Selectivity Functions of Range Queries are Learnable

Xiao Hu, Yuxi Liu, Haibo Xiu, Pankaj K. Agarwal, Debmalya Panigrahi, Sudeepa Roy, and Jun Yang
{xh102,yuxi.liu,haibo.xiu,pankaj,debmalya,sudeepa,junyang}@cs.duke.edu
Duke University, Durham, NC, USA

Abstract
This paper explores the use of machine learning for estimating the
selectivity of range queries in database systems. Using classic learn-
ing theory for real-valued functions based on shattering dimension,
we show that the selectivity function of a range space with bounded
VC-dimension is learnable. As many popular classes of queries (e.g.,
orthogonal range search, inequalities involving linear combination
of attributes, distance-based search, etc.) represent range spaces
with finite VC-dimension, our result immediately implies that their
selectivity functions are also learnable. To the best of our knowl-
edge, this is the first attempt at formally explaining the role of
machine learning techniques in selectivity estimation, and comple-
ments the growing literature in empirical studies in this direction.
Supplementing these theoretical results, our experimental results
demonstrate that, empirically, even a basic learning algorithm with
generic models is able to produce accurate predictions across set-
tings, matching state-of-art methods designed for specific queries,
and using training sample sizes commensurate with our theory.

CCS Concepts
• Information systems → Database query processing. • The-
ory of computation → Sample complexity and generalization
bounds. Database query processing and optimization (the-
ory).

KEYWORDS
learning theory, selectivity estimation, range space, fat-shattering
dimension, Vapnik–Chervonenkis dimension

ACM Reference Format:
Xiao Hu, Yuxi Liu, Haibo Xiu, Pankaj K. Agarwal, Debmalya Panigrahi,
Sudeepa Roy, and Jun Yang. 2022. Selectivity Functions of Range Queries
are Learnable. In Proceedings of the 2022 International Conference on
Management of Data (SIGMOD ’22), June 12–17, 2022, Philadelphia, PA, USA.
ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3514221.3517896

1 INTRODUCTION
In this paper, we formally model and study the problem of learning
selectivity functions for selection queries in database (DB) systems.

The selectivity of a selection query on a database is defined as
the probability that a randomly chosen tuple from the database
satisfies the query predicate. Estimating query selectivity is a core
problem in the query optimization pipeline, and has a rich history of
research over many decades (see, e.g., [24, 30, 37, 38, 40]). In recent
years, the focus has shifted from traditional optimization methods
to machine learning (ML) techniques (e.g., [16, 25–27, 34, 36]), with the
latter outperforming the former in empirical studies. In this paper,
we establish a learning-theoretic framework for the selectivity-
estimation problem, show that the estimation problem is indeed
learnable for popular classes of selection queries from a small set of
training samples using this framework. Building on this framework,
we also develop a simple, generic learning algorithm and evaluate it
empirically: not only is this approach competitive against the state-
of-the-art methods designed for specific types of queries, but it also
works effectively for other less-studied query types, demonstrating
the power and generality of our framework.

While the query selectivity estimation problem is indeed an
important component of DB research, we believe that our work
also has implications beyond this specific problem. Our research
adds to the growing and impressive body of work that seeks to
exploit the vast advances in ML in recent years to solve problems
in DB systems. The main thrust in this area of research has been in
developing ML models and algorithms, often using deep learning
techniques, that empirically outperform existing methods in real
world DB systems. We complement this by providing a formal
framework to establish the learnability of the selectivity estimation
problem. As “ML for DB” advances further, we hope that the formal
lens that we introduce in this paper can be adapted and generalized
to a broader class of DB problems.

Our Contributions. First, we formalize the learnability of the
selectivity-estimation problem. Recall that a database is a collection
of tuples, and a selection query is a predicate that selects a subset of these tuples. The selectivity of a selection query is the probability that a randomly selected tuple satisfies the query. In order to learn the selectivity function, we employ the agnostic-learning framework [17], an extension of the classical PAC learning framework for real-valued functions, where we are given a set of sample queries and their respective selectivities from a fixed distribution (the \textit{training set}), and our goal is to design an algorithm that can output the selectivity of a new query from the same distribution with high accuracy (see Figure 1 for an example).

Classical PAC learning theory asserts that a Boolean function is learnable if its VC-dimension is bounded. Generalizing this notion, it has been shown that a real-valued function is learnable using finitely many samples if its \textit{fat shattering dimension} (defined in Section 2) is bounded [5, 7, 21]. This reduces the question of learnability of selectivity functions to bounding their respective fat shattering dimensions. We further note that selectivity functions correspond to selection queries on the underlying data. Each selection query, in turn, is a binary function on the data (i.e., which data items satisfy the query predicate), and the complexity of a class of binary functions is captured by its VC-dimension [45]. Our main result shows that if a class of selection queries has bounded VC-dimension, then the fat shattering dimension of the corresponding selectivity function must also be bounded, and therefore, the selectivity function for such queries is learnable.

