches. This is known as *blocking*, which is carried out based on application-dependent heuristics. This step is optional; if skipped, then $T = X \times Y$. In the Amazon-eBay example, T may involve only those advertisement pairs (x, y) with x.`year` = y.`year`.
(2) For each remaining entity pair $(x, y) \in T$, create a multidimensional point $p_{(x,y)}$ using a number d of similarity functions

$$sim_1, sim_2, ..., sim_d,$$

each evaluated on a certain feature. The i-th coordinate of $p_{(x,y)}$ equals $sim_i(x, y)$: a higher value means that x and

y are more similar on the i-th feature. This creates a d-dimensional point set $P = \{p_{(x,y)} \mid (x,y) \in T\}$.
In our example, from a numerical attribute such as `price`, one may extract a feature that equals $-|x.\texttt{price} - y.\texttt{price}|$ (the negation is just to be consistent with "larger means more similar"). From a text attribute (such as `prod-name` and `prod-description`) one may extract a feature by evaluating the similarity between the corresponding texts of x and y using an appropriate metric (e.g., edit distance for short texts, and cosine similarity for long texts). Multiple feature dimensions may be derived even on the same attribute; e.g., one can extract two features by computing the edit-distance and jaccard-distance of $x.\texttt{prod-name}$ and $y.\texttt{prod-name}$ separately.

Every point $p_{(x,y)} \in P$ carries a *label*, which is 1 if (x,y) is a match, or 0 otherwise. The original entity matching task on X, Y is thus converted to inferring the labels of the points in P. Still, human inspection is the last resort for revealing the label of each $p_{(x,y)}$ with no errors.

1.1 Problem Definitions

In this paper, we study two problems that are at the core of the aforementioned framework of entity matching. Both problems have considerable connections to active learning, as will be clarified later.

Let P be a set of n points in $\mathbb{R}^d$. Each point $p \in P$ carries a label—denote as $label(p)$—that equals either 0 or 1. The point labels need not follow any geometric patterns, namely, $label(p)$ can be 0 or 1 regardless of the labels of the other points.

All the labels are hidden at the beginning. There is an *oracle* which an algorithm can call to disclose the label of a point $p \in P$ selected by the algorithm. When this happens, we say that p is *probed*. The algorithm's *cost* is defined as the number of points probed.

A *classifier* is a function $\mathcal{F} : \mathbb{R}^d \to \{0,1\}$. Its *error* on P is the number of points mis-labeled, namely:

$$error(\mathcal{F}, P) = |\{p \in P \mid \mathcal{F}(p) \neq label(p)\}|.$$

$\mathcal{F}$ is *monotone* if $\mathcal{F}(p) \geq \mathcal{F}(q)$ for any points $p, q \in P$ satisfying $p[i] \geq q[i]$ on all $i \in [1,d]$, where $p[i]$ is the i-th coordinate of p. The set $\mathbb{C}_{mono}$ of monotone classifiers is infinite.

The first problem we consider is:

Problem 1 (*Active Monotone Classification without Exceptions*): Find a monotone classifier $\mathcal{F}$ of small $error(\mathcal{F}, P)$ by paying a low probing cost.

One can trivially minimize the error by probing all points of P, and then taking all the time needed to fit the labels of P with the best $\mathcal{F}$ in $\mathbb{C}_{mono}$ (we do not explicitly constrain CPU computation). Note that the minimum error *need not be zero* because the labels of P may not—actually most likely do not—fully obey monotonicity. The above approach ensures the smallest $error(\mathcal{F}, P)$, but has a terrible probing cost of n. An interesting question is whether one can achieve the smallest error (perhaps asymptotically) with fewer probes.

The second problem is similar, but differs in the error targeted:

Problem 2 (*Active Monotone Classification with Exceptions*): Probe a small set Z of points in P to find a monotone classifier $\mathcal{F}$ of small $error(\mathcal{F}, P \setminus Z)$.

Note that the set Z of points is exempted from error calculation. That is, we do not care whether these points are mis-labeled by $\mathcal{F}$ (their labels have been revealed anyway). It is once again possible to minimize error by probing the entire P, making $Z = P$ and $error(\mathcal{F}, P \setminus Z) = 0$. An interesting question is whether one can choose Z strategically to strike a better tradeoff between $error(\mathcal{F}, P \setminus Z)$ and $|Z|$.

The above definitions extend in a natural way to a randomized algorithm A_{ran}. If $\mathcal{F}$ is the monotone classifier A_{ran} returns and Z is the set of points it probes, both $\mathcal{F}$ and Z are random variables. For Problem 1, the *expected error* of A_{ran} is defined as $\mathbf{E}[error(\mathcal{F}, P)]$, and its *expected probing cost* as $\mathbf{E}[|Z|]$. Likewise, for Problem 2, the *expected error* of A_{ran} is defined as $\mathbf{E}[error(\mathcal{F}, P \setminus Z)]$, while its *expected probing cost* still as $\mathbf{E}[|Z|]$. In all cases, expectation is over the random choices made by the algorithm.

Remarks. The input P to both problems corresponds to the set of points obtained from T in the entity matching framework explained earlier. Problems 1 and 2 are designed for two scenarios that arise frequently in practice:

- Scenario 1: the entity sets X and Y are "training sets" that represent the distributions D_X and D_Y of entities to be encountered from two sources, respectively. The purpose of finding a classifier $\mathcal{F}$ is to apply it on new $(\tilde{x}, \tilde{y}) \notin X \times Y$ to be received online where $\tilde{x}$ (or $\tilde{y}$) follows D_X (or D_Y, resp.). That $\mathcal{F}$ is accurate on P (a.k.a. T) implies that $\mathcal{F}$ should also work statistically well on $(\tilde{x}, \tilde{y})$. This is the application backdrop of Problem 1.
- Scenario 2: unlike the previous scenario, here X and Y are already the "ground sets". In other words, there are no future pairs (such as $(\tilde{x}, \tilde{y})$ in Scenario 1) to be cared for; and it suffices to match only the elements of X and Y. Thus, the "overall accuracy" of $\mathcal{F}$ on *all* the points in P is of no concern to us: if a point already has its label revealed, it need not be guessed by $\mathcal{F}$, and thus, should be excluded from accuracy calculation. This is the application backdrop behind Problem 2.

The rationale behind monotonicity is that, if x and y form a match according to the features picked, then any pair (x', y') more similar than (x, y) on *every* feature should also be regarded as a match, unless explicit manual inspection (i.e., a probe on $p_{(x',y')}$) reveals otherwise. Indeed, the classifiers output by all existing methods are inherently monotone. In fact, any classifier that defies monotonicity is *awkward* because it indicates that at least some of the features have been selected inappropriately.

It should be emphasized that, finding a monotone classifier is different from assuming that the point labels fully obey monotonicity. That assumption is rarely true, which is why even the best monotone classifier can incur an error above 0.

1.2 Related Work

Our discussion of previous research first aims to give a self-contained introduction to the key findings of active learning. This

would be useful because active learning does not seem to have been introduced in the database community to the same level of details. In fact, some of those findings constitute the state of the art on Problems 1 and 2. Then, we will also review methods from the database area on entity matching.

Active Learning. Classification is a fundamental topic in machine learning. Let U be a (finite or infinite) set of points in $\mathbb{R}^d$. The input is an infinite stream of pairs $(p, label(p))$ where p is a point from U, and $label(p)$ its label which is 0 or 1. Each pair is sampled independently from an unknown distribution D on $U \times \{0, 1\}$. A *classifier* is a function $\mathcal{F} : U \to \{0, 1\}$; its *error probability* is calculated as

$$\mathbf{Pr}\{p \sim D \mid \mathcal{F}(p) \neq label(p)\},$$

namely, the probability of wrongly predicting the label of a point p drawn from D. Let $\mathbb{C}$ be a *candidate class* of classifiers, and ν be the smallest error probability of all the classifiers in $\mathbb{C}$. The learning objective is stated in the *probabilistically approximately correct* (PAC) style:

> With probability at least $1 - \delta$, find a classifier $\mathcal{F}$ from $\mathbb{C}$ with error probability at most $\nu + \epsilon$ for some small $\epsilon > 0$.

In the traditional *passive* setup, $label(p)$ is directly disclosed in every pair $(p, label(p))$, where the efficiency goal is to minimize the *sample cost*, which equals the number of stream pairs that an algorithm needs to see before ensuring the aforementioned PAC guarantee. In practice, (as mentioned before) deciding the label of a point can be so expensive that its cost far dominates the cost of learning. This motivated active learning, where point labels are all hidden originally. Given an incoming point p, an algorithm can choose whether to *probe* p (pay a unit cost to an oracle for revealing $label(p)$). The primary efficiency goal is to perform the least number of probes (efforts should still be made to avoid a high sample cost, although this now becomes a secondary goal).

Active learning has been extensively studied (see excellent surveys [16, 25]). The main challenge is to identify the *intrinsic parameters* that determine the *label complexity*, i.e., the number of probes mandatory to ensure the PAC guarantee. Considerable progress has been made in various scenarios [4, 8, 15]. Our subsequent discussion will concentrate on *agnostic* active learning, where (i) $\nu > 0$, meaning that even the best classifier in $\mathbb{C}$ cannot perfectly separate points of the two labels, that is, D has "noise", and (ii) no assumptions are made on the noise. This is the branch of active learning most relevant to our work.

The state-of-the-art understanding on agnostic learning is based on two intrinsic parameters:

- VC dimension λ of $\mathbb{C}$ on U: Let $S \subseteq U$ be a set of points $\{p_1, p_2, ..., p_{|S|}\}$. S is *shattered* by $\mathbb{C}$ if, for any $(l_1, l_2, ..., l_{|S|}) \in \{0, 1\}^{|S|}$, $\mathbb{C}$ has a classifier $\mathcal{F}$ satisfying $\mathcal{F}(p_i) = l_i$ for all $i \in [1, |S|]$. The VC dimension λ is the size of the largest S that can be shattered by $\mathbb{C}$.
- Disagreement coefficient θ: This parameter is directly related to how many labels are needed to distinguish an optimal classifier $\mathcal{F}^*$ in $\mathbb{C}$ from other classifiers similar to $\mathcal{F}^*$ under the distribution D. We defer its precise definition (which is slightly involved) to Appendix A. Until then, it suffices to understand that a higher θ indicates the necessity of more labels.

