Computing Data Distribution from Query Selectivities

Pankaj K. Agarwal
Department of Computer Science, Duke University, Durham, NC, USA

Rahul Raychaudhury
Department of Computer Science, Duke University, Durham, NC, USA

Stavros Sintos
Department of Computer Science, University of Illinois at Chicago, IL, USA

Jun Yang
Department of Computer Science, Duke University, Durham, NC, USA

Abstract

We are given a set $\mathcal{Z} = \{(R_1, s_1), \ldots, (R_n, s_n)\}$, where each R_i is a *range* in $\mathbb{R}^d$, such as rectangle or ball, and $s_i \in [0, 1]$ denotes its *selectivity*. The goal is to compute a small-size *discrete data distribution* $\mathcal{D} = \{(q_1, w_1), \ldots, (q_m, w_m)\}$, where $q_j \in \mathbb{R}^d$ and $w_j \in [0, 1]$ for each $1 \leq j \leq m$, and $\sum_{1 \leq j \leq m} w_j = 1$, such that $\mathcal{D}$ is the most *consistent* with $\mathcal{Z}$, i.e., $\mathrm{err}_p(\mathcal{D}, \mathcal{Z}) = \frac{1}{n} \sum_{i=1}^{n} |s_i - \sum_{j=1}^{m} w_j \cdot \blacksquare(q_j \in R_i)|^p$ is minimized. In a database setting, $\mathcal{Z}$ corresponds to a workload of range queries over some table, together with their observed selectivities (i.e., fraction of tuples returned), and $\mathcal{D}$ can be used as compact model for approximating the data distribution within the table without accessing the underlying contents.

In this paper, we obtain both upper and lower bounds for this problem. In particular, we show that the problem of finding the best data distribution from selectivity queries is NP-complete. On the positive side, we describe a Monte Carlo algorithm that constructs, in time $O((n+\delta^{-d})\delta^{-2} \operatorname{polylog} n)$, a discrete distribution $\tilde{\mathcal{D}}$ of size $O(\delta^{-2})$, such that $\mathrm{err}_p(\tilde{\mathcal{D}}, \mathcal{Z}) \leq \min_{\mathcal{D}} \mathrm{err}_p(\mathcal{D}, \mathcal{Z}) + \delta$ (for $p = 1, 2, \infty$) where the minimum is taken over all discrete distributions. We also establish conditional lower bounds, which strongly indicate the infeasibility of relative approximations as well as removal of the exponential dependency on the dimension for additive approximations. This suggests that significant improvements to our algorithm are unlikely.

2012 ACM Subject Classification Theory of computation → Computational geometry

Keywords and phrases selectivity queries, discrete distributions, Multiplicative Weights Update, eps-approximation, learnable functions, depth problem, arrangement

Digital Object Identifier 10.4230/LIPIcs.ICDT.2024.18

Related Version *Full Version*: `https://arxiv.org/abs/2401.06047` [1]

1 Introduction

The *selectivity* of a selection query on a set of objects in a database is the probability of a random object in the database satisfying the query predicate. A key step in query optimization, selectivity estimation is used by databases for estimating costs of alternative query processing plans and picking the best one. Consequently, selectivity estimation has been studied extensively in the last few decades [31, 36, 43, 44, 46].

Historically, selectivity estimation has been data-driven. These approaches construct, or dynamically maintain, a small-size synopsis of the data distribution using histograms or random samples that minimize estimation error. While these methods work well in low dimensions, they suffer from the curse of dimensionality. As a result, interest in learning-based methods for selectivity estimation has been growing over the years [24, 32, 33, 34, 38, 40]. Many different methods have been proposed that work with the data distribution, observed

selectivities from query workloads, or a combination of both. At a high-level, many of these techniques build a model of the underlying data distribution and use it to answer queries. While they work very well in practice, often outperforming their traditional counterparts, a theoretical understanding of this line of work is missing. This leads to the natural question, whether selectivity can be learned efficiently from a small sample of query selectivities alone, without access to the data distribution. Hu et al. [27] formalize the learnability of the selectivity estimation problem in this setting. They use the agnostic-learning framework [25], an extension of the classical PAC learning framework for real-valued functions, where one is given a set of sample queries from a fixed query distribution and their respective selectivities (the *training set*), and the goal is to efficiently construct a data distribution so that the selectivity of a new query from the same query distribution can be answered with high accuracy. They show that for a wide class of range queries, the selectivity query can be learned within error $\varepsilon \in (0,1)$ with probability at least $1-\delta$ using a training set of size $\varepsilon^{-O(1)} \log \delta^{-1}$, where the exponent of ε depends on the query type; see [27] for a precise statement of their results. Informally, learnability implies that performance of a model on the training set generalizes to unseen queries from the same distribution. This reduces the task of learning to finding a (model of the) data distribution that best fits the training data i.e. *Empirical Risk Minimization (ERM)*. Although Hu et al. [27] prove a sharp bound on the sample complexity, their algorithm for ERM takes prohibitively long and produces a data distribution of large size. They also present fast heuristics to construct small-size data distributions, but they do not provide any guarantee on the performance with respect to the best data distribution fitting the training set. This raises the question of how to develop a provably efficient and effective algorithm for constructing the best data distribution (in a given family) from a training set.

Note that the size of the distribution computed by [27], can further be reduced to $O(\varepsilon^{-2})$ by choosing an ε-approximation, with an increase of ε in the error; see [23] and Section 3 below. However, the paper aims at computing a small-size distribution (whose performance is comparable to the best data distribution) directly and efficiently, without constructing a large-size distribution first. For this problem, we obtain hardness results as well as efficient algorithms.

Problem Statement. A *range space* $\Sigma = (\mathcal{X}, \mathcal{R})$ comprises a set of *objects* $\mathcal{X}$ and a collection of subsets $\mathcal{R} \subseteq 2^{\mathcal{X}}$ called *ranges*. In this paper, $\mathcal{X}$ is a finite set of points, and each range $R \in \mathcal{R}$ corresponds to a query that returns the subset of objects that fall within a simple geometric region such as a rectangle (corresponding to an orthogonal range query), a ball (neighborhood range query), or a half-space (query with a linear constraint). We will not distinguish between a region R and $R \cap \mathcal{X}$, so with a slight abuse of notation, we will use $\mathcal{R}$ to denote a set of geometric regions such as a set of rectangles or balls.

A *discrete distribution* $\mathcal{D} = \{(p_1, w_1), ..., (p_m, w_m)\}$ is defined by a finite set of points and their associated probabilities, where each p_i is a point in $\mathbb{R}^d$ for some constant $d \geq 1$, each $w_i > 0$, and $\sum_{i=1}^{m} w_i = 1$. We refer to the point set $\{p_1, ..., p_m\}$ as the *support* of $\mathcal{D}$ and denote it by $\text{supp}(\mathcal{D})$. The *size* of a discrete distribution $\mathcal{D}$ is defined as the size of its support and is denoted as $|\mathcal{D}|$. Let $\mathbb{D}$ denote the family of all discrete distributions, and let $\mathbb{D}_k$ be the family of discrete distributions of size at most k. Given a range $R \subseteq \mathcal{R}$, let $s_{\mathcal{D}}(R) = \sum_{i=1}^{m} \blacksquare(p_i \in R) w_i$ denote the *selectivity* of R over $\mathcal{D}$, which is the probability for a random point drawn from $\mathcal{D}$ to lie in R (or the total measure of $\mathcal{D}$ inside R).

In this paper, our goal is to learn a (discrete) distribution from the selectivities of range queries. For ease of exposition, we will describe our results for rectangles (orthogonal ranges), although our techniques extend to other natural range spaces.

Let $\mathcal{Z} = \{z_1, ..., z_n\}$ be a set of *training samples*, where $z_i = (R_i, s_i)$, R_i is an axis-aligned rectangle (orthogonal range) in $\mathbb{R}^d$, and $s_i \in [0, 1]$ is its observed selectivity.[1] Let $\mathcal{R} = \{R_1, ..., R_n\}$ denote the set of ranges in $\mathcal{Z}$. For a discrete distribution $\mathcal{D} \in \mathbb{D}$ and for $p \geq 1$ we define the (ℓ_p) empirical error to be

$$\mathrm{err}_p(\mathcal{D}, \mathcal{Z}) = \frac{1}{n} \sum_{i=1}^{n} |s_{\mathcal{D}}(R_i) - s_i|^p, \text{ and for } p = \infty, \mathrm{err}_\infty(\mathcal{D}, \mathcal{Z}) = \max_{1 \leq i \leq n} |s_{\mathcal{D}}(R_i) - s_i|. \quad (1)$$

We focus on $p = 1, 2$, and ∞.

Let $\alpha_p^*(\mathcal{Z}) = \min_{\mathcal{D} \in \mathbb{D}} \mathrm{err}_p(\mathcal{D}, \mathcal{Z})$ denote the minimum error achievable for all distributions in the family. Given $\mathcal{Z}$, our goal is to compute a discrete distribution $\tilde{\mathcal{D}}_p$ with $\mathrm{err}(\tilde{\mathcal{D}}_p, \mathcal{Z}) = \alpha_p^*$. As argued in [27], such a discrete distribution of size $O(n^d)$ can be computed in $n^{O(d)}$ time. However, this distribution is too large even for moderate values of n and d, so we are interested in computing a smaller size distribution at the cost of a slight increase in the empirical error. Hence, our problem becomes the following: given $\mathcal{Z}$ and some $\delta \in [0, 1]$, compute a small-size distribution $\mathcal{D}$, ideally of size that depends on δ and independent of n, such that $\mathrm{err}_p(\mathcal{D}, \mathcal{Z}) \leq \alpha_p^* + \delta$. Alternatively, given a size budget k, compute a distribution $\hat{\mathcal{D}} \in \mathbb{D}_k$ such that $\mathrm{err}_p(\hat{\mathcal{D}}, \mathcal{Z}) = \min_{\mathcal{D} \in \mathbb{D}_k} \mathrm{err}_p(\mathcal{D}, \mathcal{Z})$.

Our results. We present both negative and positive results for the data-distribution[2] learning problem. On the negative side in Section 5, we prove that the problem is NP-complete even for rectangles in $\mathbb{R}^2$. Namely, given $\mathcal{Z}$ where $\mathcal{R}$ is a set of rectangles in $\mathbb{R}^2$, $p \in \{1, 2, \infty\}$, and two parameters $\delta \in [0, 1]$ and $k \geq 1$, the problem of determining whether there is a data distribution $\mathcal{D}$ of size k such that $\mathrm{err}_p(\mathcal{D}, \mathcal{Z}) \leq \delta$ is NP-hard.