This result has several implications for important query classes:

- **Orthogonal Range Queries.** Such queries are specified as a conjunction of range conditions on individual attributes, e.g.:

  \begin{verbatim}
  SELECT * FROM T WHERE \theta_0 + \theta_1 \times A_1 + \theta_2 \times A_2 + \ldots + \theta_d \times A_d \geq 0
  \end{verbatim}

  They are widely used as building blocks in more complex queries, and their selectivity-estimation (even for the simplest 1D range queries involving just a single attribute) has been the bread and butter of cost-based query optimizers, which uses selectivity estimates to gauge the intermediate result sizes and choose low-cost query execution plans. Taking a geometric view, we can represent each data tuple as a point in \(\mathbb{R}^d\) and each query as a hyper-rectangle in the same space. Known bounds on the VC-dimension of hyper-rectangles [22] then allow us to conclude that their selectivity is learnable.

- **Linear Inequality Queries.** Such queries allow multiple attributes to be brought together into one linear inequality, e.g.:

  \begin{verbatim}
  SELECT * FROM T WHERE \theta_0 + \theta_1 \times A_1 + \theta_2 \times A_2 + \ldots + \theta_d \times A_d \geq 0
  \end{verbatim}

  Able to capture more complex conditions that can encode data correlations, these queries are popular in advanced analytical systems. As earlier, we can represent each tuple on \(d\) attributes as a point in \(\mathbb{R}^d\). Then, each query is a half-space in \(\mathbb{R}^d\). Again, using known bounds on the VC-dimension of halfspaces [22], we conclude that the corresponding selectivity function is learnable.

- **Distance-based Queries.** These queries specify a “reference” object and find all objects that within some distance of it, e.g.:

  \begin{verbatim}
  SELECT * FROM T WHERE (A_1 - a_1)^2 + (A_2 - a_2)^2 + \ldots + (A_d - a_d)^2 \leq r^2
  \end{verbatim}

  Here the reference object is \((a_1, \ldots, a_d)\) and the Euclidean \((\ell_2)\) distance threshold is \(r\). Such queries have broad applications in text and image search, product recommendations, database optimization, network traffic, etc. Again, selectivity estimation enables cost-based optimization of queries involving such constructs. Moreover, the estimates may be of interest themselves; e.g., we might be interested in just counting how many other objects are in the vicinity of one object. As before, we use a geometric view where the data points are in \(\mathbb{R}^d\) for \(d\) attributes, and the above query is a \(d\)-dimensional \(\ell_2\)-ball. Inverting the standard bound [22] on the VC-dimension of \(\ell_2\)-balls, we can conclude that the corresponding selectivity function is learnable.

While our framework establishes the learnability of the selectivity of above query types from a small set of training examples, it does not by itself prescribe any specific model or learning algorithm. As part of establishing the learnability of our selectivity query, we also need a procedure that, given a set of training samples and a family of data distributions (e.g. histograms, discrete distributions), constructs a data distribution from the given family that “best fits” the training samples. Our framework then guarantees that the learned data distribution estimates the selectivity of any query chosen from the same distribution as the training samples with high accuracy. For specific query types (e.g., orthogonal range queries), there already exists a large body of work on the selectivity-estimation problem, and our framework now gives them a solid foundation. To demonstrate the power of our framework beyond justifying existing methods, we further propose a simple, generic approach that embodies our theoretical results, and empirically validates its efficiency using extensive experiments. It is important to note that we are not designing this generic approach to “beat” existing methods with novel or sophisticated features; in fact, we intentionally avoid sophisticated features so that experimental comparison can focus on illustrating the power of our unifying framework instead of the artifacts of extra features. Despite the simplicity of our approach, our experimental results show that it performs comparably to the state-of-the-art methods for orthogonal range queries. Furthermore, for query classes that have seen less previous research, such as linear inequality and distance-based queries, our generic approach also work effectively, demonstrating the generality of the our theoretical framework.

**Roadmap.** This paper is organised as follows. In Section 2, we focus on the statistical learning question of determining the sample complexity of training for selectivity estimation problem under the agnostic-learning framework. In Section 3, we propose two simple generic algorithms for computing a data distribution that minimizes the expected loss function on a finite set of training queries. In Section 4, we implement these two algorithms to verify our theory, both of which are trained using a certain number of queries for obtaining small predication error on test queries, and compare them with state-of-art methods under the same framework.