The dominant solution to agnostic active learning is an algorithm named A^2. Its initial ideas were developed by Balcan et al. [2], and have been substantially improved/extended subsequently [4, 9, 16]. As shown in [16], the algorithm achieves the PAC guarantee with a probing cost of

$$\tilde{O}\left(\theta \cdot \lambda \cdot \frac{\nu^2}{\epsilon^2}\right) \tag{1}$$

where $\tilde{O}(.)$ hides factors polylogarithmic to $\theta, 1/\epsilon$, and $1/\delta$. On the lower bound side, extending an earlier result of Kaariainen [18], Beygelzimer et al. [4] proved that the probing cost needs to be:

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

There is a gap of θ between the upper and lower bounds. When this parameter is $O(1)$, the two bounds match up to polylog factors. Indeed, most success stories in the literature are based on candidate classes $\mathbb{C}$ with small θ (e.g., see [8, 10, 13, 27]). Unfortunately, this is not true for the class $\mathbb{C}_{mono}$ of monotone classifiers. We will prove that its disagreement coefficient θ can be very large. Not coincidentally, $\mathbb{C}_{mono}$ has very large VC dimension λ, too. The consequences are two-fold:

- The θ gap between (1) and (2) becomes significant. This means that agnostic active learning has *not* been well understood on monotone classifiers.
- With both θ and λ being large, the A^2 algorithm incurs expensive probing costs on $\mathbb{C}_{mono}$, and may no longer be attractive.

Problem 1 can be reduced to agnostic active learning. For this purpose, we set U (in active learning) to P (in Problem 1), and generate an input stream (for active learning) by repeatedly sampling P. This makes A^2 a viable solution to Problem 1. We will discuss its performance guarantees in relation to our results in the next subsection.

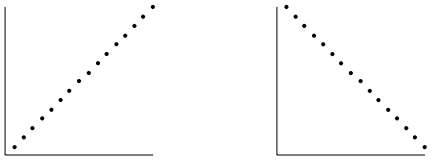
Entity Matching. There is a rich literature on entity matching; see [1, 3, 5–7, 12, 14, 19–22, 24, 26] and the references therein. Most of these works focused on designing heuristics that perform well in practice, instead of establishing theoretical bounds. The papers [1, 3] are exceptions. In [1], Arasu et al. observed that entity matching can be cast as active learning. They presented algorithms to solve (with guarantees) some subproblems that arose in their framework. Unfortunately, their overall solutions do not have attractive bounds for Problem 1 or 2. In [3], Bellare et al. showed that if one can solve Problem 1 (which minimizes the so-called *0-1 error*), the algorithm can be utilized to attain small errors of other types (e.g., those based on recalls and precisions), under certain assumptions.

1.3 Our Results

An Intrinsic Parameter. Recall that Problems 1 and 2 are defined on a set P of n points in $\mathbb{R}^d$. Given two points $p, q \in P$, we say that p *dominates* q if $p[i] \geq q[i]$ holds on all $i \in [1, d]$. Notice that a point dominates itself by this definition. The dominance relation

$$R = \{(p, q) \in P \times P \mid p \text{ dominates } q\}$$

is a poset (partially ordered set).



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

Figure 1: Illustration of dominance width

It turns out that an intrinsic parameter characterizing the hardness of both problems is the *width* w of R. Formally, w is the size of the largest $S \subseteq P$ such that no two different points in S dominate each other; we will sometime refer to it as the *dominance width* of P. Any one-dimensional P has $w = 1$. When $d \geq 2$, w can be anywhere between 1 and n; see Figure 1 for two extreme examples.

Problem 1. Denote by $\mathcal{F}^*$ an optimal monotone classifier on P, namely, this is a classifier $\mathcal{F}$ in $\mathbb{C}_{mono}$ with the smallest $error(\mathcal{F}, P)$. We will use k to denote $error(\mathcal{F}^*, P)$, and refer to k as the *minimum monotone error* of P. Remember that k can be greater than 0 because the labels of the points in P may not follow monotonicity perfectly.

Our first main result is obtained through competitive analysis:

Theorem 1. *For Problem 1:*

- *there is a randomized algorithm that has expected error at most* $2k$*, and probes* $O(w(1 + \log \frac{n}{w}))$ *points in expectation;*
- *there exists a constant* c *such that, when* $w \geq 2$ *and* $k \leq cn/w$*, any algorithm with expected error* $O(k)$ *must have an expected probing cost of* $\Omega(w \log \frac{n}{(k+1)w})$.

Several observations can be made. First, when $k = 0$—the *noise-free* scenario where the label-1 points of P can be perfectly separated from the label-0 ones by a monotone classifier—our algorithm (the first bullet) always returns such a classifier. Second, when $w = \Omega(n)$, our lower bound (the second bullet) evaluates to $\Omega(n)$, meaning that the naive solution of probing all points in P is already optimal. Third, active learning improves the naive solution as long as $w = o(n)$: observe that, for $w = o(n)$, the upper bound in the first bullet is $o(n)$. Fourth, our upper and lower bounds nearly match each other for $k = O(n/w)$. Remember that no algorithms can have an expected error less than k. Therefore, under $k = O(n/w)$, our solution is asymptotically optimal in both expected error and expected probing cost.

As explained in Section 1.2, one can apply the A^2 algorithm of agnostic active learning to solve Problem 1 by repeatedly sampling from P. To see its performance, let us fit in the appropriate values for ν, ϵ, λ, and θ, as are defined in Section 1.2. First, $\nu = k/n$, according to the definition of $error(\mathcal{F}^*, P)$. Second, to match our expected error $2k$, ϵ should be no more than $\nu = k/n$; setting ϵ to this value makes $\nu^2/\epsilon^2 = 1$. On the other hand, when $k = O(n/w)$, both θ and λ can reach w even in 2D space (see Appendix A). By (1), A^2 has expected probing cost $\tilde{O}(w^2)$, worse than our bound by a factor of $\tilde{O}(w)$.

Note that (2) does *not* give a lower bound on Problem 1. Recall that (2) applies to agnostic active learning which is just one possible way to approach Problem 1.

Problem 2. If an algorithm returns a monotone classifier $\mathcal{F}$ by probing a set Z of points, it always holds that $error(\mathcal{F}, P \setminus Z) \leq error(\mathcal{F}, P)$. Hence, if an algorithm ensures an error at most, say x, on Problem 1, the same algorithm must also achieve an error at most x on Problem 2. Theorem 1 implies:

Corollary 1. *For Problem 2, there is a randomized algorithm that has expected error at most* $2k$*, and expected probing cost* $O(w(1 + \log \frac{n}{w}))$.

What is intriguing is the opposite: can we substantially reduce the error of Problem 2 without significantly increasing the probing cost? This, subtly, is a question on the usefulness of Z. The intended purpose of Z is to push $error(\mathcal{F}, P \setminus Z)$ *below* k (in Problem 1, the problem definition determines that the best error is k, no matter what). Unfortunately, we prove:

Theorem 2. *Fix any integers* k *and* n *such that* $k \geq 1$*, and* n/k *is an integer at least 2. There is a set* S *of 1D (*$d = 1$*) inputs to Problem 1 with the same* n *and* k *such that, any algorithm guaranteeing an expected error at most* $k/2$ *on every input in* S *must entail an expected probing cost of* $\Omega(n/k)$ *on at least one input in* S.

Corollary 1 and Theorem 2 together point out a somewhat surprising fact. If we are satisfied with an expected error of $2k$, the expected probing cost is no more than $O(w(1 + \log \frac{n}{w}))$ "universally" for *all* values of k. Even better, in the context of Theorem 2 where $d = 1$, w equals 1, making $O(w(1 + \log \frac{n}{w})) = O(\log n)$. However, if we demand an expected error of $k/2$, the expected probing cost surges to $\Omega(n/k)$, which is worse than $O(\log n)$ for any $k = o(n/\log n)$.

Theorem 2 also implies that, for $k \leq n/(w \log_2 \frac{n}{w})$, the expected error must be at least $\Omega(k)$ if the expected probing cost has to be $O(w \log(n/w))$. Thus, on those values of k, our algorithm in Corollary 1 is already asymptotically optimal. Phrased differently, subject to $O(w \log \frac{n}{w})$ expected probing cost, the hardness of the problem comes from guessing the labels of points that have not been probed, such that excluding Z from error calculation makes little difference.

Finally, it is worth mentioning that all our algorithms can be implemented in $O(n \operatorname{polylog} n)$ CPU time under any fixed dimensionality d.

2 WARM UP: NOISE-FREE INPUTS

In this section, we study Problem 1 by assuming a *noise-free* input P, i.e., $k = 0$. In other words, there exists a *perfect* classifier $\mathcal{F}^*$ in $\mathbb{C}_{mono}$ with $error(\mathcal{F}^*, P) = 0$. Our objective is to discover such a perfect monotone classifier with the least number of probes. Note that a noise-free input P definitely obeys monotonicity: $label(p) \geq label(q)$ if point $p \in P$ dominates another point $q \in P$.

Starting with the noise-free version has two main benefits. First, it permits us to bring out the intrinsic parameter w (dominance width of P) naturally (Section 2.1). Second, it simplifies the explanation of some key ideas (Sections 2.2 and 2.3). Grasping these ideas at an early stage will be helpful in later sections.

2.1 The First Algorithm

We will first explain a simple method that finds a perfect monotone classifier with $O(w(1 + \log \frac{n}{w}))$ probes. Let C be a subset of P. We call C:

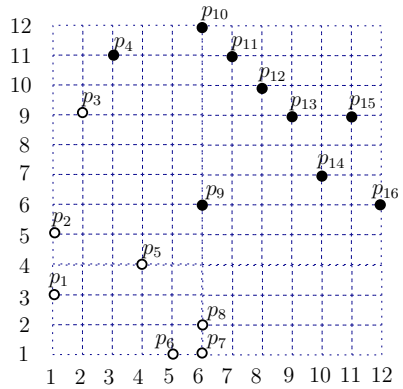


Figure 2: A noise-free 2D input (black and white points have label 1 and 0, respectively)

- A *chain*, if it is possible to linearize the points of C into a sequence $p_1, p_2, ..., p_{|C|}$ such that p_{i+1} dominates p_i for every $i \in [1, |C| - 1]$. We will refer to the sequence as the *ascending order* of C.
- An *anti-chain*, if none of the points in C dominate each other.

Figure 2 shows a noise-free input where the black (or white) points carry label 1 (or 0, resp.). Subset $\{p_6, p_8, p_{10}\}$, for instance, is a chain, whereas $\{p_4, p_{12}, p_{13}, p_{14}\}$ is an anti-chain.

When P is noise-free, the points $p_1, p_2, ..., p_{|C|}$ of a chain C in ascending order exhibit a *step-wise* pattern: the label-0 points must precede all the label-1 ones. This allows us to correctly decide the labels of all the points of C in $O(1 + \log |C|)$ probes with binary search. Specifically, probe $p_{\lceil |C|/2 \rceil}$. If its label is 1, the points after it must also carry this label, so that we can focus recursively on the first half of C. The case where $p_{\lceil |C|/2 \rceil}$ has label 0 is symmetric.