On the positive side, we focus on designing an efficient algorithm for constructing a small-size distribution with additive-approximation error. We present a Monte Carlo algorithm that computes, with high probability, a discrete distribution $\tilde{\mathcal{D}}$ of size $O(\delta^{-2})$ with $\mathrm{err}_p(\tilde{\mathcal{D}}, \mathcal{Z}) \leq \alpha_p^* + \delta$, for $p \in \{1, 2, \infty\}$, in $O(n\delta^{-2} \log n + \delta^{-d-2} \log^3 n)$ time (exact time complexity is given in Lemma 8). Starting with $p = 1$ (ℓ_1 empirical error), we first present in Section 2 a basic algorithm that maps our problem to a linear program (LP) with $O(n)$ constraints and $O(n^d)$ variables. We show how the Multiplicative-Weight-Update (MWU) method [5] can be used to solve this LP. A naive implementation of the MWU of this algorithm takes $\Omega(n^d)$ time. Next, in Section 3, we exploit underlying geometry in two ways to solve this LP involving exponential number of variables efficiently, by representing it implicitly. First, we give a geometric interpretation to the main step of the MWU method: we map a maximization problem with $O(n^d)$ variables to the problem of finding the weighted deepest point in an arrangement of n rectangles in $\mathbb{R}^d$. The best algorithm to solve this geometric problem takes $O(n^{d/2})$ time [9], which would allow one to improve the running time to $O(\delta^{-2} n^{d/2} \log n)$. But this is also expensive. Second, we use the notion of ε-approximation to quickly compute an approximately deepest point in a weighted set of rectangles efficiently, in time $O(\delta^{-d} \log n)$.

[1] Note that we do not assume the training set $\mathcal{Z}$ to be consistent with any distribution distribution in $\mathbb{D}$, or any distribution in general; i.e., there might not exist any distribution $\mathcal{D}$ such that s_i reflects the selectivity of R_i for every $1 \leq i \leq n$. This flexibility allows us to model settings where the query workload was executed on an evolving database instance, and the observed selectivities on different concrete instances may not be consistent with each other.

[2] In this paper we focus on discrete distributions. For simplicity, sometimes we use the term data distribution instead of discrete distribution.

In Section 4 we extend our algorithm to more general settings. We show that our near-linear time algorithm works for the ℓ_∞ and the ℓ_2 empirical error. Furthermore, we show that our algorithm can be extended to other ranges such as balls and halfspaces in $\mathbb{R}^d$.

In Section 5, we also give conditional lower bounds that indicate that avoiding the exponential dependency on d is not possible even for additive approximations. This makes meaningful improvements to our algorithm unlikely. We also give conditional lower bounds for a variant of our problem, allowing an arbitrary relative approximation factor on the size of the distribution or an arbitrary relative approximation factor on the error of the returned distribution. Our conditional hardness results are based on the FPT $\neq W[1]$ conjecture; see [14] for the definition. Finally, we also show that when the distribution size is fixed, any relative approximation is NP-hard.

2 Basic Algorithm

In this section we present our basic algorithm for computing a small-size discrete distribution that (approximately) minimizes the ℓ_1 empirical error. For simplicity, let $\mathrm{err}(\cdot,\cdot)$ and α^* denote $\mathrm{err}_1(\cdot,\cdot)$ and α_1^*, respectively. Given a training set $\mathcal{Z}$ and a parameter $\delta \in (0,1)$, it computes a discrete distribution $\mathcal{D}^*$ of size $O(\delta^{-2}\log n)$ such that $\mathrm{err}(\mathcal{D}^*,\mathcal{Z}) = \alpha^* + \delta$. At the heart, there is a decision procedure IsFeasible that, given $\mathcal{Z}$ and parameters $\alpha, \delta \in (0,1)$, returns a distribution $\mathcal{D}^*$ with $\mathrm{err}(\mathcal{D}^*,\mathcal{Z}) \le \alpha + \delta/2$ if $\alpha \ge \alpha^*$ and returns *No* if $\alpha < \alpha^*$.

We do not know the value of α^*, so we perform a binary search on the value of α. In particular, let $E = \{\frac{\delta}{2}, (1+\frac{\delta}{2})\frac{\delta}{2}, (1+\frac{\delta}{2})^2\frac{\delta}{2}, \ldots, 1\}$ be a discretization of the range $[0,1]$, and let α_j be the j-th value of E. Suppose the binary search currently guesses α_i to be the current guess of α^*. If IsFeasible$(\mathcal{Z},\delta,\alpha_i)$ returns a distribution $\mathcal{D}$, we continue the binary search for values less than α_i in E. Otherwise, we continue the binary search for values greater than α_i in E. At the end of the binary search, we return the last distribution $\mathcal{D}^*$ that IsFeasible found. Assuming the correctness of the decision procedure, the algorithm returns the desired distribution in $\log\frac{1}{\delta}$ iterations. If $\alpha^* \le \delta/2$, then IsFeasible$(\mathcal{Z},\delta,\alpha_1)$ returns a distribution $\mathcal{D}^*$ such that $\mathrm{err}(\mathcal{D}^*,\mathcal{Z}) \le \alpha_1 + \delta/2 = \delta \le \alpha^* + \delta$. In any other case, without loss of generality assuming that $\alpha_{i-1} < \alpha^* \le \alpha_i$, we have $\alpha_i \le (1+\delta/2)\alpha^*$. By definition, IsFeasible$(\mathcal{Z},\delta,\alpha_i)$ returns a distribution $\mathcal{D}_i$ such that $\mathrm{err}(\mathcal{D}_i,\mathcal{Z}) \le \alpha_i + \delta/2$. Hence,

$$\mathrm{err}(\mathcal{D}^*,\mathcal{Z}) \le \mathrm{err}(\mathcal{D}_i,\mathcal{Z}) \le \alpha_i + \delta/2 \le (1+\delta/2)\alpha^* + \delta/2 \le \alpha^* + \delta.$$

We now describe the decision procedure IsFeasible.

2.1 Decision procedure

Let $\mathcal{Z}, \delta$, and α be as defined above. The decision problem can be formulated as follows:

(FP1) $\exists?\ \mathcal{D} \in \mathbb{D}$ s.t.

$$\frac{1}{n}\sum_{i=1}^{n} u_i \le \alpha$$

$$|s_{\mathcal{D}}(R_i) - s_i| \le u_i \qquad \text{for } i = 1 \ldots n$$

$$u_i \in [0,1] \quad \text{for } i = 1 \ldots n.$$

Here u_i models the error in the selectivity of R_i. A challenge in solving the above decision problem is determining the candidate set of points in $\mathrm{supp}(\mathcal{D})$. The problem as stated is infinite-dimensional. Our next lemma suggests how to reduce it to a finite-dimensional problem by constructing a finite set of candidate points for $\mathrm{supp}(\mathcal{D})$.

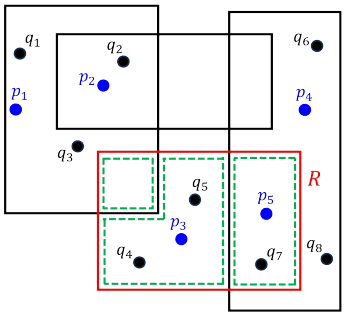

Figure 1 The (black) points q_i represent points from the underlying data distribution $\mathcal{D}$ while (blue) points $p_\tau \in \mathcal{P}$. The (green) dashed segments show three cells of the arrangement in rectangle R. The weights of points in $\mathcal{P}$ are: $w(p_1) = w_1 + w_3$, $w(p_2) = w_2$, $w(p_3) = w_4 + w_5$, $w(p_4) = w_6 + w_8$, $w(p_5) = w_7$.

The *arrangement* of $\mathcal{R}$, denoted $\mathcal{A}(\mathcal{R})$, is a partitioning of $\mathbb{R}^d$ into contiguous regions called *cells* such that for every (d-dimensional) cell τ in the arrangement, τ lies in the same subset of $\mathcal{R}$. For each d-dimensional cell $\tau \in \mathcal{A}(\mathcal{R})$, we choose an arbitrary point p_τ in the interior of τ. Let $\mathcal{P} = \{p_\tau \mid \tau \in \mathcal{A}(\mathcal{R})\}$ be the set of candidate points. Note that $\mathcal{A}(\mathcal{R})$ has $O(n^d)$ cells, and it can be computed in $O(n^d \log n)$ time [3]. Therefore, $|\mathcal{P}| = O(n^d)$ and $\mathcal{P}$ can be computed in $O(n^d \log n)$ time.

▶ **Lemma 1.** *For any discrete distribution $\mathcal{D} \in \mathbb{D}$, there is another discrete distribution $\mathcal{D}'$ such that* $\text{supp}(\mathcal{D}') \subseteq \mathcal{P}$ *and* $\text{err}(\mathcal{D}, \mathcal{Z}) = \text{err}(\mathcal{D}', \mathcal{Z})$.

Proof. For each d-dimensional cell $\tau \in \mathcal{A}(\mathcal{R})$, let

$$w_\tau = \sum_{p_i \in \text{supp}(\mathcal{D})} \mathbb{1}(p_i \in \tau) w_i$$

be the total weight of points of $\text{supp}(\mathcal{D})$ that lie in τ. We define

$$\mathcal{D}' = \{(p_\tau, w_\tau) \mid \tau \in \mathcal{A}(\mathcal{R}), w_\tau > 0\}.$$

See also Figure 1. By definition, for any rectangle $R \in \mathcal{R}$, the cells of $\mathcal{A}(\mathcal{R})$ lying inside R induce a partitioning of R. Let $\mathcal{A}(\mathcal{R} \mid R)$ denote this partitioning of R into cells. Then

$$\begin{aligned} s_{\mathcal{D}'}(R) &= \sum_{p_\tau \in \mathcal{P}} \mathbb{1}(p_\tau \in R) w_\tau = \sum_{\tau \in \mathcal{A}(\mathcal{R}|R)} w_\tau = \sum_{\tau \in \mathcal{A}(\mathcal{R}|R)} \sum_{p_i \in \tau \cap \text{supp}(\mathcal{D})} w_i \\ &= \sum_{p_i \in \text{supp}(\mathcal{D})} \mathbb{1}(p_i \in R) w_i = s_{\mathcal{D}}(R). \end{aligned}$$

Hence, $\text{err}(\mathcal{D}, \mathcal{Z}) = \text{err}(\mathcal{D}', \mathcal{Z})$. ◀

▶ Remark. We note that the choice of point p_τ in each cell $\tau \in \mathcal{A}(\mathcal{R})$ is arbitrary; one can use any point in τ as its representative point.

In view of Lemma 1, it suffices to restrict $\text{supp}(\mathcal{D})$ to be a subset of $\mathcal{P}$. We order the cells of $\mathcal{A}(\mathcal{R})$ arbitrarily and let p_j denote the point chosen from the j-th cell, so $\mathcal{P} = \{p_1, \ldots, p_m\}$ where $m = |P|$. We introduce a real variable $v_j \in [0, 1]$ that models the weight of $p_j \in \mathcal{P}$. Then (FP1) can be rewritten as

$$\text{(FP2)} \qquad \frac{1}{n}\sum_{i=1}^{n} u_i \le \alpha$$

$$\left|\sum_{j:p_j\in R_i} v_j - s_i\right| \le u_i \qquad \text{for } i = 1\dots n \tag{2}$$

$$\sum_{j=1}^{m} v_j \le 1 \tag{3}$$

$$u_i, v_j \in [0,1] \quad \text{for } i = 1\dots n,\ j = 1\dots m$$

We require that the sum of weights of the discrete distribution we compute is at most 1 because, for any distribution returned with $\sum_j^m v_j < 1$, we can add an arbitrary point that is not contained in any range with weight $1 - \sum_j^m v_j$.