## 2 LEARNABILITY OF QUERY SELECTIVITY

A range space \(\Sigma\) is a pair \((X, R)\), where \(X\) is a set of objects and \(R\) is a collection of subsets of \(X\) called \textit{ranges}. For example, \(X = \mathbb{R}^d\) and \(R\) can be the set of all \(d\)-dimensional rectangles, halfspaces, or balls. Let \(D\) be a probability distribution over \(X\). For a given \(D\), we
define the selectivity function \( s_D : R \to [0, 1] \) as
\[
s_D(z) = \Pr_{x \sim D} [x \in R].
\]

Our goal is to learn the selectivities of the ranges in a range space \( \Sigma \) under an unknown data distribution from a finite sample of ranges and their respective selectivities. Formally, we define this learning task as follows.

2.1 The Learning Framework

Learnability. Following the agnostic learning model proposed by Haussler [17] (see also [5, 7]), which generalizes the PAC model, we define learnability in a more general setting. Let \( \mathcal{H} \) be a family of functions from a domain \( Y \) to \([0, 1]\]. Set \( Z = Y \times [0, 1] \). For a function \( H \in \mathcal{H} \), we define the loss function \( \ell_H : Z \to [0, 1] \). For \( z = (y, w) \in Z \),
\[
\ell_H(z) = (H(y) - w)^2.
\]

For a probability distribution \( Q \) over \( Z \) and for a function \( H \in \mathcal{H} \), we define
\[
er_Q(H) = \int_Z \ell_H(z) dQ(z)
\]
for some data distribution \( Q \) and \( \mathcal{H} \), to be the mean square loss of \( H \) with respect to distribution \( Q \).

A learning procedure \( \mathcal{A} \) is mapping from finite sequences in \( Z \) to \( \mathcal{H} \). Given a training sample \( z^n = (z_1, z_2, \ldots, z_n) \in Z^n \), \( \mathcal{A} \) returns a function \( A(z^n) \). Given \( \epsilon, \delta \in (0, 1) \), and an integer \( n > 0 \), we say that \( A \) \( (\epsilon, \delta) \)-learns (agnostically) from \( n \) random training samples with respect to \( \mathcal{H} \) if
\[
\sup_Q \Pr \left[ \epsilon_Q(A(z^n)) \geq \inf_{H \in \mathcal{H}} \epsilon_Q(H) + \epsilon \right] \leq \delta,
\]
where \( \Pr \) denotes the probability with respect to a random sample \( z^n \in Z^n \), each of \( z_1, z_2, \ldots, z_n \) drawn independently from \( Z \) at random according to \( Q \), and supremum is taken over all distributions defined on \( Z \). For \( \epsilon > 0 \), \( \mathcal{H} \) is called \( \epsilon \)-learnable if there exists a function \( n_0 : [0, 1]^2 \to \mathbb{N} \) and a learning procedure \( \mathcal{A} \) such that for all \( \delta > 0 \) and for all \( n \geq n_0(\epsilon, \delta) \), \( \mathcal{A} \) \( (\epsilon, \delta) \)-learns from \( n \) examples with respect to \( \mathcal{H} \). For \( n \geq n_0(\epsilon, \delta) \), \( \mathcal{A} \) is referred to as the minimum training set size for \( \mathcal{H} \). Finally, \( \mathcal{H} \) is learnable if it is \( \epsilon \)-learnable for all \( \epsilon > 0 \).

VC dimension. Returning to the selectivity function of range space \( \Sigma = (X, \mathcal{R}) \), let \( \mathcal{D} \) be a set of distributions defined on \( X \). Set \( \mathcal{S}_{\Sigma, \mathcal{D}} = \{ s_D : D \in \mathcal{D} \} \), a family of functions from \( \mathcal{R} \) to \([0, 1]\]. Set \( Z = X \times [0, 1] \). Our main result is a characterization of learnability of \( \mathcal{S}_{\Sigma, \mathcal{D}} \) in terms of the VC-dimension of \( \Sigma \), defined below.

A subset \( P \subseteq X \) is shattered by \( \mathcal{R} \) if \( \{ P \cap R \mid R \in \mathcal{R} \} = 2^P \). The VC-dimension of \( \mathcal{R} \), denoted by \( \text{VC-dim}(\Sigma) \), is the size of the largest subset of \( X \) that can be shattered by \( \Sigma \). An example is given in Figure 2. If the VC-dimension of \( \Sigma \) is not bounded by a constant, then \( \text{VC-dim}(\Sigma) = \infty \). Our main result, stated in the theorem below, is that \( \mathcal{S}_{\Sigma, \mathcal{D}} \) is learnable if and only if \( \text{VC-dim}(\Sigma) \) is finite.