This observation is what directed our attention to *chain decompositions*, each being a collection of disjoint chains $C_1, C_2, ..., C_t$ (for some $t \geq 1$) whose union equals P. Once a chain decomposition is available, the labels of all points can be correctly decided in $O(t(1 + \log \frac{n}{t}))$ probes by performing binary search on every chain separately[1]. How to minimize the number t of chains is a fundamental result in order theory:

> **Dilworth's Theorem:** Consider any poset; let w be the largest size of all anti-chains. Then, (i) there is a chain decomposition that contains w chains, and (ii) no chain decompositions can have less than w chains.

For instance, the input set P of Figure 2 can be divided 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 $C_6 = \{p_6, p_7, p_8, p_{14}, p_{15}\}$. This is a smallest chain decomposition due to the anti-chain $\{p_{10}, p_{11}, p_{12}, p_{16}, p_{13}, p_{14}\}$. Hence, the dominance width w of P is 6.

We have obtained an algorithm that performs $O(w \log \frac{n}{w})$ probes, but the algorithm has two defects:

- First, the idea of binary search no longer works on a "noisy" P, because in that scenario the 0- and 1-labels can arbitrarily interleave with each other on a chain.
- Second, finding a smallest chain decomposition for P is computationally expensive. The fastest algorithm we are aware of requires $O(wn^2)$ CPU time [17] for $d \geq 3$. While CPU time is not explicitly constrained in Problems 1 and 2, $O(wn^2)$ CPU calculation is unlikely to be appealing in practice.

Next, we describe another algorithm that remedies the above defects with drastically different ideas.

[1] $\sum_{i=1}^{t} O(1 + \log |C_i|)$ is maximized when $|C_1| = |C_2| = ... = |C_t| = n/t$.

2.2 A Randomized Algorithm

Our second algorithm—called *random probe with elimination* (RPE)—can be described in 6 lines:

```
Algorithm RPE(P)
1.  while P ≠ ∅
2.      pick a point p from P uniformly at random
3.      probe p
4.      if label(p) = 1 then
5.          discard from P the points dominating p
        else
6.          discard from P the points that p dominates
```

If Z is the set of points probed, we produce a monotone classifier $\mathcal{F}$ as:

$$\mathcal{F}(p) = \begin{cases} 1 & \text{if } p \text{ dominates a label-1 point in } Z \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

Lemma 1. *On a noise-free P, the above $\mathcal{F}$ produced from RPE is monotone, and satisfies error$(\mathcal{F}, P) = 0$.*

Proof. Let p be an arbitrary point $p \in P$. Consider first $label(p) = 1$. If $p \in Z$, then clearly $\mathcal{F}(p) = 1$. Otherwise, p must have been discarded at Step 5 when RPE probed another label-1 p'; but then this means $p' \in Z$, ensuring $\mathcal{F}(p) = 1$. On the other hand, if $label(p) = 0$, then $\mathcal{F}(p)$ must be 0 because p cannot dominate any label-1 points in Z. □

Next, we will prove the algorithm's probing cost:

Lemma 2. *When P noise-free, RPE probes $O(w(1 + \log \frac{n}{w}))$ points in expectation.*

An Attrition-and-Elimination Game. Let us take a detour to discuss a relevant problem first. Consider the following game between two players Alice and Bob. At the beginning, Alice is given the set $S = \{1, 2, ..., s\}$ for some integer $s \geq 1$. The game goes in *rounds*. In each round:

- Bob performs "attrition" first, by either doing nothing or arbitrarily deleting some elements from S;
- Alice then performs "elimination" by picking a number $p \in S$ uniformly at random, and deleting from S all the numbers larger than or equal to p.

The game ends when S 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 variable's expectation.

It is fairly intuitive that Bob should do nothing at all in every round. To prove this, denote by function $f(s)$ the smallest real number such that, Bob has a strategy to make the expected number of rounds equal $f(s)$. Consider the first attrition of Bob. Clearly, what matters is the number x of elements that Bob decides to remove (by symmetry, what those elements are does not matter). If $x < s$, the game continues for Alice to work on a set S of size $s - x$.

After her elimination, S has i elements—for each $i \in [0, s-x-1]$—left with probability $1/(s-x)$. Therefore:

$$f(s) \leq 1 + \max_{x=0}^{s-1} \left\{ \frac{1}{s-x} \sum_{i=0}^{s-x-1} f(i) \right\}.$$

As the base case, $f(0) = 0$. Solving the recurrence gives $f(s) = O(1 + \log s)$ for $s \geq 1$.

Proof of Lemma 2. Let $\{C_1, C_2, ..., C_w\}$ be an arbitrary smallest chain decomposition (which is *not* known to RPE). We will prove that in expectation RPE probes $O(1 + \log |C_i|)$ points in C_i for all $i \in [1, w]$. It will then follow that the total expected number of probes is $O(\sum_{i=1}^{w}(1 + \log |C_i|))$, which peaks at $O(w(1 + \log \frac{n}{w}))$ when all the chains have the same size n/w.

Without loss of generality, let us focus on C_1. Break C_1 into (i) the set C_1^{true} of points with label 1, and (ii) the set C_1^{false} of points with label 0. Due to symmetry, it suffices to prove that RPE probes $O(1 + \log |C_1^{true}|)$ points from C_1^{true} in expectation.

Set $s = |C_1^{true}|$. List the points of C_1^{true} in ascending order $p_1, p_2, ..., p_s$. The operations that RPE performs on this chain can be captured as an attrition-and-elimination game on an initial input $S = \{p_1, p_2, ..., p_s\}$:

- Bob formulates his strategy by observing the execution of RPE. Suppose that the algorithm probes a point $p \notin C_1^{true}$, and shrinks P at Step 5 or 6. Bob deletes from S all those points of C_1^{true} that are discarded in the shrinking.
- When RPE probes a point $p \in C_1^{true}$, Bob finishes his attrition in this round, and passes S to Alice. *Conditioned* on $p \in C_1^{true}$, p was chosen uniformly at random from the *current* S, i.e., the set of points from C_1^{true} that are still in P. Hence, p can be regarded as the choice of Alice. When RPE shrinks P at Step 5, Alice discards p, as well as all the points behind p, from S. This finishes a round of the game. The control is passed back to Bob to start the next round.

By our earlier analysis on the attrition-and-elimination game, RPE probes $O(1 + \log s)$ points from C_1^{true} in expectation, thus completing the proof of Lemma 2.

Remark on CPU Time. RPE does not produce any chains at all (let alone a chain decomposition), and neither is it aware of the width w of the dominance relation R. This allows the algorithm to be implemented in $O(n \operatorname{polylog} n)$ time for fixed dimensionality d. By maintaining P in a binary search tree, a random point $p \in P$ can be drawn at Step 2 in $O(\log n)$ time, such that in total we spend $O(n \log n)$ time on this step. By maintaining a range tree [11] on P, Steps 5 and 6 can be implemented in $O(x \log^d n)$ time, if x points are eliminated from P. Each point contributes to the x-term exactly once (it can be deleted only once). Hence, the total CPU time spent on these two steps is bounded by $O(n \log^d n)$.

2.3 A Lower Bound

We finish the section by proving the optimality of RPE in the noise-free scenario. Given a rectangle in 2D space, we define its *main diagonal* as the segment connecting the bottom-left and top-right corners, and call the other diagonal as the *anti-diagonal* (where the term "anti" reflects anti-chain).

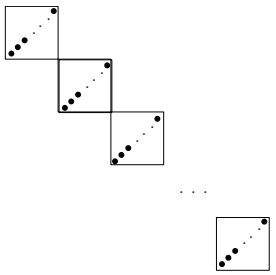


Figure 3: A hard input for Problem 1: w boxes, each with n/w points

Our lower bound is based on hard inputs constructed in 2D space as follows. Fix any integers w and n such that $1 \leq w \leq n$ and n/w is an integer. Divide the data space into a $w \times w$ grid. Place points only in the w cells along the anti-diagonal of the data space (see Figure 3), with n/w points per cell. In each cell, the n/w points are positioned evenly on the cell's main diagonal, but make sure that no points are at the cell corners. The set P of n points thus obtained clearly has width w. An adversary sets the labels of different cells independently. For each cell, there are $1 + \frac{n}{w}$ different ways to do so: for each $i \in [0, \frac{n}{w}]$, assign label 0 to the lowest i points, and 1 to the rest. This gives a collection $\mathcal{H}(n, w)$ of $(1 + \frac{n}{w})^w$ labeled inputs.

Let A_{det} be a deterministic algorithm operating on $\mathcal{H}(n, w)$. We can thus model A_{det} as a binary decision tree T. At an internal node of T, A_{det} probes a point p of P, and branches left (or right) if $label(p) = 0$ (or 1, resp.). At a leaf of T, A_{det} must output a monotone classifier.

Suppose that A_{det} guarantees returning a monotone classifier with error 0. A_{det} must distinguish all the inputs in $\mathcal{H}(n, w)$, by outputting a distinct classifier at each leaf of its decision tree. Define the *average cost* of A_{det} as the average of its probing costs on all the inputs in $\mathcal{H}(n, w)$. This is equivalent to the "average depth" of the leaves in T. A binary tree with $(1 + \frac{n}{w})^w$ leaves must have an average depth of $\Omega(w \log(1 + \frac{n}{w}))$. Hence, A_{det} must have an average cost $\Omega(w(1 + \log \frac{n}{w}))$.

We model a randomized algorithm as a procedure that consults an infinite sequence of random bits, and behaves as a deterministic algorithm when the bit sequence is fixed. Alternatively, one can regard a randomized algorithm as a function that maps a random-bit sequence to a binary decision tree. Combining our earlier argument with Yao's minimax theorem [23], any randomized algorithm which outputs a perfect monotone classifier must entail $\Omega(w(1 + \log \frac{n}{w}))$ expected cost on at least one input of $\mathcal{H}(n, w)$.

3 THE NOISY CASE—PROBLEM 1

This section attacks Problem 1 in the general scenario where P is *noisy*, namely, even an optimal classifier $\mathcal{F}^*$ in $\mathbb{C}_{mono}$ has error $k = error(\mathcal{F}^*, P) > 0$. We will establish Theorem 1 by proving its first bullet in Sections 3.1-3.3, and its second bullet in Section 3.4.

3.1 Algorithm

Our solution is simply to run RPE directly on (noisy) P *anyway*, and still return the classifier $\mathcal{F}$ in (3).