The above decision problem can be written as a linear program by replacing (2) with two linear inequalities.

$$\text{(LP1)} \quad -\frac{1}{n}\sum_{i=1}^{n} u_i \ge -\alpha \tag{4}$$

$$u_i - \sum_{j:p_j\in R_i} v_j \ge -s_i \qquad \text{for } 1 \le i \le n \tag{5}$$

$$u_i + \sum_{j:p_j\in R_i} v_j \ge s_i \qquad \text{for } 1 \le i \le n \tag{6}$$

$$\sum_{j=1}^{m} v_j \le 1 \tag{7}$$

$$u_i, v_j \in [0,1] \quad \text{for } i = 1\dots n,\ j = 1\dots m$$

It will be convenient to write the above LP in a compact form. Let $\mathsf{u} = (u_1, \dots, u_n)$, $\mathsf{v} = (v_1, \dots, v_m)$, and $\mathsf{x} = (\mathsf{u}, \mathsf{v})$. Set

$$\mathbb{X} = \{\mathsf{x} = (\mathsf{u}, \mathsf{v}) \in [0,1]^{n+m} \mid \|\mathsf{v}\|_1 \le 1\}.$$

For $0 \le \kappa \le 2n$, let $\mathsf{A}_\kappa\mathsf{x} \ge \mathsf{b}_\kappa$ denote the κ-th constraint of (4), (5), (6). Namely, $\mathsf{A}_0\mathsf{x} \ge \mathsf{b}_0$ denotes (4), and for $1 \le i \le n$, $\mathsf{A}_{2i-1}\mathsf{x} \ge \mathsf{b}_{2i-1}$ and $\mathsf{A}_{2i}\mathsf{x} \ge \mathsf{b}_{2i}$ denote the constraints (5) and (6), respectively, for the rectangle R_i. Then (LP1) asks whether there exists an $\mathsf{x} \in \mathbb{X}$ such that $\mathsf{A}\mathsf{x} \ge \mathsf{b}$.

We use a Multiplicative-Weight-Update (MWU) method to solve this linear program, following the general approach described by Arora et al. [5], though the exact implementation depends on the specific LP; see also [42]. We describe how this approach is implemented in our setting. We will need this algorithm for the faster implementation described in Section 3.

2.2 MWU algorithm

We describe an algorithm that either returns a $\tilde{\mathsf{x}} \in \mathbb{X}$ such that $\mathsf{A}\tilde{\mathsf{x}} \ge \mathsf{b} - \delta/4$ or returns that there is no feasible solution for (LP1). We set two parameters $\eta = \frac{\delta}{c_1}$ and $T = \lceil c_2\delta^{-2}\ln(2n+1)\rceil$, where $c_1, c_2 > 0$ are sufficiently large constants to be chosen later. The algorithm works in T rounds. At the beginning of round t, it has a $(2n+1)$-dimensional probability vector $\mathsf{w}^{(t)} = (w_0^{(t)}, \dots, w_{2n}^{(t)})$. Initially, $\mathsf{w}^{(1)} = (\frac{1}{2n+1}, \dots, \frac{1}{2n+1})$. In the t-th round, the algorithm solves the decision problem consisting of one constraint

$$\mathsf{w}^{(t)\top}\mathsf{A}\mathsf{x} \ge \mathsf{w}^{(t)\top}\mathsf{b}, \quad \mathsf{x} \in \mathbb{X} \tag{8}$$

which we refer to as the *expected constraint*. The algorithm computes, as described below, $\mathsf{x}^{(t)} = \arg\max_{x\in\mathbb{X}} \mathsf{w}^{(t)\top}\mathsf{A}\mathsf{x}$.

If $\mathsf{w}^{(t)\top}\mathsf{A}\mathsf{x}^{(t)} < \mathsf{w}^{(t)\top}\mathsf{b}$, we conclude that (LP1) is infeasible because, by definition, any feasible solution of (LP1) satisfies (8), and we return *No*. Otherwise, for $i = 0 \dots 2n$, we set

$$w_i^{(t+1)} = w_i^{(t)} \frac{1 - \eta(\mathsf{A}_i\mathsf{x}^{(t)} - \mathsf{b}_i)}{\mu^{(t+1)}} \tag{9}$$

where $\mu^{(t+1)}$ is a normalization factor so that $\|\mathsf{w}^{(t+1)}\|_1 = 1$.

If the algorithm does not return *No* within the first T rounds, then after completing T rounds, it returns $\tilde{\mathsf{x}} = \frac{1}{T}\sum_{i=1}^{T} \mathsf{x}^{(t)}$. Suppose $\tilde{\mathsf{x}} = (\tilde{\mathsf{u}}, \tilde{\mathsf{v}})$. We return the distribution $\tilde{D} = \{(p_j, \tilde{v}_j) \mid \tilde{v}_j > 0\}$.

Computing $\mathsf{x}^{(t)}$. We now describe the computation of $\mathsf{x}^{(t)} = (u_1^{(t)}, \dots, u_n^{(t)}, v_1^{(t)}, \dots, v_m^{(t)})$ at each step. We can express the LHS of the expected constraint (8) as

$$\mathsf{w}^{(t)\top}\mathsf{A}\mathsf{x} = \sum_{i=1}^{n} \varphi_i^{(t)} u_i + \sum_{j=1}^{m} \psi_j^{(t)} v_j \tag{10}$$

$$\text{where} \quad \varphi_i^{(t)} = w_{2i}^{(t)} + w_{2i-1}^{(t)} - \frac{w_0^{(t)}}{n} \quad \text{for } 1 \le i \le n, \tag{11}$$

$$\psi_j^{(t)} = \sum_{i:p_j \in R_i} w_{2i}^{(t)} - w_{2i-1}^{(t)} \quad \text{for } 1 \le j \le m. \tag{12}$$

Note that the two terms in (10) do not share variables and (7) does not involve u_i's, so we can maximize each of them independently. Recall that $u_i \in [0, 1]$, so to maximize (10), we choose for $1 \le i \le n$,

$$u_i^{(t)} = \begin{cases} 1 & \text{if } \varphi_i^{(t)} > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{13}$$

Next, we choose $v_j^{(t)}$ as follows. Let $j^{(t)} = \arg\max_{1 \le j \le m} \psi_j^{(t)}$. Since $\mathsf{v} \in [0, 1]^m$ and $\|\mathsf{v}\|_1 \le 1$ for $(\mathsf{u}, \mathsf{v}) \in \mathbb{X}$, we choose for $1 \le j \le m$,

$$v_j^{(t)} = \begin{cases} 1 & \text{if } j = j^{(t)} \text{ and } \psi_j^{(t)} > 0, \\ 0 & \text{otherwise.} \end{cases} \tag{14}$$

Note that although many u_i's could be set to 1, at most one v_j is set to 1. This property will be crucial for our faster implementation in the next section. Since at most T v_j's are non-zero, $|\tilde{\mathcal{D}}| \le T = O(\delta^{-2} \log n)$.

This completes the description of our basic algorithm. We now analyze its running time and correctness.

2.3 Analysis

▶ **Lemma 2.** *Assume that the MWU algorithm returns a solution $\tilde{\mathsf{x}}$ after T rounds. Then $\mathsf{A}_i\tilde{\mathsf{x}} \ge \mathsf{b}_i - \delta/4$, for every $0 \le i \le 2n$.*

Proof. It is straightforward to verify that for any round $1 \le t \le T$, $|\mathsf{A}_i\mathsf{x}^{(t)} - \mathsf{b}_i| \le 2$. Let $\mathcal{T}_i^- = \{t \le T \mid \mathsf{A}_i\mathsf{x}^{(t)} - \mathsf{b}_i < 0\}$ be the subset of rounds where $\mathsf{A}_i\mathsf{x}^{(t)} - \mathsf{b}_i < 0$. Using the analysis in Arora et al. [5] (see the proof of Theorem 3.3), for every $0 \le i \le 2n$, we obtain,

$$0 \le \sum_{t=1}^{T} \frac{1}{2}(\mathsf{A}_i\mathsf{x}^{(t)} - \mathsf{b}_i) + \eta \sum_{t=1}^{T} \frac{1}{2}|\mathsf{A}_i\mathsf{x}^{(t)} - \mathsf{b}_i| + \frac{\ln(2n+1)}{\eta}$$
$$= (1+\eta)\sum_{t=1}^{T} \frac{1}{2}(\mathsf{A}_i\mathsf{x}^{(t)} - \mathsf{b}_i) + 2\eta \sum_{t \in \mathcal{T}_i^-} \frac{1}{2}|\mathsf{A}_i\mathsf{x}^{(t)} - \mathsf{b}_i| + \frac{\ln(2n+1)}{\eta}$$
$$\le (1+\eta)\sum_{t=1}^{T} \frac{1}{2}(\mathsf{A}_i\mathsf{x}^{(t)} - \mathsf{b}_i) + 2\eta T + \frac{\ln(2n+1)}{\eta}.$$

The last inequality follows because $|\mathsf{A}_i\mathsf{x}^{(t)} - \mathsf{b}_i| \le 2$. Noting that $\tilde{\mathsf{x}} = \frac{1}{T}\sum_{i=1}^{T} \mathsf{x}^{(t)}$, we get $0 \le (1+\eta)(\mathsf{A}_i\tilde{\mathsf{x}} - \mathsf{b}_i) + 4\eta + \frac{2\ln(2n+1)}{\eta T}$. By choosing $c_1 = 32$ and $c_2 = 512$, we obtain $\eta = \frac{\delta}{32}$ and $T = \lceil 512\delta^{-2}\ln(2n+1)\rceil$. Therefore

$$(1+\eta)(\mathsf{A}_i\tilde{\mathsf{x}} - \mathsf{b}_i) + \delta/4 \ge 0 \Leftrightarrow \mathsf{A}_i\tilde{\mathsf{x}} \ge \mathsf{b}_i - \delta/4. \quad \blacktriangleleft$$

▶ **Lemma 3.** *Given a training set $\mathcal{Z}$ and parameters α, δ, the algorithm* IsFeasible *either returns a discrete distribution $\tilde{\mathcal{D}}$ of size $O(\delta^{-2}\log n)$ such that $\mathrm{err}(\tilde{\mathcal{D}}, \mathcal{Z}) \le \alpha + \delta/2$, or returns* No *for $\alpha < \alpha^*$.*

Proof. First, by definition, our algorithm always solves the expected constraint optimally. Hence, if the algorithm stops in iteration $t \le T$ (because we cannot satisfy the expected constraint $\mathsf{w}^{(t)\top}\mathsf{A}\mathsf{x} \ge \mathsf{w}^{(t)\top}\mathsf{b}$ for $\mathsf{x} \in \mathbb{X}$) then it is also true that there is no $\mathsf{x} \in \mathbb{X}$ such that $\mathsf{A}\mathsf{x} \ge \mathsf{b}$. So the algorithm correctly returns that the LP is infeasible and $\alpha < \alpha^*$.