**Theorem 2.1.** Let \( \Sigma = (X, \mathcal{R}) \) be a range space, let \( \mathcal{D} \) be a set of distributions defined on \( X \), and let \( \epsilon, \delta \in (0, 1) \) be a parameter. If \( \text{VC-dim}(\Sigma) = \lambda \), for some constant \( \lambda > 0 \), then the family \( \mathcal{S}_{\Sigma, \mathcal{D}} \) of selectivity functions is \( \epsilon \)-learnable with a training set size of \( \tilde{O}\left(\frac{1}{\epsilon^2 \lambda^2}\right) \). Conversely, if \( \text{VC-dim}(\Sigma) = \infty \), \( \mathcal{S}_{\Sigma, \mathcal{D}} \) is not (agnostically) learnable.\(^1\)

\(^1\)\(\tilde{O}(\cdot)\) to hide lower order terms that are in polylog \((\frac{1}{\epsilon}, \frac{1}{\delta}, \lambda)\) for constant \( \lambda \).

Remark. Note that we do not assume training sample \( z_i = (R_i, s_i) \in Z \) to be of the form \( s_i = s_D(R_i) \) for some data distribution \( D \in \mathcal{D} \). They are drawn from some distribution \( Q \) defined on \( \mathcal{R} \times [0, 1] \), and the goal is to learn the selectivity function in \( \mathcal{S}_{\Sigma, \mathcal{D}} \) that minimizes the mean square loss. This is important, which allows us to decouple training samples from the family of functions, and the problem just becomes to find a function from the given family that minimizes the expected loss. This model is more general than the one assuming training sample in a form of \( z_i = (R_i, s_D(R_i)) \) for some data distribution \( D \in \mathcal{D} \), for example, capturing the noisy input for learning the selectivity functions.

Instead of using the mean square error in (1), we can use other loss functions such as the \( L_1 \)-norm or \( L_\infty \)-norm of the error, i.e., \( \int_{(y, w)} |H(y) - w|.dQ(y, w) \) or \( \sup_{(y, w)} |H(y) - w| \). Furthermore, the theorem holds for any \( \mathcal{D} \), the family of data distributions and the bound on the training size is independent of \( \mathcal{D} \). It might be possible to obtain an improved bound on the training size for certain family of data distributions. Finally, the theorem assumes the existence of a procedure that efficiently computes the function in \( \mathcal{S}_{\Sigma, \mathcal{D}} \) that minimizes, or minimizes within additive error \( \epsilon \), the mean square loss over the finite sequence of training samples; see Section 3.

2.2 Implications of Theorem 2.1

Before proving Theorem 2.1, we give some of its implications. We begin with the query classes mentioned in the introduction.

**Orthogonal Range Queries:** The range space \( \Sigma_{\mathcal{O}} = (\mathbb{R}^d, \mathcal{R}_{\mathcal{O}}) \) for orthogonal range queries is defined as
\[
\mathcal{R}_{\mathcal{O}} = \{ [a,b] \times [a,b] : [a,b] \in \mathbb{R}^2, a \leq b, \forall i \in [d] \}.
\]
It is well known that \( \text{VC-dim}(\Sigma_{\mathcal{O}}) = 2d \) (see Figure 2 for \( d = 3 \)), therefore Theorem 2.1 implies that for any family \( \mathcal{D} \) of distributions defined on \( \mathbb{R}^d \) and for any \( \epsilon > 0 \), the selectivity functions are \( \epsilon \)-learnable with training set size of \( \tilde{O}\left(\frac{d}{\epsilon^2}\right) \).

**Linear Inequality Queries:** The range space \( \Sigma_\perp = (\mathbb{R}^d, \mathcal{R}_\perp) \) for linear inequality queries is defined as
\[
\mathcal{R}_\perp = \{ (a,b) : a \in \mathbb{R}^d, b \in \mathbb{R} \},
\]
where \( \mathcal{R}_{\perp(a,b)} = \{ x \in \mathbb{R}^d : a \cdot x \geq b \} \). It is known that \( \text{VC-dim}(\Sigma_\perp) = d + 1 \) (see Theorem 2.1), therefore Theorem 2.1 implies that for any family \( \mathcal{D} \)
of distributions defined on $\mathbb{R}^d$ and for any $\epsilon > 0$, the selectivity functions are $\epsilon$-learnable with training set of size $\tilde{O}\left(\frac{1}{\epsilon^d}\right)$.