To illustrate, consider the input in Figure 4. Here, $k = 3$: an optimal classifier captures the labels of all points but p_1, p_{11}, and

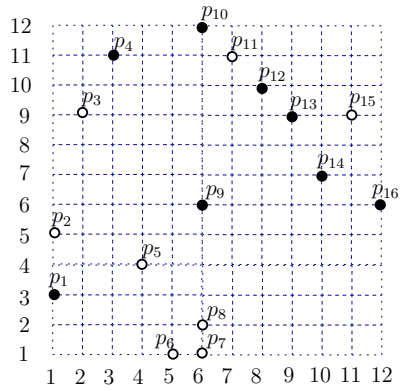


Figure 4: A noisy input where $k = 3$; black and white points have label 1 and 0, respectively.

p_{15}. Assume that RPE happens to probe (at Step 2) p_1 first, after which it eliminates the entire P except p_6, p_7, and p_8. Suppose also that the algorithm then chooses to probe p_8, which removes all the remaining points in P. With $Z = \{p_1, p_8\}$, the monotone classifier of (3) has an error 5 because it incorrectly maps p_2, p_3, p_5, p_{11}, and p_{15} to 1.

Let us give two basic properties of running RPE on a noisy input. First, its output is always legal:

Proposition 1. *RPE returns a monotone classifier $\mathcal{F}$ even when P is noisy.*

Proof. See Appendix B. □

The second property, together with (3), indicates symmetry between the points mapped to 0 and 1 by $\mathcal{F}$.

Proposition 2. *For any $p \in P$, $\mathcal{F}(p) = 0$ if and only if p is dominated by a label-0 point in Z.*

Proof. See Appendix C. □

3.2 Bounding the Error

We now proceed to analyze the error of RPE. Fix an arbitrary optimal monotone classifier $\mathcal{F}^*$, i.e., $k = error(F^*, P)$. Call a point $p \in P$ *good* if $\mathcal{F}^*(p) = label(p)$; otherwise, p is a *noise* point. Divide the set of good points into:

$$G_1^* = \{p \in P \mid label(p) = 1 \text{ and } p \text{ is good}\}, \text{ and}$$
$$G_0^* = \{p \in P \mid label(p) = 0 \text{ and } p \text{ is good}\}.$$

Let $\mathcal{F}$ be the classifier output by RPE. Since P is the union of G_1^*, G_0^*, and the set of k noise points, we know:

$$error(\mathcal{F}, P) \leq error(\mathcal{F}, G_1^*) + error(\mathcal{F}, G_0^*) + k. \quad (4)$$

Let k_0 be the number of label-0 noise points, that is, points p satisfying $label(p) = 0$ but $\mathcal{F}^*(p) = 1$. The rest of Section 3.2 serves as a proof of:

Lemma 3. $\mathbf{E}[error(\mathcal{F}, G_1^*)]$ *is at most* k_0.

Due to the symmetry manifested in Proposition 2, the above lemma implies that $\mathbf{E}[error(\mathcal{F}, G_0^*)]$ is at most the number k_1 of label-1 noise points. (4) then gives $\mathbf{E}[error(\mathcal{F}, P)] \leq k_0 + k_1 + k = 2k$, as claimed in the first bullet of Theorem 1.

3.2.1 RPE by Permutation. To analyze $error(\mathcal{F}, G_1^*)$, it will be convenient to consider an alternative implementation of RPE:

Algorithm RPE-perm(P)
1. randomly permute the points of P
   /* if a point $p \in P$ is the i-th ($i \in [1, n]$) in the permutation, define its *rank* $r(p)$ to be i */
2. **while** $P \neq \emptyset$
3. pick the point $p \in P$ with the smallest rank
4. probe p
5. **if** $label(p) = 1$ **then**
6. discard from P the points dominating p
   else
7. discard from P the points that p dominates

Compared to RPE, RPE-perm differs only in how randomization is injected: this is now done by randomly permuting P. The two implementations have the same expected error and expected probing cost, as proved in Appendix D.

3.2.2 Influence of Noise Points. Let us first gain some intuition on why $error(\mathcal{F}, G_1^*)$ is small in expectation. Consider the example in Figure 4, where G_1^* consists of all the black points except p_1. The bad news is that, if noise point p_{15} is probed first, $\mathcal{F}$ will map p_9, p_{13}, p_{14} to 0 incorrectly (see Proposition 2), causing an increase of 3 to $error(\mathcal{F}, G_1^*)$. The good news is that, if p_{15} is probed after *any* of the good points p_9, p_{13}, p_{14}, then p_{15} will be discarded and can do no harms. Under a random permutation, p_{15} has only 1/4 probability to rank before all of p_9, p_{13}, p_{14}, which seems to suggest that p_{15} could trigger an increase of only 3/4 to $error(\mathcal{F}, G_1^*)$ in expectation.

Unfortunately, the analysis is not as simple as this, due to the presence of noise point p_{11}, which complicates the conditions for p_{15} to be probed. For example, observe that, if p_{11} did not exist, p_{15} can never be probed when p_9 ranks before p_{15}. This is no longer true with the presence of p_{11}. To see this, imagine that p_{11} ranks before p_9, which in turn ranks before p_{15}. The probing of p_{11} evicts p_9 from P. On the other hand, p_{15} remains in P, and hence, gets a chance to be probed later.

The above issue arises because p_9 is dominated—and thereby is "influenced"—by both noise points p_{11} and p_{15}. *Separating* and *quantifying* the influence of each noise point turns out to be the most crucial idea behind our analysis. Let N_0 be the set of label-0 noise points, i.e., $k_0 = |N_0|$. Next, we will describe a way to calculate the "exclusive influence" $I(q)$ of each point $q \in N_0$. In particular, we will do so incrementally by observing how RPE-perm executes.

At the beginning, initialize $I(q) = 0$ for every $q \in N_0$. Whenever RPE-perm is *about* to probe a point $q \in N_0$, capture the set $P(q)$ of points that are still in P at this moment. Then, finalize $I(q)$ as:

$$I(q) = \text{the number of points in } P(q) \cap G_1^* \text{ that are dominated by } q.$$

At the end of RPE-perm, if a point $q \in N_0$ is never probed, define $P(q) = \emptyset$ and finalize its $I(q)$ to be 0.

The next lemma explains why the set $\{I(q) \mid q \in N_0\}$ computed separates and quantifies the influence of the noise points in N_0.

Lemma 4. $\sum_{q \in N_0} I(q) = error(\mathcal{F}, G_1^*)$.

Proof. Consider an arbitrary $q \in N_0$ that was probed by RPE-perm. Let p be any point in $P(q) \cap G_1^*$ that is dominated by q. By Proposition 2, $\mathcal{F}(p) = 0$ because of q. Thus, p contributes 1 to $error(\mathcal{F}, G_1^*)$. Hence, $\sum_{q \in N_0} I(q) \leq error(\mathcal{F}, G_1^*)$.

Conversely, let p be a point contributing 1 to $error(\mathcal{F}, G_1^*)$, that is, $label(p) = 1$ but $\mathcal{F}(p) = 0$. Let S by the set of label-0 points in Z that dominate p. By Proposition 2, $|S| \geq 1$. Define q to be point in S that was probed the earliest. Because p is a good point with label 1 dominated by q, q must be a noise point, i.e., $q \in N_0$. Next, we argue that p must be in $P(q)$, meaning that p contributes 1 to $I(q)$. Therefore, $error(\mathcal{F}, G_1^*) \leq \sum_{q \in N_0} I(q)$.

On the contrary, suppose that $p \notin P(q)$. Thus, p had already disappeared when RPE-perm was about to probe q. By definition of q, this implies that RPE-perm had probed a label-1 point dominated by p; but doing so should have got q discarded, giving a contradiction. □

3.2.3 Proof of Lemma 3. Let us denote the points of N_0 as $q_1, q_2, ..., q_{k_0}$ (in an arbitrary order), and set random variable

$$X = \sum_{i=1}^{k_0} I(q_i).$$

We will show $\mathbf{E}[X] \leq |N_0| = k_0$, which proves Lemma 3 by way of Lemma 4. Given a subset S of P and any point $q \in P$, define:

$$D_S(q) = \{p \in S \mid q \text{ dominates } p\}.$$

Our proof of $\mathbf{E}[X] \leq k_0$ is inductive on k_0.

Base Case $k_0 = 1$. That is, $N_0 = \{q_1\}$.

Lemma 5. *$I(q_1) > 0$ only if q_1 has a smaller rank than all the points in $D_{G_1^*}(q_1)$.*

Proof. Suppose that $D_{G_1^*}(q_1)$ has a point p that ranks before q_1 in the permutation. We argue that RPE-perm will not probe q_1.

Suppose that RPE-perm probes q_1. Consider the moment right before the probing happens. Point p must have disappeared from P (otherwise, RPE-perm cannot have chosen to probe q since the rank of p is smaller). Could it have been discarded due to the probing of a label-0 point $p' \neq q_1$? No, because otherwise, $p \in G_1^*$ asserts that p' must also be a label-0 noise point, contradicting $k_0 = 1$. Thus, p must have been discarded due to the probing of a label-1 point that p dominates. But this should have evicted q_1 as well, also giving a contradiction. □

Hence, $I(q_1) > 0$ with a probability at most $1/(1 + |D_{G_1^*}(q_1)|)$. Since $I(q_1)$ obviously cannot exceed $|D_{G_1^*}(q_1)|$, we have:

$$\mathbf{E}[I(q_1)] \leq \frac{|D_{G_1^*}(q_1)|}{1 + |D_{G_1^*}(q_1)|} < 1.$$

Inductive Case. Assuming $\mathbf{E}[X] \leq k_0$ when $k_0 = t - 1$ for some integer $t \geq 2$, we will prove that the same holds also for $k_0 = t$.

Define $J(i)$ ($i \in [1, t]$) as the event that q_i has the largest permutation rank among $q_1, q_2, ..., q_t$. It suffices to prove:

$$\mathbf{E}[X \mid J(t)] \quad \leq \quad t. \tag{5}$$

By symmetry, this means

$$\mathbf{E}[X] \quad = \quad \sum_{i=1}^{t} \mathbf{E}[X \mid J(i)] \cdot \mathbf{Pr}[J(i)] \leq \sum_{i=1}^{t} t \cdot \frac{1}{t} = t$$

as desired. The subsequent discussion is conditioned on $J(t)$.

RPE-perm probes points in ascending order of rank. Define the *watershed moment* as:

- The moment right before RPE-perm probes the *first* point with a *larger* rank than all of $q_1, q_2, ..., q_{t-1}$;
- End of RPE-perm, if it does not probe any point that ranks after $q_1, q_2, ..., q_{t-1}$.