Next, assume that the algorithm returns $\tilde{\mathcal{D}}$. Recall that $\tilde{\mathsf{x}} = (\tilde{\mathsf{u}}, \tilde{\mathsf{v}}) = (\tilde{u}_1, \ldots, \tilde{u}_n, \tilde{v}_1, \ldots, \tilde{v}_n) \in \mathbb{X}$. Since $\mathbb{X}$ is convex and $\mathsf{x}^{(t)} \in \mathbb{X}$ for each $1 \le t \le T$, it also holds that $\tilde{\mathsf{x}} \in \mathbb{X}$ and $\|\tilde{\mathsf{v}}\|_1 \le 1$. Hence, it follows that $\tilde{\mathcal{D}}$ is indeed a distribution of size at most $T = O(\delta^{-2}\log n)$. From Lemma 2 and the equivalence of (FP2) and (LP1), we get

$$\frac{1}{n}\sum_{i=1}^{n} \tilde{u}_i \le \alpha + \delta/4 \quad \text{and} \quad \left|\sum_{j:p_j \in R_i} \tilde{v}_j - s_i\right| \le u_i + \delta/4$$

for $i = 1 \ldots n$. Recall that $s_{\tilde{\mathcal{D}}}(R_i) = \sum_j \blacksquare(p_j \in R_i)\tilde{v}_j$. Hence

$$\mathrm{err}(\tilde{\mathcal{D}}, \mathcal{Z}) = \frac{1}{n}\sum_{i=1}^{n} |s_{\tilde{\mathcal{D}}}(R_i) - s_i| \le \frac{1}{n}\sum_{i=1}^{n} \tilde{u}_i + \delta/4 \le \alpha + \delta/4 + \delta/4 = \alpha + \delta/2. \quad \blacktriangleleft$$

As for the running time, the binary search executes $O(\log \frac{1}{\delta})$ iterations, each of which runs the IsFeasible algorithm. We need $O(m\log n) = O(n^d \log n)$ time to construct (LP1). The IsFeasible algorithm runs for $T = O(\delta^{-2}\log n)$ rounds. In each round t, we need $O(n)$ time to compute the values of $\mathsf{u}^{(t)}$ by (11) and (13). We need $O(m\log n) = O(n^d \log n)$ time to find all the values $\psi_j^{(t)}$ by (12). Putting everything together, we have the following lemma.

▶ **Lemma 4.** *The algorithm runs in $O(\delta^{-2} n^d \log^2 n \log \delta^{-1})$ time.*

3 The Improved Algorithm

We present an implementation of a small variant of the algorithm in the last section that computes the desired distribution with high probability in $n(\delta^{-1}\log n)^{O(1)}$ time (see Theorem 9 below for a more precise characterization). There are two challenges in implementing

the IsFeasible procedure efficiently. First, (LP1) has $O(n^d)$ variables – although u has dimension n, v has dimension $m = O(n^d)$ – so we cannot afford to represent (LP1) explicitly. Second, in each round t, computing $\mathsf{u}^{(t)}$ is easy, but computing $\mathsf{v}^{(t)}$ quickly seems challenging even though only one of the values in v is non-zero. We exploit underlying geometry to address both challenges. We first describe how to implement IsFeasible without writing the LP explicitly and then describe how to compute $\mathsf{x}^{(t)}$ quickly.

Implicit representation of LP. We maintain the probability vector $\mathsf{w}^{(t)}$ as before. We also maintain a multi-set $\mathcal{F} \subset \mathbb{R}^d$ of points. Initially, $\mathcal{F} = \emptyset$. Each round adds one point to $\mathcal{F}$, so $|\mathcal{F}| \leq T = O(\delta^{-2} \log n)$. Since we have $\mathsf{w}^{(t)}$ at our disposal, we can compute $u_i^{(t)}$, for $1 \leq i \leq n$, using (11) and (13) as before. The main question is how to compute $j^{(t)} = \arg\max_{1 \leq j \leq m} \psi_j^{(t)}$ without maintaining all the variables explicitly. For each rectangle $R_i \in \mathcal{R}$, let $\omega^{(t)}(R_i) = w_{2i}^{(t)} - w_{2i-1}^{(t)}$. For each point $x \in \mathbb{R}^d$, we define its *depth* with respect to $(\mathcal{R}, \omega^{(t)})$, denoted by $\Delta(x)$, to be

$$\Delta(x) = \sum_{R \in \mathcal{R}} \mathbb{1}(x \in R)\omega^{(t)}(R).$$

It is easily checked that the depth of all points lying in the same cell of $\mathcal{A}(\mathcal{R})$ is the same, and that many cells may have the same depth. By the definition of the depth, $\psi_j^{(t)} = \Delta(p_j)$. Thus the problem of computing $j^{(t)}$ is equivalent to choosing a point $p^{(t)}$ in $\mathbb{R}^d$ of the maximum depth, i.e., $\Delta(p^{(t)}) = \max_{p \in \mathbb{R}^d} \Delta(p)$. Chan [9] has described an $O(n^{d/2})$ time algorithm to compute a point of maximum depth in a set of n weighted rectangles in $\mathbb{R}^d$. We refer to this algorithm as DeepestPt$(\mathcal{R}, \omega)$, where $\mathcal{R}$ is a set of n rectangles in $\mathbb{R}^d$ and $\omega \in \mathbb{R}^n$ is the weight vector.

We thus proceed in the t-th round of IsFeasible as follows. First, compute the vector $\mathsf{u}^{(t)}$ as before. Next, we compute the deepest point $p^{(t)}$ by calling DeepestPt$(\mathcal{R}, \omega^{(t)})$. If $\Delta(p^{(t)}) > 0$, then we add $p^{(t)}$ to the set F; otherwise we ignore it. Intuitively, this is equivalent to setting $v_{j^{(t)}} = 1$ if $\psi_{j^{(t)}} > 0$ and 0 otherwise. We do not know how to compute $j^{(t)}$ efficiently and thus cannot compute which v_j needs to be set to 1. But what comes to our rescue is that we only need to compute $w_q^{(t)} \mathsf{A}_q \mathsf{x}$, for all $0 \leq q \leq 2n$, to check whether the expected constraint holds and to update the weight factors. Fortunately, we can accomplish this using only u and $p^{(t)}$, without an explicit representation of x, as follows. We set $w_{2i-1}^{(t)} \mathsf{A}_{2i-1} \mathsf{x} = w_{2i-1}^{(t)} u_i^{(t)}$ and $w_{2i}^{(t)} \mathsf{A}_{2i} \mathsf{x} = \mathsf{w}_{2i}^{(t)} u_i^{(t)}$ if $\Delta(p^{(t)}) \leq 0$ or $p^{(t)} \notin R_i$, and we set $w_{2i-1}^{(t)} \mathsf{A}_{2i-1} \mathsf{x} = w_{2i-1}^{(t)} (u_i^{(t)} - 1)$ and $w_{2i}^{(t)} \mathsf{A}_{2i} \mathsf{x} = w_{2i}^{(t)} (u_i^{(t)} + 1)$ if $\Delta(p^{(t)}) > 0$ and $p^{(t)} \in R_i$. It can be checked that these values are the same as when we set $v_{j^{(t)}}$ as above.

After having computed $w_q^{(t)} \mathsf{A}_q \mathsf{x}$ for all $0 \leq q \leq 2n$, we check whether $\mathsf{w}^{(t)\top} \mathsf{A} \mathsf{x} < \mathsf{w}^{(t)\top} \mathsf{b}$. If so, we stop and return *No*. Otherwise, we compute $\mathsf{w}^{(t+1)}$ using (9) as before. If the algorithm is not aborted within the first T rounds, we return $\hat{\mathcal{D}} = \{(p, \frac{1}{T}) \mid p \in \mathcal{F}\}$. Recall that $\mathcal{F}$ is a multi-set. If there are s copies of a point p, we keep only one copy of p and set its weight to $\frac{s}{T}$. The following lemma establishes the correctness of the algorithm.

▶ **Lemma 5.** *Given $\mathcal{Z}$ and $\delta \in [0, 1]$, $\mathrm{err}(\tilde{\mathcal{D}}, \mathcal{Z}) = \mathrm{err}(\hat{\mathcal{D}}, \mathcal{Z})$.*

Proof. For simplicity, we assume that for any $\omega^{(t)}$, $1 \leq t \leq T$, the maximum-depth cell in $\mathcal{A}(\mathcal{R})$ with respect to $\omega^{(t)}$ is unique and that $j^{(t)}$ are distinct for each $t \leq T$. Then by the definition of depth, $p^{(t)}$ and $p_{j^{(t)}}$ lie in the same cell of $\mathcal{A}(\mathcal{R})$, so the value of $\mathsf{w}^{(t)\top} \mathsf{A} \mathsf{x}$ computed by the implicit algorithm is the same as $\mathsf{w}^{(t)\top} \mathsf{A} \mathsf{x}^{(t)}$ computed by the basic algorithm. Hence, assuming $\mathsf{w}^{(t)}$ computed by the two algorithms is the same, $\mathsf{w}^{(t+1)}$ computed by them is also the same. Recall that $v_{j^{(t)}}^{(t)}$ is set to 1 if and only if $\psi_{j^{(t)}}^{(t)} > 0$, which is the same

condition when $p^{(t)}$ is added to $\mathcal{F}$ by the implicit algorithm. $v^{(t)}_{j^{(t)}} = 1$ if and only if $p^{(t)} \in F$. Hence, $p_{j^{(t)}} \in \mathrm{supp}(\tilde{\mathcal{D}})$ if and only if $p^{(t)} \in \mathrm{supp}(\hat{\mathcal{D}})$ and $p_{j^{(t)}}$ and $p^{(t)}$ lie in the same cell of $\mathcal{A}(\mathcal{R})$. The same argument as in Lemma 1 now implies that $\mathrm{err}(\tilde{\mathcal{D}}, \mathcal{Z}) = \mathrm{err}(\hat{\mathcal{D}}, \mathcal{Z})$. ◀

Fast computation of $p^{(t)}$. The main observation is that it is not crucial to compute $\mathsf{x}^{(t)} = \arg\max_{\mathsf{x}\in\mathbb{X}} \mathsf{w}^{(t)\top}\mathsf{A}\mathsf{x}$ in the t-th round of the MWU algorithm. Instead, it suffices to compute $\tilde{\mathsf{x}}^{(t)}$ such that $\mathsf{w}^{(t)\top}\mathsf{A}\tilde{\mathsf{x}}^{(t)} \geq \mathsf{w}^{(t)\top}\mathsf{A}\mathsf{x}^{(t)} - \delta/12$. In the terminology of the implicit LP algorithm, this is equivalent to saying that it suffices to compute a point $\tilde{p}^{(t)}$ with $\Delta(\tilde{p}^{(t)}) \geq \max_{p\in\mathbb{R}^d} \Delta(p) - \delta/12$. We use ε-approximations and random sampling [23, 11] to compute $\tilde{p}^{(t)}$ quickly. The notion of ε-approximation is defined for general range spaces but we define ε-approximation in our setting.

Given a set $\mathcal{R}$ of rectangles and a weight function $\omega : \mathcal{R} \to [0,1]$ a multi-subset $\mathcal{N} \subset \mathcal{R}$ is called an *ε-approximation* of $(\mathcal{R}, \omega)$ if for any point $x \in \mathbb{R}^d$,

$$\left| \frac{\Delta(x, \mathcal{R}, \omega)}{\omega(\mathcal{R})} - \frac{\Delta(x, \mathcal{N})}{|\mathcal{N}|} \right| \leq \varepsilon, \tag{15}$$

where $\Delta(x, \mathcal{R}, \omega)$ is the weighted depth of point x in the set of rectangles $\mathcal{R}$ with weights ω, and $\Delta(x, \mathcal{N})$ is the depth of point x in the set of rectangles $\mathcal{N}$ assuming that the weight of each rectangle in $\mathcal{N}$ is 1.

Let A be a random (multi-)subset of $\mathcal{R}$ of size $O(\varepsilon^{-2} \ln \phi^{-1})$ where at each step each rectangle of $\mathcal{R}$ is chosen with probability proportional to its weight (with repetition). It is well known [23, 11] that A is an ε-approximation with probability at least $1 - \phi$. With this result at our disposal, we compute $\tilde{p}^{(t)}$, an approximately deepest point with respect to $(\mathcal{R}, \mathsf{w}^{(t)})$, as follows. Let $\mathsf{w}^{(t)}$ be as defined above. Let $\mathsf{w}^+ = (w^+_1, \ldots, w^+_n)$ and $\mathsf{w}^- = (w^-_1, \ldots, w^-_n)$ be two vectors such that $w^+_i = w^{(t)}_{2i}$ and $w^-_i = w^{(t)}_{2i-1}$ for every $1 \leq i \leq n$. We set $r = c_3\delta^{-2}$, where $c_3 > 0$ is a sufficiently large constant. We repeat the following for $\mu = O(\log n)$ times: For each $h \leq \mu$, we choose a random sample $\mathcal{N}^+_h$ of $(\mathcal{R}, \mathsf{w}^+)$ of size r and another random sample $\mathcal{N}^-_h$ of $(\mathcal{R}, \mathsf{w}^-)$ of size r. Let $\mathcal{N}_h = \mathcal{N}^+_h \cup \mathcal{N}^-_h$. We define the weight function $\bar{\omega}_h : \mathcal{N}_h \to \mathbb{R}$ as

$$\bar{\omega}_h(R) = \begin{cases} \frac{\|\mathsf{w}^+\|_1}{r} & \text{if } R \in \mathcal{N}^+_h, \\ -\frac{\|\mathsf{w}^-\|_1}{r} & \text{if } R \in \mathcal{N}^-_h. \end{cases} \tag{16}$$

We compute a deepest point $\tilde{p}^{(t)}_h$ with respect to $(\mathcal{N}_h, \bar{\omega}_h)$ along with $\Delta(\tilde{p}^{(t)}_h, \mathcal{N}_h, \bar{\omega}_h)$ by calling DeepestPt$(\mathcal{N}_h, \bar{\omega}_h)$. After repeating μ times, we choose as $\tilde{p}^{(t)}$ the point $\tilde{p}^{(t)}_\xi$ with the median $\Delta(\tilde{p}^{(t)}_\xi, \mathcal{N}_\xi, \bar{\omega}_\xi)$ among depths $\{\Delta(\tilde{p}^{(t)}_1, \mathcal{N}_1, \bar{\omega}_1), \ldots, \Delta(\tilde{p}^{(t)}_\mu, \mathcal{N}_\mu, \bar{\omega}_\mu)\}$, where $\xi \in [1, \mu]$. Recall that $\omega^{(t)}_i = w^{(t)}_{2i} - w^{(t)}_{2i-1}$.

▶ **Lemma 6.** *$\Delta(\tilde{p}^{(t)}, \mathcal{R}, \omega^{(t)}) \geq \max_{x\in\mathbb{R}^d} \Delta(x, \mathcal{R}, \omega^{(t)}) - \delta/12$ with probability at least $1 - 1/n^{O(1)}$.*

Proof. If we choose the constant c_3 sufficiently large, then both $\mathcal{N}^+_h$ and $\mathcal{N}^-_h$ are $\frac{\delta}{48}$-approximations with probability greater than $1/2$. Therefore, for any point $x \in \mathbb{R}^d$,

$$\left| \frac{\Delta(x, \mathcal{R}, \mathsf{w}^+)}{\|\mathsf{w}^+\|_1} - \frac{\Delta(x, \mathcal{N}^+_h)}{|\mathcal{N}^+_h|} \right|, \left| \frac{\Delta(x, \mathcal{R}, \mathsf{w}^-)}{\|\mathsf{w}^-\|_1} - \frac{\Delta(x, \mathcal{N}^-_h)}{|\mathcal{N}^-_h|} \right| \leq \frac{\delta}{48}. \tag{17}$$

Using (17) and the fact that $\|\mathsf{w}^+\|_1, \|\mathsf{w}^-\|_1 \leq 1$, we obtain

$$\begin{aligned} \Delta(x, \mathcal{R}, \omega^{(t)}) &= \Delta(x, \mathcal{R}, \mathsf{w}^+) - \Delta(x, \mathcal{R}, \mathsf{w}^-) \leq \frac{\|\mathsf{w}^+\|_1}{r}\Delta(x, \mathcal{N}^+_h) - \frac{\|\mathsf{w}^-\|_1}{r}\Delta(x, \mathcal{N}^-_h) + 2\frac{\delta}{48} \\ &= \Delta(x, \mathcal{N}_h, \bar{\omega}_h) + \frac{\delta}{24}. \end{aligned}$$

Similarly, $\Delta(x, \mathcal{R}, \omega^{(t)}) \geq \Delta(x, \mathcal{N}_h, \bar{\omega}_h) - \frac{\delta}{24}$. Next, we define $x^* = \arg\max_{x \in \mathbb{R}^d} \Delta(x, \mathcal{R}, \omega^{(t)})$. Hence, with probability greater than 1/2,

$$\Delta(\tilde{p}_h^{(t)}, \mathcal{R}, \omega^{(t)}) \geq \Delta(x^*, \mathcal{R}, \omega^{(t)}) - \frac{\delta}{12}.$$

Using the well known *median trick* (an application of the median trick can be found in [29]), we know that $\tilde{p}^{(t)} = \tilde{p}_\xi^{(t)}$ that has the median depth satisfies

$$\Delta(\tilde{p}^{(t)}, \mathcal{R}, \omega^{(t)}) \geq \Delta(x^*, \mathcal{R}, \omega^{(t)}) - \frac{\delta}{12},$$

with probability at least $1 - 1/n^{O(1)}$. ◀

▶ **Lemma 7.** *With probability at least $1 - 1/n^{O(1)}$, the* IsFeasible *decision procedure either returns a discrete distribution $\tilde{\mathcal{D}}$ of size $O(\delta^{-2} \log n)$ such that $\mathrm{err}(\tilde{\mathcal{D}}, \mathcal{Z}) \leq \alpha + \delta/2$ or returns* No *for $\alpha < \alpha^*$.*

Proof. Using the proofs of Lemma 2 and Lemma 3 (by slightly increasing the constants c_1, c_2) we get that with probability at least $1 - 1/n^{O(1)}$, if the algorithm does not abort in the first T iterations in the end it finds $\tilde{\mathsf{x}} \in \mathbb{X}$ such that $\mathsf{A}\tilde{\mathsf{x}} \geq \mathsf{b} - 3\delta/12 = \mathsf{b} - \delta/4$. Using Lemma 5 and Lemma 3, we conclude the result. ◀

▶ **Lemma 8.** *The improved algorithm runs in time*

$$O((n + \delta^{-2} \log^2 n + \delta^{-d} \log n)\delta^{-2} \log n \log \delta^{-1}).$$

Proof. In each round, we spend $O(n)$ time to compute $\mathsf{u}^{(t)}$. For each h, we construct two ε-approximations $\mathcal{N}_h^-, \mathcal{N}_h^+$ of size $O(\delta^{-2})$ by applying weighted random sampling. Using a binary search tree constructed at the beginning of each round of the MWU algorithm, we get each sample in $O(\log n)$ time. Hence, we construct $\mathcal{N}_h$ in $O(\delta^{-2} \log n)$. We spend $O((\delta^{-2})^{d/2})$ time to compute $\tilde{p}_h^{(t)}$ using [9]. Overall we spend $O((\delta^{-2} \log n + \delta^{-d}) \log n)$ time to compute the point $\tilde{p}^{(t)}$. Summing these bounds over all iterations of the binary search and the MWU method, we have the desired bound. ◀

The algorithm we proposed computes a discrete distribution $\tilde{\mathcal{D}}$ of size $O(\delta^{-2} \log n)$. The size can be reduced to $O(\delta^{-2})$ as follows. We first run the algorithm above with $\delta \leftarrow \delta/2$. After obtaining $\tilde{\mathcal{D}}$, we repeat the following procedure $O(\log n)$ times. In the h-th iteration, we get a $\tilde{\mathcal{N}}_h$ of $O(\delta^{-2})$ weighted random samples from $\tilde{\mathcal{D}}$. This is a $\delta/2$-approximation with probability at least 1/2. Let $\tilde{\mathcal{D}}_h$ be the distribution of size $O(\delta^{-2})$ defined by $\tilde{\mathcal{N}}_h$. In the end, we return the best $\delta/2$-distribution $\tilde{\mathcal{D}}_h$ with respect to $\mathrm{err}(\tilde{\mathcal{D}}_h, \mathcal{Z})$. With probability at least $1 - 1/n^{O(1)}$, we find a distribution of size $O(\delta^{-2})$ and additive error δ. The running time of the additional steps is dominated by the running time in Lemma 8.

▶ **Theorem 9.** *Let $\mathcal{Z} = \{z_1, \ldots, z_n\}$ be a set of training samples such that $z_i = (R_i, s_i)$, where R_i is an axis aligned rectangle in $\mathbb{R}^d$ and $s_i \in [0, 1]$ is its selectivity. Given a parameter $\delta \in (0, 1)$, a discrete distribution $\tilde{\mathcal{D}}$ of size $O(\delta^{-2})$ can be computed in $O((n + \delta^{-2} \log^2 n + \delta^{-d} \log n)\delta^{-2} \log n \log \delta^{-1})$ time such that $\mathrm{err}_1(\mathcal{Z}, \tilde{\mathcal{D}}) \leq \alpha_1^*(\mathcal{Z}) + \delta$, with probability at least $1 - 1/n^{O(1)}$.*

▶ Remark. As we will see in the next section, our main algorithm can be extended to other settings. However, for axis-aligned rectangles, we can improve the dependency on d if we use a more sophisticated construction of ε-approximations for rectangular ranges. In fact, using [41] to construct the ε-approximation [11], we can get a distribution $\tilde{D}$ with the same properties as in Theorem 9 in $O((n + \delta^{-6} \log^2 n + \delta^{-d/2})\delta^{-2} \log n \log \delta^{-1})$ time.

4 Extensions

4.1 Extending to other error functions

So far, we only focused on the ℓ_1 empirical error. Our algorithm can be extended to ℓ_∞ and ℓ_2 empirical errors with the same asymptotic complexity. In both cases, we run the same binary search as we had for the ℓ_1 error, and we call the IsFeasible$(\mathcal{Z}, \delta, \alpha)$.