At the watershed moment, $I(q_1), ..., I(q_{t-1})$ have been finalized. Set $Y = \sum_{i=1}^{t-1} I(q_i)$. Denote by P_{water} the content of P at this instant.

The inductive assumption implies that $\mathbf{E}[Y] \leq t - 1$! To understand why, imagine deleting q_t from P, after which the input set P' has $t - 1$ label-0 noise points, but the same G_1^*. The permutation after removing q_t is a random permutation of P'. Thus, Y is exactly the value of $error(\mathcal{F}, G_1^*)$ on P'.

The remainder of the proof shows $\mathbf{E}[I(q_t) \mid J(t)] \leq 1$. This will establish (5) because

$$\mathbf{E}[X \mid J(t)] = \mathbf{E}[Y] + \mathbf{E}[I(q_t) \mid J(t)].$$

$I(q_t) = 0$ when q_t is not in P_{water} (and hence, will not be probed). Hence, it suffices to prove

$$\mathbf{E}[I(q_t) \mid J(t), q_t \in P_{water}] \leq 1.$$

Towards the purpose, we expand the left hand side over all possible sets W that P_{water} may be equal to:

$$\begin{aligned} & \mathbf{E}[I(q_t) \mid J(t), q_t \in P_{water}] \\ = \quad & \sum_{W} \mathbf{E}[I(q_t) \mid J(t), q_t \in P_{water} = W] \cdot \mathbf{Pr}[W]. \end{aligned} \tag{6}$$

We will concentrate on proving that

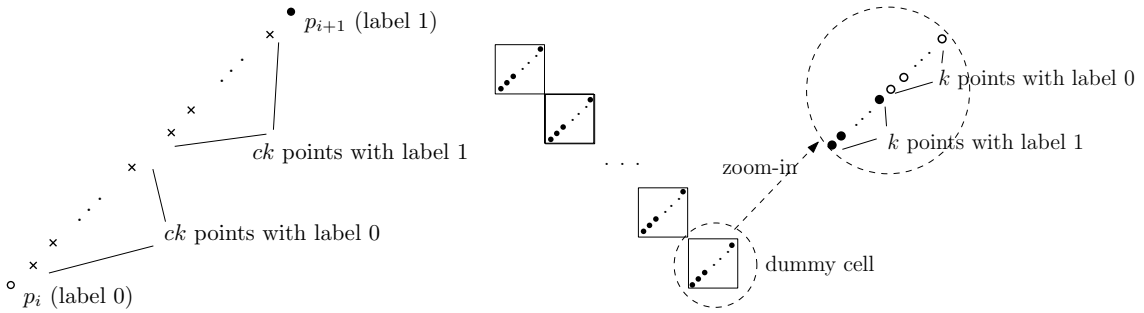
$$\mathbf{E}[I(q_t) \mid J(t), q_t \in P_{water} = W] \leq 1$$

regardless of W, with which (6) can be bounded from above by $\sum_W \mathbf{Pr}[W] = 1$.

Subject to the joint event "$J(t)$ and $q_t \in P_{water} = W$", the elements of W are symmetric with respect to their *relative* ordering in the permutation: any of the $|W|!$ orderings can take place with an equal probability. The analysis of $\mathbf{E}[I(q_t)]$ under that joint event is essentially the same as the base case. By the same argument as in the proof of Lemma 5, we assert that $I(q_t) > 0$ only if q_t ranks before all the points in $D_{W \cap G_1^*}(q_t)$, which happens with a probability of $1/(1 + |D_{W \cap G_1^*}(q_t)|)$. As $I(q_t)$ cannot exceed $|D_{W \cap G_1^*}(q_t)|$ under the joint event, we conclude that $\mathbf{E}[I(q_t) \mid J(t), q_t \in P_{water} = W]$ is no more than $\frac{|D_{W \cap G_1^*}(q_t)|}{1 + |D_{W \cap G_1^*}(q_t)|} \leq 1$.

3.3 Bounding the Probing Cost

Interestingly, the proof of Lemma 2 still holds verbatim even when P is noisy. Indeed, the probing behavior of RPE is captured by the attrition-and-elimination game regardless of whether noise exists in P. We have finished proving the first bullet of Theorem 1.



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

Figure 5: Adding dummy points for a Las Vegas lower bound

3.4 Lower Bound

We will generalize the argument of Section 2.3 to prove the second bullet of Theorem 1 for minimum monotone error $k \geq 1$.

Algorithms with Guessing Power. We first extend the model in Section 2.3 to strengthen the power of a deterministic algorithm A_{det}. As before, A_{det} is described by a binary decision tree T where each leaf returns a monotone classifier. However, there are two types of internal nodes:

- *Probe node*: At such a node, A_{det} probes a point p of P, and branches left (or right) if $label(p) = 0$ (or 1, resp.). This was the only type of internal nodes allowed in Section 2.3.
- *Guess node*: At such a node, A_{det} proposes a monotone classifier $\mathcal{F}_{guess}$, and asks an almighty guru *whether* $error(\mathcal{F}_{guess}, P) \leq K$ where K an arbitrary value fixed at this node. On a "yes" answer from the guru, A_{det} descends to the left child, which must be a leaf returning $\mathcal{F}_{guess}$. On a "no" answer, A_{det} branches right and continues.

We charge one unit of cost to every probe and guess node. A *randomized algorithm* is still modeled as a function that maps a random-bit sequence to a deterministic algorithm.

An almighty guru may not exist in reality. However, for proving lower bounds, we can increase the power of an algorithm at will. Any lower bounds on such "strong" algorithms must also hold on the algorithms that do not use guess nodes.

The argument in Section 2.3 essentially proved that, any deterministic algorithm of a binary decision tree must incur $\Omega(w(1 + \log \frac{n}{w}))$ average cost over the set $\mathcal{H}(n, w)$ of noise-free hard inputs (review Section 2.3 for the definition of "average cost", and the construction of $\mathcal{H}(n, w)$). Hence, this is also true for a deterministic algorithm with guess nodes. By Yao's minimax theorem, any randomized algorithm with guess nodes must entail $\Omega(w(1 + \log \frac{n}{w}))$ expected cost on at least one input of $\mathcal{H}(n, w)$. This holds for any integers w, n such that n is a multiple of w.

A Las Vegas Lower Bound. Suppose that A_{ran} is a randomized algorithm (with guessing) having the following guarantee: for any input P of n points, dominance width w, and minimum monotone error k, A_{ran} must (i) return a monotone classifier with error at most ck for some constant $c > 0$, and (ii) probe no more than $g(n, w)$ points in expectation. We will show that $g(n, w) = \Omega(w(1 + \log \frac{n}{kw}))$.

We can leverage A_{ran} to solve a (noise-free) input P' from $\mathcal{H}(n', w')$ by constructing a set P of size $n = O(n'k)$ as follows. Recall that the points of P' are placed in w' cells, each with n'/w' points, as shown in Figure 3. Duplicate P' to P. For each cell, add (to P) a dummy point with label 0 (or 1) infinitesimally close to the bottom-left (or upper-right, resp.) corner. Now, list out the points $p_1, p_2, ..., p_{2+\frac{n}{w}}$ in the cell, sorted in ascending order of y-coordinate. For every $i \in [1, 1 + \frac{n}{w}]$, place $2ck$ dummy points evenly on the cell's main diagonal between p_i and p_{i+1}, and decide their labels as follows:

- If $label(p_i) = label(p_{i+1})$, set the labels of all $2ck$ dummy points to $label(p_i)$.
- Otherwise ($label(p_i) = 0$ and $label(p_{i+1}) = 1$), set the labels of the lowest (or highest) ck dummy points to 0 (or 1, resp.); see Figure 5a.

The set P designed so far does not have minimum monotone error k (in fact, its minimum monotone error is 0). To fix this, add another dummy cell to P making sure that no point in the dummy cell can dominate any existing point in P, and vice versa (see Figure 5b). Add $2k$ dummy points on the main diagonal of the dummy cell, setting the labels of the k lowest (or highest) points to 1 (or 0), respectively. Now P has minimum monotone error k because (i) any monotone classifier must mis-label at least k points in the dummy cell, and (ii) obviously there is a monotone classifier $\mathcal{F}^*$ with error k ($\mathcal{F}^*$ correctly captures the labels of all the points in non-dummy cells, and simply maps all the points in the dummy cell to 1). Note also that P has dominance width $w' + 1$.

Suppose that A_{ran} returns a monotone classifier $\mathcal{F}$ on P. We argue that $\mathcal{F}$ cannot mis-label any non-dummy point p (i.e., p originated from P'). Otherwise, suppose that $label(p) = 0$ but $\mathcal{F}(p) = 1$. By our construction, there are at least ck label-0 points that (i) are in the same cell as p, and (ii) dominate p. As $\mathcal{F}$ maps all these points incorrectly to 1, implying that $error(\mathcal{F}, P) \geq ck + 1$, contradicting the guarantee of A_{ran}. A symmetric argument shows the impossibility of $label(p) = 0$ but $\mathcal{F}(p) = 1$. We can thus return $\mathcal{F}$ as a perfect classifier for P'.

It follows from our earlier lower bound on $\mathcal{H}(n, w)$ that:

$$g(O(n'k), w' + 1) = \Omega\left(w'\left(1 + \log \frac{n'}{w'}\right)\right).$$

Therefore, for $g(n, w)$ must be $\Omega(w(1 + \log \frac{n}{kw}))$ when $w \geq 2$ and $n \geq c_1 kw$ for a sufficiently large constant $c_1 > 0$.

A Monte-Carlo Lower Bound. Let A_{ran} be a randomized algorithm with a guarantee of the form: on any noisy input P of n points, dominance width w, and minimum monotone error k, A_{ran} has (i) expected error at most ck for some constant $c > 0$, and (ii) expected probing cost $f(n, w)$. We will prove that $f(n, w) = \Omega(w(1 + \log \frac{n}{kw}))$.

On such an input P, with probability at least 1/2, A_{ran} must (i) return a monotone classifier that has error at most $4ck$, and (ii) probes at most $4f(n, w)$ points. Otherwise, one of the following holds with at least 1/4 probability: either A_{ran} has error more than $4ck$, or A_{ran} probes more than $4f(n, w)$ points. However, this means that A_{ran} either has expected error higher than ck, or expected probing cost higher than $f(n, w)$, contradicting its guarantee.