ℓ_∞ error. Following the same arguments as in Section 2.1, we can formulate the problem as a simpler LP than (LP1). More specifically, by definition, we have to satisfy $u_i \leq \alpha$ for $1 \leq i \leq n$ instead of constraint (4). However, we observe that the linear problem is then equivalent to (LP1) setting $u_i = \alpha$ for $1 \leq i \leq n$. Without having a vector u, in each round t, the algorithm only computes the vector v by computing $p^{(t)}$ as we had in Section 3. The result follows.

ℓ_2 error. The goal now is to find a distribution $\mathcal{D}$ that minimizes $\mathrm{err}(\mathcal{D}, \mathcal{Z})$. This is equivalent to the feasibility problem (LP1) replacing constraint (4) with

$$-\frac{1}{n}\sum_{i=1}^{n} u_i^2 \geq -\alpha. \tag{18}$$

Interestingly, the MWU method works even if the feasibility problem is on the form $f(\mathsf{x}) \geq 0$, where $f(\cdot)$ is concave. It is straightforward to see that the new constraint (18) is concave. The rest of the constraints remain the same as in (LP1) so they are linear. Hence, we can use the same technique to optimize the new feasibility problem. Still the goal is to try and satisfy the expected constraint (8) by maximizing its LHS. In fact the new LHS of (8) is

$$\mathsf{w}^{(t)\top}\mathsf{A}\mathsf{x} = \sum_{i=1}^{n} -\frac{w_0^{(t)}}{n}u_i^2 + (w_{2i}^{(t)} + w_{2i-1}^{(t)})u_i + \sum_{j=1}^{m}\psi_j^{(t)}v_j \tag{19}$$

The variables v_i are set by efficiently computing $p^{(t)}$ as shown in Section 3. So we only focus on setting the variables u_i. Recall that our goal is to maximize the LHS in order to check if the expected constraint is satisfied. By simple algebraic calculations it is straightforward to see that the function $g(u_i) = -\frac{w_0^{(t)}}{n}u_i^2 + (w_{2i}^{(t)} + w_{2i-1}^{(t)})u_i$ under the constraint $u_i \in [0,1]$ is maximized for

$$u_i^{(t)} = \min\left\{\frac{n(w_{2i}^{(t)} + w_{2i-1}^{(t)})}{2w_0^{(t)}}, 1\right\}. \tag{20}$$

After computing $\mathsf{w}^{(t)}$, we can find $\mathsf{u}^{(t)}$ in $O(n)$ time. Hence, using (20) instead of (13) we can set u that maximizes the expected constraint. All the other steps of the algorithm remain the same. Following the improved algorithm along with the proofs of Lemmas 2, 3, if $\alpha > \alpha_2^*(\mathcal{Z})$, with high probability, we get a distribution $\tilde{\mathcal{D}}$ of size $O(\delta^{-2})$ in near linear time such that $\mathrm{err}_2(\tilde{\mathcal{D}}, \mathcal{Z}) \leq \alpha + \delta/2$.

▶ **Theorem 10.** *Let $\mathcal{Z} = \{z_1, \ldots, z_n\}$ be a set of training samples such that $z_i = (R_i, s_i)$, where R_i is an axis-aligned rectangle in $\mathbb{R}^d$ and $s_i \in [0,1]$ is its selectivity. Given the parameters $\delta \in (0,1)$ and $p \in \{1, 2, \infty\}$, a discrete distribution $\tilde{\mathcal{D}}$ of size $O(\delta^{-2})$ can be computed in $O((n + \delta^{-2}\log^2 n + \delta^{-d}\log n)\delta^{-2}\log n \log \delta^{-1})$ time such that $\mathrm{err}_p(\mathcal{Z}, \tilde{\mathcal{D}}) \leq \alpha_p^*(\mathcal{Z}) + \delta$ with probability at least $1 - 1/n^{O(1)}$.*

4.2 Extending to other types of ranges

Besides rectangles, our algorithm can be extended to more general ranges such as balls and halfspaces in $\mathbb{R}^d$. A *semi-algebraic* set in $\mathbb{R}^d$ is a set defined by Boolean functions over polynomial inequalities, such as a unit semi-disc $\{(x^2+y^2 \leq 1) \cap (x \geq 0)\}$, or an annulus $\{(x^2+y^2 \leq 4) \cap (x^2+y^2 \geq 1)\}$. Most familiar geometric shapes such as balls, halfspaces, simplices, ellipsoids, are semi-algebraic sets. The *complexity* of a semi-algebraic set is the sum of the number of polynomial inequalities and their maximum degree. See [6] for a discussion on semi-algebraic sets. Our results extend to semi-algebraic ranges of constant complexity. Let $\mathcal{Z} = \{(\Gamma_1, s_1), \ldots, (\Gamma_n, s_n)\}$ be a training set where each Γ_i is a semi-algebraic set of constant complexity. Let $\Gamma = \{\Gamma_1, \ldots, \Gamma_n\}$. It is known that $\mathcal{A}(\Gamma)$ has $n^{O(d)}$ complexity, that for any weight function $\omega \colon \Gamma \to [0,1]$, the deepest point can be computed in $O(n^d)$ time, and that a random sample of size $O(\delta^{-2}\log\phi^{-1})$ is a δ-approximation with probability at least $1-\phi$, [2, 3]. With these primitives at our disposal, the algorithm in Section 3 can be extended to this setting. Omitting all the details we obtain the following.

▶ **Theorem 11.** *Let $\mathcal{Z} = \{z_1, \ldots, z_n\}$ be a set of training samples such that $z_i = (\Gamma_i, s_i)$, where Γ_i is a semi-algebraic set of constant complexity and $s_i \in [0,1]$ is its selectivity. Given the parameters $\delta \in (0,1)$ and $p \in \{1, 2, \infty\}$, a discrete distribution $\tilde{\mathcal{D}}$ of size $O(\delta^{-2})$ can be computed in $O((n+\delta^{-2}\log^2 n + \delta^{-2d}\log n)\delta^{-2}\log n \log \delta^{-1})$ time such that $\mathrm{err}_p(\mathcal{Z}, \tilde{\mathcal{D}}) \leq \alpha^*_p(\mathcal{Z}) + \delta$ with probability at least $1 - 1/n^{O(1)}$.*

5 Hardness Results

5.1 NP-Completeness

Let Selectivity be the decision problem of constructing a distribution with small size and minimum error. More specifically, let $\mathcal{Z} = \{(R_1, s_1), ..., (R_n, s_n)\}$ be the input *training set* consisting of n rectangles $R_1, \ldots, R_n$ in $\mathbb{R}^d$ along with their selectivities $s_1, \ldots, s_n$, respectively. Let k be a positive natural number and $\varepsilon \in [0,1]$ be an error parameter. The Selectivity problem asks if there exists a distribution $\mathcal{D} \in \mathbb{D}_k$ such that $\mathrm{err}_p(\mathcal{D}, \mathcal{Z}) \leq \varepsilon$. In the full version of the paper [1], we prove the following theorem.

▶ **Theorem 12.** *The* Selectivity *problem is* NP*-Complete, even for $d = 2$.*

The NP-hardness proof is based on a gadget used by [37] to prove the NP-hardness of the *Square Cover* problem: given a set $\mathcal{R}$ of n squares, and a parameter k, decide if there are k points $S \in \mathbb{R}^2$ such that $R \cap S \neq \emptyset$ for each $R \in \mathcal{R}$. We prove Theorem 12 by reducing 3-SAT to the Selectivity problem in $\mathbb{R}^2$. Let (X, C) be the 3-SAT formula consisting of variables $x_1, ..., x_n$ and clauses $C_1, ..., C_m$. We construct a set of weighted axis-aligned rectangles $\mathcal{Z}$ and a number k such that (X, C) is satisfiable if and only if there exists a discrete distribution $\mathcal{D}$ of size k such that $\mathrm{err}_p(\mathcal{D}, \mathcal{Z}) = 0$.

In the above reduction, we construct a training set $\mathcal{Z}$ such that there exists a $\mathcal{D} \in \mathbb{D}_k$ with $\mathrm{err}_p(\mathcal{D}, \mathcal{Z}) = 0$ if and only if the formula is satisfiable. This construction implies a stronger result. Given $\mathcal{Z}$ and k, let $\alpha^*_{p,k} = \min_{\mathcal{D} \in \mathbb{D}_k} \mathrm{err}_p(\mathcal{D}, \mathcal{Z})$.

▶ **Corollary 13.** *Let $f : \mathbb{N} \to \mathbb{N}$ be a monotonically non-decreasing function. Assuming $P \neq NP$, there is no polynomial-time algorithm that given a training set $\mathcal{Z}$ and an input parameter $k \in \mathbb{N}$ can compute $\mathcal{D} \in \mathbb{D}_k$ such that $\mathrm{err}_p(\mathcal{D}, \mathcal{Z}) \leq f(n)\alpha^*_{p,k}(\mathcal{Z})$.*

5.2 Conditional lower bounds

A natural question to ask is whether given $\mathcal{Z}$ and k, a discrete distribution of size at most $g(n)k$ can be computed in polynomial time such that the error is at most $f(n)\alpha^*_{p,k}$, where $f(n), g(n)$ are monotonically non-decreasing non-negative functions. We prove conditional lower bounds for this problem, assuming the FPT $\neq W[1]$ conjecture; see [14] for details of this conjecture.

Consider the following problem, which we refer to as SELECTIVITY+: given a training set $\mathcal{Z}$ of n rectangles in $\mathbb{R}^d$ along with their associated selectivities and a parameter k, compute a discrete distribution $\mathcal{D} \in \mathbb{D}$ such that $|\mathcal{D}| \leq g(n)k$ and $\text{err}_p(\mathcal{D}, \mathcal{Z}) \leq f(n)\alpha^*_{p,k}$, where $g : \mathbb{N} \to \mathbb{N}$ and $f(n) : \mathbb{N} \to \mathbb{N}$ are any functions that satisfy $f(n) \geq 1$, $g(n) \geq 1$, and $g(n) \cdot k = O(n)$.

We prove a conditional lower bound on SELECTIVITY+ by a reduction from the *coverage problem*: given a set $\mathcal{R}$ of n rectangles in $\mathbb{R}^d$ and a box B, does the union of rectangles in $\mathcal{R}$ cover B, i.e. $B \subseteq \bigcup_{R \in \mathcal{R}} R$. The coverage problem is known to be $W[1]$-Hard, therefore, there cannot be an algorithm for coverage with running time $h(d)n^{O(1)}$, where $h : \mathbb{N} \to \mathbb{N}$, under the FPT $\neq W[1]$ conjecture.

▶ **Theorem 14.** *The SELECTIVITY+ problem cannot be solved in $h(d)n^{O(1)}$ time, where h is any computable function $h : \mathbb{N} \to \mathbb{N}$, under the FPT $\neq W[1]$ conjecture.*

Proof. Suppose on the contrary there is an algorithm $\mathcal{A}$ for SELECTIVITY+ that runs in $h(d)n^{O(1)}$ time.

We show a reduction from the coverage problem. Let $\mathcal{R}, B$, as defined above, be an instance of the coverage problem. For each rectangle $R_i \in \mathcal{R}$, we construct $z_i = (R_i, 0)$, i.e., rectangle R_i with selectivity 0. Finally, we add $z_n = (B, 1)$. We set $k = 1$. The running time to construct the instance of the SELECTIVITY+ problem is $O(d \cdot n)$. We then run $\mathcal{A}$ on this instance.

We first consider the case when the input is a NO instance of the coverage problem. This implies that there exists a point $q \in \mathbb{R}^d$ such that $q \cap B \neq \emptyset$ and $q \bigcap (\bigcup_{R_i \in \mathcal{R}} R_i) = \emptyset$. Consider the distribution $\mathcal{D} = \{(q, 1)\}$. Notice that $\text{err}_p(\mathcal{D}, \mathcal{Z}) = 0 = \alpha^*_{p,1} = \alpha^*_{p,k} = f(n)\alpha^*_{p,k}$ because q lies only inside B, where B has selectivity 1 and all other rectangles R_i have selectivity 0. By definition of SELECTIVITY+, $\mathcal{A}$ in this instance must return a distribution $\mathcal{D}$ such that $\text{err}_p(\mathcal{D}, \mathcal{Z}) = 0$.

In the *YES* instance of the coverage problem, if $q \in B$ then $q \in \bigcup_{R_i \in \mathcal{R}} R_i$. Hence, if we add any point $q \in B$ in distribution $\mathcal{D}$ with positive probability then $\text{err}_p(\mathcal{D}, \mathcal{Z}) > 0$ because the selectivity of every rectangle in $\mathcal{R}$ is 0. On the other hand, the selectivity of B is 1 so if we do not add any point $q \in B$ in $\mathcal{D}$ with positive probability then $\text{err}_p(\mathcal{D}, \mathcal{Z}) > 0$. In any case, $f(n)\alpha^*_{p,k} \geq \text{err}_p(\mathcal{D}, \mathcal{Z}) > 0$. Actually, it is easy to verify that for any $\mathcal{D} \in \mathbb{D}$, $\text{err}_p(\mathcal{D}, \mathcal{Z}) > \frac{1}{2n^2}$ for any $p \in \{1, 2, \infty\}$. Thus, $f(n)\alpha^*_{p,k} > \frac{1}{2n^2} > 0$.

Overall, let $\mathcal{D}$ be the distribution returned by $\mathcal{A}$. Given $\mathcal{D}$, we can compute $\text{err}_p(\mathcal{D}, \mathcal{Z})$ in $O(n^2)$ time, because $|\mathcal{D}| \leq g(n)k \leq n$. If $\text{err}_p(\mathcal{D}, \mathcal{Z}) = 0$ then the solution to the coverage problem in the original instance is *NO*. Otherwise, if $\text{err}_p(\mathcal{D}, \mathcal{Z}) > 0$, then the solution to the coverage problem in the original instance is *YES*.

We thus obtain a $h(d)n^{O(1)}$ time algorithm for the coverage problem, which is a contradiction under the FPT $\neq W[1]$ conjecture. Hence, $\mathcal{A}$ cannot run in $h(d)n^{O(1)}$ time under the FPT $\neq W[1]$ conjecture, as claimed. ◀

The above theorem implies there is no near-linear algorithm for SELECTIVITY+ under the FPT $\neq W[1]$ conjecture. In fact, we can prove a stronger conditional lower bound. We define the APPROX-SELECTIVITY problem: given a training set $\mathcal{Z}$ of n rectangles in $\mathbb{R}^d$ along with their associated selectivities and a parameter $\delta > 0$, compute a $\mathcal{D} \in \mathbb{D}$ such that $|\mathcal{D}| \leq \frac{1}{\delta^3}$ and $\text{err}_p(\mathcal{D}, \mathcal{Z}) \leq f(n)\alpha^*_{p,1} + \delta$, where $f(n) : \mathbb{N} \to \mathbb{N}$ is any function that satisfies $f(n) \geq 1$.

The requirements for APPROX-SELECTIVITY are considerably weak. First, we are allowed to use more points than our algorithm i.e. $\frac{1}{\delta^3}$ points. Second, we only need to compete with a multiplicative factor of the best solution using just one point i.e. $f(n)\alpha^*_{p,1}$. It turns out that for small δ, the conditions of the APPROX-SELECTIVITY become the same as SELECTIVITY+ and hence the proof technique for Theorem 14 also proves the following:

▶ **Theorem 15.** *The APPROX-SELECTIVITY problem cannot be solved in $h(d)\big(n^{O(1)} + \delta^{-O(1)} + n^{O(1)}\delta^{-O(1)}\big)$ time, where h is any computable function $h : \mathbb{N} \to \mathbb{N}$, under the* FPT $\neq W[1]$ *conjecture.*

Proof. Similar to the proof of Theorem 14, we map an instance of the coverage problem to an instance of APPROX-SELECTIVITY, picking $\delta = \frac{1}{8n^2}$. The same argument shows that in the NO instance an algorithm for APPROX-SELECTIVITY returns a distribution with error at most $\frac{1}{8n^2}$ while in the YES instance, it returns a distribution with error at least $\frac{1}{2n^2}$. ◀

▶ Remark. Notice that both theorems are based on reductions from the coverage problem. Chan [8] proved that the coverage problem is $W[1]$-hard with respect to dimension d, showing that the existence of a d-clique in a graph with $\sqrt{n}$ vertices can be reduced to the coverage problem with $O(n)$ boxes in $\mathbb{R}^d$. Thus, the time complexity of the d-clique problem is intimately related to our problem. If fast matrix multiplication is allowed, the best known algorithm for the d-clique problem requires $\omega(n^{d/3})$ time [39].

Due to inefficiencies of fast matrix multiplication, there is a lot of interest in solving the clique problem using combinatorial algorithms (i.e. without fast matrix multiplication). However, the best current combinatorial algorithm for the d-clique problem in general graphs requires $\Omega(n^d/\operatorname{polylog}(n))$ running time. This lower bound implies that an $o(n^{d/2-\eta})$ time combinatorial algorithm for the SELECTIVITY+ problem or a $o(n + \delta^{-d/2+\eta})$ time combinatorial algorithm for the the APPROX-SELECTIVITY problem, for any $\eta > 0$, is unlikely. Such an algorithm would lead to a $o(n^{d-\eta})$ combinatorial algorithm for d-clique, solving a major open problem in graph theory.

6 Related Work

Selectivity Estimation. Selectivity estimation techniques can be broadly classified into three regimes: query-driven, data-driven, and hybrid. Most literature focuses on orthogonal range queries.

Query-driven methods: These methods derive selectivity estimates based on previous queries and their results. They do not require access to the underlying data distribution. Methods falling under this category include DQM [24] and the query-driven histogram techniques such as STHoles [7], Isomer [45], and QuickSel [40]. They construct models or histograms using results from previous queries and apply these models to estimate selectivity for new queries.

Data-driven methods: These methods, on the other hand, derive selectivity estimates from the underlying data distribution. They sample data and build statistical models that capture the data distribution, which are then used for selectivity estimation. There is long history along this direction, but some recent examples in this category include Naru [49], DQM-D [35], and DeepDB[26].

Hybrid methods: These methods take into account both the query workload and the underlying data distribution. They not only model the data distribution but also incorporate query feedback to refine the model over time. Examples of such methods include MSCN [28] and LW [15], which utilize both data and query features in a regression-based framework for selectivity estimation. See [47] for a detailed literature review.

Our method is query-driven, but is unique because none of query-driven and data-driven models above offered theoretical guarantees on the learnability of the functions or their learning procedures.

Multiplicative-Weights-Update (MWU) method. The MWU method has been independently rediscovered multiple times and has found applications in several areas including machine learning [17], game theory [18, 20], online algorithms [16], computational geometry [12], etc. In the context of optimization problems, there is a rich body of work on using MWU and related techniques to efficiently solve special classes of LPs and SDPs [30, 42, 50, 4]. The technique we use in this paper is an adaption of the one used by [42] to solve packing and covering LPs. See also [5] for a detailed overview. The MWU method has also been used to implicitly solve linear programs. A classical example is the multi-commodity flow problem where the describing the LP explicitly requires exponential number constraints [19]. Furthermore, there has been a lot of work related to implicitly solving special classes of LPs in computational geometry or combinatorial optimization using the MWU method [10, 12, 13], the primal dual method [48], or the ellipsoid method [22, 21].

Despite the geometric nature of our problem, to the best of our knowledge, all previous (geometric) techniques that implicitly solve an LP do not extend to our problem. In particular, in [12] the authors study various geometric packing and covering problems, such as the weighted set cover of points by disks, and the maximum weight independent set of disks, using the MWU method. The main difference from our setting is that for all problems they study, the matrix A and the vector b are non-negative. One of the main challenges in our setting is that we have to deal with both positive and negative values in A and b. It is not clear how their algorithms can be extended to our setting. For example, for the maximum weight independent set problem, they also describe an efficient algorithm to compute an approximately deepest point. However, their algorithm works in 2 dimensions while all ranges have positive weights. In our setting, we have ranges in any constant dimension d with both positive and negative weights.

7 Conclusion and future work

In this work, we studied the problem of finding the data distribution to fit a query training set consisting of axis-aligned rectangles (representing orthogonal range selectivity queries) with the smallest error. While the problem has been studied in the past, only an expensive $\Omega(n^d)$ algorithm was known for constructing a distribution of size $O(n^d)$. We showed that the decision problem is NP-complete even for $d = 2$. Based on a standard complexity conjecture, we also gave conditional lower bounds showing that the exponential dependency on d is inevitable for additive or relative approximations. On the positive side, for the ℓ_1 empirical error, we gave a $O((n + \delta^{-d})\delta^{-2} \operatorname{polylog} n)$ time algorithm that returns a data distribution of size $O(\delta^{-2})$ with additive error δ. Furthermore, we showed that our algorithm for ℓ_1 error can be extended to ℓ_2 and ℓ_∞, as well as any type of ranges as long as they are algebraic sets of constant complexity. In view of our hardness results, significant improvements to our upper bounds are unlikely.

There are some interesting directions following from this work. What properties should the data and query distribution satisfy so that the running time is polynomial in d as well? In addition to selectivity queries, another interesting line of work is to study the construction of distributions for other types of database queries such as aggregation and joins.

References

1 Pankaj Agarwal, Rahul Raychaudhury, Stavros Sintos, and Jun Yang. Computing data distribution from query selectivities. *CoRR*, abs/2401.06047, 2024. doi:10.48550/arXiv.2401.06047.

2 Pankaj K. Agarwal, Boris Aronov, Esther Ezra, and Joshua Zahl. Efficient algorithm for generalized polynomial partitioning and its applications. *SIAM Journal on Computing*, 50(2):760–787, 2021. doi:10.1137/19M1268550.

3 Pankaj K. Agarwal and Micha Sharir. Arrangements and their applications. In *Handbook of computational geometry*, pages 49–119. Elsevier, 2000. doi:10.1016/b978-044482537-7/50003-6.

4 Sanjeev Arora, Elad Hazan, and Satyen Kale. Fast algorithms for approximate semidefinite programming using the multiplicative weights update method. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pages 339–348. IEEE, 2005. doi:10.1109/SFCS.2005.35.

5 Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of computing*, 8(1):121–164, 2012. doi:10.4086/toc.2012.v008a006.