We can deploy A_{ran} as a black box to design a randomized algorithm that always returns a monotone classifier of error at most $4ck$, and probes $O(f(n, w))$ points in expectation. For this purpose, run A_{ran} until either it returns a monotone classifier $\mathcal{F}$, or has probed $4f(n, w)$ points. In the former situation, we ask the almighty guru whether $error(\mathcal{F}, P) \leq 4ck$. If so, return $\mathcal{F}$. In all other situations (i.e., the guru answers "no" or A_{ran} did not terminate after $4f(n, w)$ probes), we declare "failure", and start all over again. If, however, we have failed $\lceil \log_2 n \rceil$ times, we simply probe the entire P, and return an optimal monotone classifier with error k. Clearly, we always return a classifier with error at most $4ck$. Since each time we fail with probability at most 1/2, the expected probing cost is bounded by

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

where the term $1/n$ on the left hand side bounds from above the probability of failing $\lceil \log_2 n \rceil$ times.

From our Las Vegas lower bound, we have that $f(n, w) = \Omega(w(1 + \log \frac{n}{kw}))$ when $w \geq 2$ and $n \geq c_2 kw$ for some constant $c_2 > 0$. This proves the second bullet of Theorem 1.

4 THE NOISY CASE—PROBLEM 2

The core of Problem 2 is to understand to what extent the exception set Z would help. Our study will concentrate on the lower bound side (for upper bounds, see Corollary 1). We will prove Theorem 2, which says that one cannot reduce the expected error of Corollary 1 by more than a constant factor without increasing the probing cost substantially.

We will focus on 1D space ($d = 1$). Remember that, at this dimensionality, every set P of points has dominance width $w = 1$; hence, this parameter is not relevant in the subsequent discussion. Deterministic and randomized algorithms are modeled as explained in Section 2.3 (i.e., no guess nodes).

4.1 Lower Bounds for $k = 1$

Given an integer $n \geq 2$, we define a set $\mathcal{S}_n$ of inputs as follows. Let P be the set $\{1, 2, ..., n\}$. Choose a *seed* integer $\sigma \in [2, n]$, and set $label(\sigma) = 0$. For every other point $p \in P \setminus \{\sigma\}$, set $label(p) = 1$. $\mathcal{S}_n$ collects all the $n - 1$ inputs with different seeds.

Notice that every $P \in \mathcal{S}_n$ has minimum monotone error $k = 1$. Indeed, $label(1) = 1$ and $label(\sigma) = 0$ rule out the existence of a perfect monotone classifier with error 0. On the other hand, there exist monotone classifiers that achieve error 1, e.g., the *all-true classifier*:

$$\mathcal{F}_{true}(p) = 1 \text{ for all } p \in \mathbb{R}.$$

Fix a (deterministic or randomized) algorithm A. Given an input $P \in \mathcal{S}_n$, let $\mathcal{F}_P$ and Z_P represent the monotone classifier and exception set returned by A, respectively. We define the *total error* of A on $\mathcal{S}_n$ as:

$$totalerr_n(A) = \sum_{P \in \mathcal{S}_n} error(\mathcal{F}_P, Z_P)$$

and the *total cost* of P on $\mathcal{S}_n$ as:

$$totalcost_n(A) = \sum_{P \in \mathcal{S}_n} |Z_P|.$$

Note that, when A is randomized, $\mathcal{F}_P$, Z_P, $error(\mathcal{F}_P, Z_P)$, $totalerr_n(A)$, and $totalcost_n(A)$ are all random variables.

Deterministic Algorithms. We will establish a tradeoff between $totalerr_n(A_{det})$ and $totalcost_n(A_{det})$ for any deterministic algorithm A_{det}. Let us first see two extremes of the tradeoff. On one extreme, the "probe-all" algorithm, which simply probes the entire P, achieves a total error of 0 but has a total cost of $n(n-1)$. At the other extreme, the "lazy" algorithm, which directly returns the all-true classifier $\mathcal{F}_{true}$ with no probing at all, has a total cost of 0 but a total error of $n - 1$. The following lemma states that, interestingly, $\Omega(n^2)$ total cost is necessary even just to do slightly better than the lazy algorithm in total error:

Lemma 6. *If $totalerr_n(A_{det}) \leq cn$ for an arbitrary constant $c < 1$, then $totalcost_n(A_{det}) = \Omega(n^2)$.*

Proof of Lemma 6. As before, a deterministic algorithm A_{det} can be modeled as a decision tree T. We say that A_{det} is *canonical* if all the conditions below hold:

- T is a "right-deep" tree, namely, the left child of every internal node is a leaf.
- The first point probed by A_{det} is point 1.
- At termination, A_{det} always outputs $\mathcal{F}_{true}$.

The lemma below shows that canonical algorithms cannot be improved much:

Lemma 7. *For every non-canonical (deterministic) algorithm A, there is a canonical algorithm A' with $totalerr_n(A') \leq totalerr(A) + 1$, and $totalcost_n(A') \leq totalcost(A) + n - 1$.*

Proof. If A returns monotone classifier $\mathcal{F}$ and exception set Z on an input P, we will refer to $error(\mathcal{F}, Z)$ conveniently as the "error of A on P".

We will convert A gradually into a desired A'. Let T be the decision tree of A. First, if T does not probe point 1 at the root, add a new root that does so, and has the original root as the right child. This increases the total cost of A by $n - 1$ (since one more probe is needed for every input of $\mathcal{S}_n$), but cannot increase its total error. Now, let us move to the original root of T.

Suppose that, in general, we are standing at an internal node u of T which probes a point $p \in P$. Recall that A branches into the left child v_1 of u if $label(p) = 0$, or the right child v_2 otherwise. If $p = 1$, delete u and its left subtree (effectively, replace the subtree of u with that of v_2). Consider now $p \neq 1$. If v_1 is not a leaf, make it

so by deleting its subtree and setting $\mathcal{F}_{true}$ as its output. Obviously, this does not increase the total cost. Furthermore, the total error is not increased, either. To see why, recall that every input $P \in \mathcal{S}_n$ has only one label-0 point σ. Once σ is probed, A achieves 0 error on P by returning $\mathcal{F}_{true}$, making any extra probes redundant.

Repetitively doing the above modifies T into a right-deep tree, and takes us to the rightmost leaf z of T. Let $\mathcal{F}$ be the classifier that A returns at z, with the general form: $\mathcal{F}(p) = 1$ if $p \geq x$ for some constant $x \in \mathbb{R}$, and 0, otherwise. We change this by asking z to output $\mathcal{F}_{true}$ instead. The remainder of the proof will show that the change can increase the total error of A by at most 1. This will complete the proof because the current T is the decision tree of a canonical algorithm.

Suppose that A has probed $Z = \{p_1, p_2, ..., p_h\}$ to reach z. Define $\bar{Z} = \{1, 2, ..., n\} \setminus Z$, i.e., $\bar{Z}$ is the set of points that have not been probed. Let S be the set of inputs $P \in \mathcal{S}_n$ with seeds $\sigma \in \bar{Z}$. We will consider only the inputs in S, because A ensures error 0 on any $P \notin S$ no matter it outputs $\mathcal{F}$ or $\mathcal{F}_{true}$ at z. Clearly, $\mathcal{F}_{true}$ allows A to achieve total error of $|S| = |\bar{Z}| = n - h$, because $\mathcal{F}_{true}$ mis-labels the only label-0 point of every $P \in S$.

It remains to prove that A has total error at least $n - h - 1$ if it outputs $\mathcal{F}$ at z. Denote by p_{min} the smallest point in $\bar{Z}$. If $x \leq p_{min}$, $\mathcal{F}$ always maps the entire $\bar{Z}$ to 1, and hence, also gives total error $n - h$. On the other hand, if $x > p_{min}$, then $\mathcal{F}$ gives total error at least $n - h - 1$ because, for every $P \in S$ with seed $\sigma \neq p_{min}$, $\mathcal{F}$ mis-labels at least the label-1 point p_{min}. There are $n - h - 1$ such inputs P. □

Consider a canonical algorithm A_{det} whose decision tree T has h internal nodes. Let $p_1, p_2, ..., p_h$ be the sequence of points probed at those nodes in top-down order ($p_1 = 1$). As explained in the above proof, $totalerr_n(A_{det}) = n - h$. To calculate its total cost, observe that when the input P has seed $\sigma = p_i$ for $i \in [2, h]$, A_{det} performs i probes, while for each of the other $n - h$ inputs P with $\sigma \notin \{p_2, ..., p_h\}$, A_{det} probes h points. Therefore:

$$totalcost_n(A_{det}) = (n - h)h + \sum_{i=2}^{h} i > \frac{nh}{2} - 1$$

If the total error $n - h$ needs to be at most cn, then h must be at least $(1 - c)n$, rendering $totalcost_n(A_{det}) = \Omega(n^2)$.

Combining this with Lemma 7 shows that, to ensure total error at most cn, any non-canonical algorithms must also incur total cost $\Omega(n^2)$. This completes the proof of Lemma 6.

Randomized Algorithms. Lemma 6 implies the following lower bound for randomized algorithms:

Corollary 2. *For any randomized algorithm A_{ran}, if $\mathbf{E}[totalerr_n(A_{ran})] \leq n/2$, then A_{ran} has $\Omega(n)$ expected probing cost on at least one input of $\mathcal{S}_n$.*

Proof. It suffices to prove that $\mathbf{E}[totalcost_n(A_{ran})] = \Omega(n^2)$. Recall that A_{ran} can be regarded as a random variable that is drawn from a set $W(A_{ran})$ deterministic algorithms. We say that an algorithm $A_{det} \in W(A_{ran})$ is *accurate* if $totalerr_n(A_{det}) \leq 3n/4$. Define W_{acc} as the set of accurate algorithms in $W(A_{ran})$. Then, $\mathbf{Pr}[A_{ran} \in W_{acc}]$ must be at least 1/3. Otherwise, with probability at least 2/3, A_{ran} has a total error greater than $3n/4$, thus forcing $\mathbf{E}[totalerr_n(A_{ran})]$ to be over $n/2$ and giving a contradiction. By Lemma 6, however, every accurate algorithm must have a total cost of $\Omega(n^2)$. Therefore, $\mathbf{E}[totalcost_n(A_{ran})] \geq \Omega(n^2) \cdot \mathbf{Pr}[A_{ran} \in W_{acc}] = \Omega(n^2)$. □

The above corollary proves Theorem 2 for $k = 1$ because if A_{ran} guarantees expected error at most 1/2 on every input, it ensures $\mathbf{E}[totalerr_n(A_{ran})] \leq n/2$.

4.2 A Randomized Lower Bound for $k \geq 2$

In this subsection, we will prove:

Lemma 8. *For Problem 2, let A_{ran} be a randomized algorithm which, given an input set of size n and minimum monotone error $k \geq 2$, guarantees expected error at most $k/2$ and expected probing cost at most $f(n, k)$. Then, for $\mathcal{S}_n$ (as defined in Section 4.1), there exists a randomized algorithm which has expected total error at most $n/2$ and expected total cost at most $f(nk, k)$.*

Putting together the above lemma with Corollary 2 shows that $f(nk, k) = \Omega(n)$. This leads to $f(n, k) = \Omega(n/k)$, as claimed in Theorem 2.

Hard Inputs. To prove Lemma 8, we generate a set $\mathcal{S}_n^k$ of inputs which can be regarded as the cartesian product of k copies of $\mathcal{S}_n$. Specifically, for every $(P_1, P_2, ..., P_k) \in (\mathcal{S}_n)^k$, create a set P of nk points by concatenating—in ascending order of $i \in [1, k]$—the sequence of n points in P_i. Each point $p \in P_i$ retains its label in P. This spawns $(n - 1)^k$ different input sets for P, which constitute $\mathcal{S}_n^k$.

Lemma 9. *Each input set $P \in \mathcal{S}_n^k$ has a minimum monotone error k.*

Proof. No monotone classifier $\mathcal{F}$ can ensure $error(\mathcal{F}, P) < k$. Otherwise, there exists at least one $i \in [1, k]$ such that $\mathcal{F}$ does not mis-label any point in P_i. This, however, contradicts the fact that P_i has minimum monotone error 1. On the other hand, the all-true classifier (i.e., $\mathcal{F}_{true}(p) = 1$ for all $p \in \mathbb{R}$) achieves $error(\mathcal{F}_{true}, P) = k$. □

***A_{ran}* Accurate on At Least One of $P_1, ..., P_k$.** Fix an input $P = (P_1, P_2, ..., P_k)$ of $\mathcal{S}_n^k$. Denote by $\mathcal{F}_P$ and Z_P the monotone classifier and exception set that A_{ran} (the randomized algorithm in Lemma 8) returns on P, respectively. For each $i \in [1, k]$, define $X_i(P_1, ..., P_k)$ as the number of mis-labeled points from P_i, or formally:

$$\begin{aligned} & X_i(P_1, ..., P_k) \\ = \; & |\{p \in P_i \mid p \notin Z_P \text{ and } \mathcal{F}(p) \neq label(p)\}|. \end{aligned}$$

Clearly, $\sum_{i=1}^{k} X_i(P_1, ..., P_k) = error(\mathcal{F}_P, P \setminus Z_P)$. Since A_{ran} guarantees $\mathbf{E}[error(\mathcal{F}_P, P \setminus Z_P)] \leq k/2$, we know:

$$\begin{aligned} \sum_{i=1}^{k} \mathbf{E}[X_i(P_1, ..., P_k)] &\leq k/2 \\ \Rightarrow \sum_{(P_1, ..., P_k) \in \mathcal{S}_n^k} \sum_{i=1}^{k} \mathbf{E}[X_i(P_1, ..., P_k)] &\leq \frac{k}{2} |\mathcal{S}_n^k| \qquad (7) \end{aligned}$$

For each $i \in [1,k]$, define:

$$Y_i \quad = \quad \sum_{P_i \in \mathcal{S}_n} \sum_{(P_1,\ldots,P_{i-1},P_{i+1},\ldots,P_k) \in (\mathcal{S}_n)^{k-1}} X_i(P_1,...,P_k).$$

From (7), we have:

$$\sum_{i=1}^{k} \mathbf{E}[Y_i] \quad \leq \quad \frac{k}{2}|\mathcal{S}_n^k|.$$

Therefore, there must be at least one $i \in [1,k]$ satisfying $\mathbf{E}[Y_i] \leq |\mathcal{S}_n^k|/2$. Henceforth, we will use j to denote this value of i. Intuitively, this means that A_{ran} does not mis-label too many points from P_j overall.

Designing a Good Algorithm for $\mathcal{S}_n$. We now deploy A_{ran} to design an algorithm that achieves expected total error at most $n/2$ on $\mathcal{S}_n$ with expected probing cost at most $f(nk,n)$.

Given an input P' of $\mathcal{S}_n$, we construct an input $P = (P_1,...,P_k)$ in $\mathcal{S}_n^k$ by (i) setting $P_j = P$, and (ii) independently choosing P_i uniformly at random from $\mathcal{S}_n$ for every $i \in [1,k] \setminus \{j\}$. Then, run A_{ran} on P. Every time it asks to probe a point $p \in P_i$ for $i \neq j$, we reveal $label(p)$ directly on behalf of the oracle. Only when A_{ran} asks to probe a point $p \in P_j$ will we relay the request to the oracle. Suppose that A_{ran} returns a monotone classifier $\mathcal{F}$ and exception set Z, we return $\mathcal{F}$ and $Z \cap P_j$. It is clear that our algorithm—referred to as A'_{ran} henceforth—probes at most as many points on P' as A_{ran}. Thus, the expected probing cost of A'_{ran} is at most $f(nk,n)$.

It remains to prove that A'_{ran} achieves an expected total error of at most $n/2$ on $\mathcal{S}_n$, that is, $\mathbf{E}[totalerr_n(A'_{ran})] \leq n/2$. Observe that:

$$\begin{aligned} & \mathbf{E}[error(\mathcal{F}, P' \setminus Z)] \\ = \quad & \frac{1}{(n-1)^{k-1}} \cdot \Big(\sum_{(P_1,\ldots,P_{j-1},P_{j+1},\ldots,P_k) \in (\mathcal{S}_n)^{k-1}} \\ & \mathbf{E}[X_j(P_1,...,P_{j-1},P',P_{j+1},...P_k)] \Big). \end{aligned}$$

Thus, $\mathbf{E}[totalerr_n(A'_{ran})]$, which sums up $\mathbf{E}[error(\mathcal{F}, P' \setminus Z)]$ for all $P' \in \mathcal{S}_n$, equals

$$\frac{\sum_{P_j \in \mathcal{S}_n} \sum_{(P_1,\ldots,P_{j-1},P_{j+1},\ldots,P_k) \in (\mathcal{S}_n)^{k-1}} \mathbf{E}[X_j(P_1,...,P_k)]}{(n-1)^{k-1}}$$

$$= \frac{\mathbf{E}[Y_j]}{(n-1)^{k-1}}.$$

By definition of j, the above is at most $\frac{1}{2}|S_n^k|/(n-1)^{k-1} = (n-1)/2$.

This completes the proof of Lemma 8, and hence, the proof of Theorem 2.

5 CONCLUSIONS

This paper has studied active learning problems that model the core of entity matching in two important scenarios (the first: the input set represents the distribution of entity pairs to be classified; and the second: the input set already includes all the entity pairs to be classified). We have designed new algorithms that are able to strike a non-trivial balance between the probing cost and the accuracy of matching. We have also proved hardness results showing that our algorithms are asymptotically optimal in a variety of parameter ranges.

Interestingly, our solutions manage to improve the state of the art (i.e., the A^2 algorithm) for agnostic active learning. For fairness, this should be taken with a grain of salt: the proposed algorithms focus on one type of classifiers (i.e., monotone classifiers), while A^2 aims to support arbitrary types of classifiers. Nevertheless, our analysis—upper and lower bounds combined—indicates that disagreement coefficient and VC dimension are not suitable parameters for characterizing the hardness of learning monotone classifiers. The phenomenon is intriguing: does it mean that monotone classifiers fall into an unknown generic class of learning problems that has escaped the literature of active learning? If so, what is that class, and what are *its* intrinsic hardness-defining parameters?

REFERENCES

[1] Arvind Arasu, Michaela Götz, and Raghav Kaushik. 2010. On active learning of record matching packages. In *SIGMOD*. 783–794.
[2] Maria-Florina Balcan, Alina Beygelzimer, and John Langford. 2009. Agnostic active learning. *JCSS* 75, 1 (2009), 78–89.
[3] Kedar Bellare, Suresh Iyengar, Aditya G. Parameswaran, and Vibhor Rastogi. 2013. Active Sampling for Entity Matching with Guarantees. *TKDD* 7, 3 (2013), 12:1–12:24.
[4] Alina Beygelzimer, Sanjoy Dasgupta, and John Langford. 2009. Importance weighted active learning. In *ICML*. 49–56.
[5] Guilherme Dal Bianco, Renata Galante, Marcos Andre Goncalves, Sergio D. Canuto, and Carlos Alberto Heuser. 2015. A Practical and Effective Sampling Selection Strategy for Large Scale Deduplication. *TKDE* 27, 9 (2015), 2305–2319.
[6] Peter Christen, Dinusha Vatsalan, and Qing Wang. 2015. Efficient Entity Resolution with Adaptive and Interactive Training Data Selection. In *ICDM*. 727–732.
[7] Xu Chu, Ihab F. Ilyas, and Paraschos Koutris. 2016. Distributed Data Deduplication. *PVLDB* 9, 11 (2016), 864–875.
[8] Sanjoy Dasgupta. 2005. Coarse sample complexity bounds for active learning. In *NIPS*. 235–242.
[9] Sanjoy Dasgupta, Daniel J. Hsu, and Claire Monteleoni. 2007. A general agnostic active learning algorithm. In *NIPS*. 353–360.
[10] 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.
[11] Mark de Berg, Otfried Cheong, Marc van Kreveld, and Mark Overmars. 2008. *Computational Geometry: Algorithms and Applications* (3rd ed.). Springer-Verlag.
[12] 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.
[13] 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.
[14] 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 *SIGMOD*. 601–612.
[15] Steve Hanneke. 2007. A bound on the label complexity of agnostic active learning. In *ICML*. 353–360.
[16] Steve Hanneke. 2014. Theory of Disagreement-Based Active Learning. *Foundations and Trends in Machine Learning* 7, 2-3 (2014), 131–309.
[17] S. Ikiz and K. Vijay Garg. [n. d.]. Online Algorithms for Dilworth's Chain Partition. *Tech. Rep., Department of Electrical and Computer Engineering, University of Texas at Austin.* ([n. d.]).
[18] Matti Kääriäinen. 2006. Active Learning in the Non-realizable Case. In *Proceedings of International Conference on Algorithmic Learning Theory (ALT)*. 63–77.
[19] Lars Kolb, Andreas Thor, and Erhard Rahm. 2012. Load Balancing for MapReduce-based Entity Resolution. In *ICDE*. 618–629.
[20] Pradap Konda, Sanjib Das, Paul Suganthan G. C., AnHai Doan, Adel Ardalan, Jeffrey R. Ballard, Han Li, Fatemah Panahi, Haojun Zhang, Jeffrey F. Naughton, Shishir Prasad, Ganesh Krishnan, Rohit Deep, and Vijay Raghavendra. 2016. Magellan: Toward Building Entity Matching Management Systems. *PVLDB* 9, 12 (2016), 1197–1208.
[21] Hanna Köpcke and Erhard Rahm. 2010. Frameworks for entity matching: A comparison. *DKE* 69, 2 (2010), 197–210.
[22] Hanna Köpcke, Andreas Thor, and Erhard Rahm. 2010. Evaluation of entity resolution approaches on real-world match problems. *PVLDB* 3, 1 (2010), 484–493.
[23] Rajeev Motwani and Prabhakar Raghavan. 1995. *Randomized Algorithms*. Cambridge University Press.