6 S. Basu, R. Pollack, and M. F. Roy. Algorithms in real algebraic geometry. In *Algorithms and Computation in Mathematics*. 2nd ed., Springer-Verlag, 2000.

7 Nicolas Bruno, Surajit Chaudhuri, and Luis Gravano. Stholes: A multidimensional workload-aware histogram. In Sharad Mehrotra and Timos K. Sellis, editors, *Proceedings of the 2001 ACM SIGMOD international conference on Management of data, Santa Barbara, CA, USA, May 21-24, 2001*, pages 211–222. ACM, 2001. doi:10.1145/375663.375686.

8 Timothy M. Chan. A (slightly) faster algorithm for klee's measure problem. In *Proceedings of the twenty-fourth annual symposium on Computational geometry*, pages 94–100, 2008. doi:10.1145/1377676.1377693.

9 Timothy M. Chan. Klee's measure problem made easy. In *2013 IEEE 54th annual symposium on foundations of computer science*, pages 410–419. IEEE, 2013. doi:10.1109/FOCS.2013.51.

10 Timothy M. Chan and Qizheng He. Faster approximation algorithms for geometric set cover. In *36th International Symposium on Computational Geometry (SoCG 2020)*, 2020. doi:10.4230/LIPIcs.SoCG.2020.27.

11 Bernard Chazelle. *The discrepancy method: randomness and complexity*. Cambridge University Press, 2000.

12 Chandra Chekuri, Sariel Har-Peled, and Kent Quanrud. Fast lp-based approximations for geometric packing and covering problems. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1019–1038. SIAM, 2020. doi:10.1137/1.9781611975994.62.

13 Kenneth L. Clarkson and Kasturi Varadarajan. Improved approximation algorithms for geometric set cover. In *Proceedings of the twenty-first annual symposium on Computational geometry*, pages 135–141, 2005. doi:10.1145/1064092.1064115.

14 Marek Cygan, Fedor V. Fomin, Lukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized algorithms*, volume 5(4). Springer, 2015. doi:10.1007/978-3-319-21275-3.

15 Anshuman Dutt, Chi Wang, Azade Nazi, Srikanth Kandula, Vivek R. Narasayya, and Surajit Chaudhuri. Selectivity estimation for range predicates using lightweight models. *Proc. VLDB Endow.*, 12(9):1044–1057, 2019. doi:10.14778/3329772.3329780.

16 Dean P. Foster and Rakesh Vohra. Regret in the on-line decision problem. *Games and Economic Behavior*, 29(1-2):7–35, 1999.

17 Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997. doi:10.1006/jcss.1997.1504.

18 Yoav Freund and Robert E. Schapire. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29(1-2):79–103, 1999.

19 Naveen Garg and Jochen Könemann. Faster and simpler algorithms for multicommodity flow and other fractional packing problems. *SIAM Journal on Computing*, 37(2):630–652, 2007. doi:10.1137/S0097539704446232.

20 Michael D. Grigoriadis and Leonid G. Khachiyan. A sublinear-time randomized approximation algorithm for matrix games. *Operations Research Letters*, 18(2):53–58, 1995. doi:10.1016/0167-6377(95)00032-0.

21 Martin Grötschel, László Lovász, and Alexander Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1:169–197, 1981. doi:10.1007/BF02579273.

22 Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric algorithms and combinatorial optimization*, volume 2. Springer Science & Business Media, 2012.

23 Sariel Har-Peled. Geometric approximation algorithms. In *Mathematical Surveys and Monographs*, volume 173. American Mathematical Soc., 2011.

24 Shohedul Hasan, Saravanan Thirumuruganathan, Jees Augustine, Nick Koudas, and Gautam Das. Deep learning models for selectivity estimation of multi-attribute queries. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1035–1050, 2020. doi:10.1145/3318464.3389741.

25 David Haussler. Decision theoretic generalizations of the pac model for neural net and other learning applications. *Information and computation*, 100(1):78–150, 1992. doi:10.1016/0890-5401(92)90010-D.

26 Benjamin Hilprecht, Andreas Schmidt, Moritz Kulessa, Alejandro Molina, Kristian Kersting, and Carsten Binnig. Deepdb: learn from data, not from queries! *Proceedings of the VLDB Endowment*, 13(7):992–1005, 2020. doi:10.14778/3384345.3384349.

27 Xiao Hu, Yuxi Liu, Haibo Xiu, Pankaj K. Agarwal, Debmalya Panigrahi, Sudeepa Roy, and Jun Yang. Selectivity functions of range queries are learnable. In *SIGMOD '22: International Conference on Management of Data, Philadelphia, PA, USA, June 12–17, 2022*, pages 959–972. ACM, 2022. doi:10.1145/3514221.3517896.

28 Andreas Kipf, Thomas Kipf, Bernhard Radke, Viktor Leis, Peter A. Boncz, and Alfons Kemper. Learned cardinalities: Estimating correlated joins with deep learning. In *9th Biennial Conference on Innovative Data Systems Research, CIDR 2019, Asilomar, CA, USA, January 13–16, 2019, Online Proceedings*, 2019. URL: http://cidrdb.org/cidr2019/papers/p101-kipf-cidr19.pdf.

29 Alexander Kogler and Patrick Traxler. Parallel and robust empirical risk minimization via the median trick. In *Mathematical Aspects of Computer and Information Sciences: 7th International Conference, MACIS 2017, Vienna, Austria, November 15–17, 2017, Proceedings*, pages 378–391. Springer, 2017. doi:10.1007/978-3-319-72453-9_31.

30 Christos Koufogiannakis and Neal E. Young. Beating simplex for fractional packing and covering linear programs. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20–23, 2007, Providence, RI, USA, Proceedings*, pages 494–504. IEEE Computer Society, 2007. doi:10.1109/FOCS.2007.16.

31 Richard J. Lipton, Jeffrey F. Naughton, and Donovan A. Schneider. Practical selectivity estimation through adaptive sampling. In *Proceedings of the 1990 ACM SIGMOD international conference on Management of data*, pages 1–11, 1990. doi:10.1145/93597.93611.

32 Ryan Marcus, Parimarjan Negi, Hongzi Mao, Nesime Tatbul, Mohammad Alizadeh, and Tim Kraska. Bao: Making learned query optimization practical. *SIGMOD Rec.*, 51(1):6–13, 2022. doi:10.1145/3542700.3542703.

33 Ryan Marcus, Parimarjan Negi, Hongzi Mao, Chi Zhang, Mohammad Alizadeh, Tim Kraska, Olga Papaemmanouil, and Nesime Tatbul. Neo: a learned query optimizer. *Proceedings of the VLDB Endowment*, 12(11), 2019. doi:10.14778/3342263.3342644.

34 Ryan Marcus and Olga Papaemmanouil. Deep reinforcement learning for join order enumeration. In *Proceedings of the First International Workshop on Exploiting Artificial Intelligence Techniques for Data Management*, pages 1–4, 2018. doi:10.1145/3211954.3211957.

35 Volker Markl, Peter J. Haas, Marcel Kutsch, Nimrod Megiddo, Utkarsh Srivastava, and Tam Minh Tran. Consistent selectivity estimation via maximum entropy. *The VLDB journal*, 16:55–76, 2007. doi:10.1007/s00778-006-0030-1.

36 Yossi Matias, Jeffrey Scott Vitter, and Min Wang. Wavelet-based histograms for selectivity estimation. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pages 448–459, 1998. doi:10.1145/276304.276344.

37 Nimrod Megiddo and Kenneth J. Supowit. On the complexity of some common geometric location problems. *SIAM J. Comput.*, 13(1):182–196, 1984. doi:10.1137/0213014.

38 Parimarjan Negi, Ryan Marcus, Hongzi Mao, Nesime Tatbul, Tim Kraska, and Mohammad Alizadeh. Cost-guided cardinality estimation: Focus where it matters. In *2020 IEEE 36th International Conference on Data Engineering Workshops (ICDEW)*, pages 154–157. IEEE, 2020. doi:10.1109/ICDEW49219.2020.00034.

39 Jaroslav Nešetřil and Svatopluk Poljak. On the complexity of the subgraph problem. *Commentationes Mathematicae Universitatis Carolinae*, 26(2):415–419, 1985.

40 Yongjoo Park, Shucheng Zhong, and Barzan Mozafari. Quicksel: Quick selectivity learning with mixture models. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*, pages 1017–1033, 2020. doi:10.1145/3318464.3389727.

41 Jeff M. Phillips. Algorithms for epsilon-approximations of terrains. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games*, volume 5125 of *Lecture Notes in Computer Science*, pages 447–458. Springer, 2008. doi:10.1007/978-3-540-70575-8_37.

42 Serge A. Plotkin, David B. Shmoys, and Éva Tardos. Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research*, 20(2):257–301, 1995. doi:10.1287/moor.20.2.257.

43 Viswanath Poosala, Peter J. Haas, Yannis E. Ioannidis, and Eugene J. Shekita. Improved histograms for selectivity estimation of range predicates. *ACM Sigmod Record*, 25(2):294–305, 1996. doi:10.1145/233269.233342.

44 Viswanath Poosala and Yannis E. Ioannidis. Selectivity estimation without the attribute value independence assumption. In *VLDB*, volume 97, pages 486–495, 1997. URL: http://www.vldb.org/conf/1997/P486.PDF.

45 Utkarsh Srivastava, Peter J. Haas, Volker Markl, Marcel Kutsch, and Tam Minh Tran. ISOMER: consistent histogram construction using query feedback. In Ling Liu, Andreas Reuter, Kyu-Young Whang, and Jianjun Zhang, editors, *Proceedings of the 22nd International Conference on Data Engineering, ICDE 2006, 3-8 April 2006, Atlanta, GA, USA*, page 39. IEEE Computer Society, 2006. doi:10.1109/ICDE.2006.84.

46 Markus Stocker, Andy Seaborne, Abraham Bernstein, Christoph Kiefer, and Dave Reynolds. Sparql basic graph pattern optimization using selectivity estimation. In *Proceedings of the 17th international conference on World Wide Web*, pages 595–604, 2008. doi:10.1145/1367497.1367578.

47 Xiaoying Wang, Changbo Qu, Weiyuan Wu, Jiannan Wang, and Qingqing Zhou. Are we ready for learned cardinality estimation? *Proceedings of the VLDB Endowment*, 14(9):1640–1654, 2021. doi:10.14778/3461535.3461552.

48 David P. Williamson. The primal-dual method for approximation algorithms. *Mathematical Programming*, 91:447–478, 2002. doi:10.1007/s101070100262.

49 Zongheng Yang, Eric Liang, Amog Kamsetty, Chenggang Wu, Yan Duan, Xi Chen, Pieter Abbeel, Joseph M. Hellerstein, Sanjay Krishnan, and Ion Stoica. Deep unsupervised cardinality estimation. *Proceedings of the VLDB Endowment*, 13(3):279–292, 2019. doi:10.14778/3368289.3368294.

50 Neal E. Young. Sequential and parallel algorithms for mixed packing and covering. In *Proceedings 42nd IEEE symposium on foundations of computer science*, pages 538–546. IEEE, 2001. doi:10.1109/SFCS.2001.959930.