[24] Sunita Sarawagi and Anuradha Bhamidipaty. 2002. Interactive deduplication using active learning. In *SIGKDD*. 269–278.
[25] Burr Settles. 2010. Active learning Literature Survey. *Technical Report, University of Wisconsin-Madison* (2010).
[26] Andreas Thor and Erhard Rahm. 2007. MOMA - A Mapping-based Object Matching System. In *CIDR*. 247–258.
[27] Liwei Wang. 2011. Smoothness, Disagreement Coefficient, and the Label Complexity of Agnostic Active Learning. *Journal of Machine Learning Research* 12 (2011), 2269–2292.

A: VC DIMENSION AND DISAGREEMENT COEFFICIENT OF MONOTONE CLASSIFIERS

Let U be an arbitrary set of n points in $\mathbb{R}^2$ with dominance width w. We will show that the class $\mathbb{C}_{mono}$ of monotone classifiers has a VC dimension w on U (review Section 1.2 for the definition of VC dimension). In fact, simply take an arbitrary anti-chain S of U with size w. For any $(l_1, l_2, ..., l_w) \in \{0,1\}^w$, assign label l_i to the i-th point in S. Then, construct a classifier $\mathcal{F}$ as follows. First, for each point $p \in S$, $\mathcal{F}(p)$ equals the label just assigned. Second, for any point $p' \in \mathbb{R}^2$ outside S, $\mathcal{F}(p') = 1$ if p' dominates a label-1 point in S, or 0 otherwise. $\mathcal{F}$ is indeed monotone. Therefore, S can be shattered by $\mathbb{C}_{mono}$.

We now switch our attention to disagreement coefficient. The purpose is to compare Theorem 1 to the upper bound of the A^2 algorithm given in (1). Hence, our discussion will concentrate on the context where Problem 1 with input set P is reduced to agnostic active learning with $U = P$ in the way described in Section 1.2. As explained in Section 1.3, ν and ϵ are both set to k/n in order to match of error bound in Theorem 1.

Given a set $X \subseteq U$, denote by $\mathbf{Pr}(X) = |X|/n$ the probability that a point drawn uniformly at random from U falls in S. Given a set $C \subseteq \mathbb{C}_{mono}$ of monotone classifiers. The *disagreement region* of C—denoted as $\text{DIS}(C)$—as the set of points p in U such that $\mathcal{F}_1(p) \neq \mathcal{F}_2(p)$ for some $\mathcal{F}_1, \mathcal{F}_2$ in C. Note that $\mathbf{Pr}(\text{DIS}(C))$ gives the probability that, not all the classifiers in C agree on the label of a point p taken uniformly at random from U.

Let $\mathcal{F}^*$ be an arbitrary monotone classifier attaining the smallest error, namely, $error(\mathcal{F}^*, P) = k > 0$. Let r be a real value in the range $(2k/n, 1]$. Define the *ball* $B(\mathcal{F}^*, r)$ as the set of all classifiers $\mathcal{F} \in \mathbb{C}_{mono}$ such that $\mathbf{Pr}[\mathcal{F}^*(p) \neq \mathcal{F}(p)] \leq r$ when p is drawn uniformly at random from U. The *disagreement coefficient* θ [16] is defined as:

$$\theta = \max\left\{\sup_{r > \frac{2k}{n}} \frac{\mathbf{Pr}(\text{DIS}(B(\mathcal{F}^*, r)))}{r}, 1\right\}. \quad (8)$$

We will show that $\theta \geq \max\{\Omega(w/k), 1\}$ holds for *any* P (later, we will prove a much worse result for specific inputs). For this purpose, it suffices to look at a value of r that is infinitesimally larger than $2k/n$. Let S be an arbitrary anti-chain of size w. There is a classifier $\mathcal{F} \in \mathbb{C}_{mono}$ that agrees with $\mathcal{F}^*$ on the labels of all points in P, except for one point $p \in S$, regardless of how p is chosen. It thus follows that $\text{DIS}(B(\mathcal{F}^*, r)) \geq w$, and hence, $\mathbf{Pr}(\text{DIS}(B(\mathcal{F}^*, r))) \geq w/n$. This fulfills our purpose because $\mathbf{Pr}(\text{DIS}(B(\mathcal{F}^*, r)))/r$ is infinitesimally close to $\frac{w/n}{2k/n} = w/(2k)$.

Finally, we will construct a bad input P to force $\theta = \Omega(w)$. Fix values of n', w', k such that $2k \leq n'/w'$. First, decide the locations of n' points in P according to Figure 5, where the number of cells is w', and each cell has n'/w' points. Set the labels of all these points to be 1. Finally, add a dummy cell in the way illustrated in Figure 5b. Add to P $2k$ points that are placed on the main diagonal of that cell, with the labels of the k highest (or lowest) points set to 0 (or 1), respectively. It is easy to verify that P has minimum monotone error k. Indeed, the best $\mathcal{F}^*$ simply maps all points to 1. Furthermore, P has size $n = n' + 2k$, and dominance width $w = w' + 1$.

We will show that $\text{DIS}(B(\mathcal{F}^*, r)) \geq 2w'k$. Again, set r to be infinitesimally larger than $2k/n$. Pick an arbitrary non-dummy cell. There is obviously a classifier $\mathcal{F} \in \mathbb{C}_{mono}$ that agrees with $\mathcal{F}^*$ on the labels of all points, except the $2k$ lowest ones in the selected cell. This means that every non-dummy cell contributes $2k$ points to $\text{DIS}(B(\mathcal{F}^*, r))$, which thus has a size of at least $2w'k$. Therefore, $\mathbf{Pr}(\text{DIS}(B(\mathcal{F}^*, r)))/r$ is infinitesimally close to $\frac{2w'k/n}{2k/n} = w' = \Omega(w)$.

B: PROOF OF PROPOSITION 1

We will first prove that Z (the set of points probed by RPE) obeys monotonicity (even though P does not): for any $p, q \in Z$, if p dominates q, then $label(p) \geq label(q)$.

Assume, on the contrary, that this is not true, meaning that $label(p) = 0$ and $label(q) = 1$. But which point was probed earlier by RPE? If it was p, then q must have been discarded after discovering that p is label-0. Likewise, probing q first would have discarded p. Therefore, it is impossible that both p and q were probed.

Z satisfying monotonicity implies that $\mathcal{F}$ must be monotone. Otherwise, there exist p, q such that p dominates q, but $\mathcal{F}(p) = 0$, $\mathcal{F}(q) = 1$. By (3), $\mathcal{F}(q) = 1$ means that Z has a label-1 point that is dominated by q, and hence, also dominated by p. This contradicts the fact that $\mathcal{F}(p)$ is 0.

C: PROOF OF PROPOSITION 2

We prove first the "if" direction. If p is dominated by a label-0 point in Z, p cannot dominate any label-1 point in Z, due to the monotonicity of Z (see the proof of Proposition 1). Hence, $\mathcal{F}(p) = 0$.

It remains to prove the "only-if" direction. By (3), $\mathcal{F}(p) = 0$ means that p does not dominate any label-1 point in Z. Hence, the disappearance of p from P must be because p is dominated by a label-0 point that was probed (remember that a point dominates itself).

D: EQUIVALENCE OF RPE AND RPE-PERM

We will prove:

Lemma 10. *RPE and RPE-perm have the same expected error and expected probing cost on every input P.*

Both RPE and RPE-perm can be described as a randomized decision tree T defined as follows. Each node u of T is associated with a subset of P, denoted as $u(P)$. If u is the root, $u(P) = P$, whereas if u is a leaf, $u(P) = \emptyset$. An internal node u has $|u(P)|$ child nodes. Each directed edge (u, v) from u to a child v stores a point—denoted as $point(u, v)$—of $u(P)$. Every point of $u(P)$ is stored on one and exactly one outgoing edge of u. For each child v, the set $v(P)$ is determined as:

- If $label(p) = 0$, $v(P)$ is the set of points in $u(P)$ that are not dominated by $point(u, v)$ (recall that a point dominates itself).

- If $label(p) = 1$, $v(P)$ is the set of points in $u(P)$ that do not dominate $point(u, v)$.

Each root-to-leaf path π represents a possible probing sequence of RPE or RPE-perm. Specifically, for each node u on π, $u(P)$ represents the content of P after the algorithm probes the points stored on (the edges of) the root-to-u path.

We will prove that, for every leaf z of T, RPE and RPE-perm reach z with exactly the same probability. This establishes Lemma 10 because both error and probing cost are determined by the sequence of points probed.

Let $u_1, u_2, ..., u_\ell$ the nodes on the root-to-z path (u_1 is the root and $z = u_\ell$). Obviously, RPE reaches z with probability $\Pi_{i=1}^{\ell-1} \frac{1}{|u_i(P)|}$. It remains to show that this is also true for RPE-perm.

The execution of RPE-perm is a function of the permutation of P—denoted as P_{perm}—obtained at Step 1. For each node u of T, denote by $S(u)$ the set of all possible P_{perm} that will bring the execution to u. When u is the root, $S(u)$ is the set of all $n!$ permutations.

Lemma 11. *For $i \in [2, \ell]$, $S(u_i)$ is the set of permutations $\pi \in S(u_{i-1})$ such that $point(u_{i-1}, u_i)$ has the smallest rank in π among all the points in $P(u_{i-1})$.*

Proof. We prove the claim by induction. It holds for $i = 2$ because RPE-perm descends from u_1 (the root) to u_2 only when $point(u_1, u_2)$ is the first point of P_{perm}. Inductively, assume that the claim is true for $i = j-1$. As mentioned before, $P(u_{j-1})$ is the content of P after RPE-perm probes the points stored on the root-to-u_{j-1} path. Hence, the algorithm branches to u_j only if $point(u_{j-1}, u_j)$ is the next to pick in P_{perm} among the points in $P(u_{j-1})$. So the claim holds also for $i = j$. □

The lemma indicates that $|S(u_i)| = |S(u_{i-1})|/|u_{i-1}(P)|$. Hence, $|S(u_\ell)| = |S(u_1)| \cdot \Pi_{i=1}^{\ell-1} \frac{1}{|u_i(P)|}$. The probability that RPE-perm reaches u_ℓ equals $|S(u_\ell)|/n!$ which is simply $\Pi_{i=1}^{\ell-1} \frac{1}{|u_i(P)|}$.