**Distance-Based Queries:** The range space $\Sigma_\sigma = (\mathbb{R}^d, \mathcal{R}_\sigma)$ for distance-based queries is defined as

$$\mathcal{R}_\sigma = \{R_{\sigma(i,a,b)} : a \in \mathbb{R}^d, b \in \mathbb{R}\},$$

where $R_{\sigma(i,a,b)} = \{x \in \mathbb{R}^d : ||x - a|| \leq b\}$ and $||\cdot||$ is the Euclidean norm. It is known that VC-dim($\Sigma_\sigma$) $\leq d + 2$ [22], therefore Theorem 2.1 implies that for any family $\mathcal{D}$ of distributions defined on $\mathbb{R}^d$ and for any $\epsilon > 0$, the selectivity functions are $\epsilon$-learnable with training set of size $\tilde{O}\left(\frac{1}{\epsilon^d}\right)$.

**Semi-algebraic Range Queries.** A very general class of range queries is the so-called *semi-algebraic range query*. A $d$-dimensional semi-algebraic set is subset of $\mathbb{R}^d$ defined by a Boolean formula over polynomial inequalities. For example, $R = \{(x,y) \in \mathbb{R}^2 : (x^2 + y^2 \leq 4) \wedge (x^2 + y^2 \geq 1) \wedge (y - 2x^2 \leq 0)\}$ is a semi-algebraic sets; see Figure 3. All the three above examples are special cases of semi-algebraic range queries. Let $\mathcal{T}_{d,b,\lambda}$ be the set of all semi-algebraic sets defined by at most $b$ $d$-variate polynomial inequalities, each of degree at most $\lambda$. It is known that VC-dim($\mathcal{T}_{d,b,\lambda}$) $\leq \lambda$ $\cdot$ $\lambda(d,b,\Delta)$ [9]. Hence the selectivity functions on $(\mathbb{R}^d, \mathcal{T}_{d,b,\lambda})$ are also learnable for any constants $d, b, \Delta$.

Semi-algebraic sets enable us to handle range spaces in which $X$ is not a set of points in $\mathbb{R}^d$. For example, let $\mathcal{B}$ be the set of all discs in $\mathbb{R}^2$. For a query disc $B$, let $\mathcal{R}_B \subseteq \mathcal{B}$ be the set of discs that intersect $B$; see Figure 3. Define $\mathcal{R}_* = \{R_B : B \in \mathcal{B}\}$, and consider the range space $\Sigma_* = (\mathcal{B}, \mathcal{R}_*)$. We can map each disc in $\mathcal{B}$ to a point $(x, y, z)$ in $\mathbb{R}^3$ where $(x, y)$ is the center of the disc and $z$ is its radius. Then for a query disc $B$ centered at $(c_x, c_y)$ and radius $r$, the range $\mathcal{R}_B$ maps to the set

$$\gamma_B = \{(x, y, z) \in \mathbb{R}^3 : (x - c_x)^2 + (y - c_y)^2 \leq (r + z)^2, z \geq 0\}.$$ 

Set $\mathcal{R}_{\geq 0} = \mathbb{R}^2 \times \mathbb{R}_{\geq 0}$ and $\mathcal{R}_* = \{(x, y, z) \in \mathbb{R}^3 : B \in \mathcal{B}\}$. Then $\Sigma_*$ is mapped to $(\mathcal{R}_{\geq 0}, \mathcal{R}_*)$. Since ranges in $\mathcal{R}$ are semi-algebraic sets with $b = 1$ and $\Delta = 2$, VC-dim($\mathcal{R}_{\geq 0}, \mathcal{R}_*$) is finite and hence selectivity functions on $(\mathcal{B}, \mathcal{R}_*)$ are learnable.

We conclude this discussion by giving an example of range space for which selectivity functions are not learnable.

**Polygon range queries with arbitrary number of vertices.** Let $\mathcal{C}$ be the set of all convex polygons in $\mathbb{R}^2$ with arbitrary number of vertices. Consider the range space $\Sigma = (\mathbb{R}^2, \mathcal{C})$. It is known that

$$\text{VC-dim}(\Sigma) = \infty$$

therefore Theorem 2.1 implies that selectivity functions on $\Sigma$ are not learnable.

### 2.3 Proof of Theorem 2.1

We prove Theorem 2.1 using the notion of *fat-shattering dimension* introduced by Kearns and Schapire [21], which is a generalization of VC-dimension, and the results by Alon et al. [5] and Bartlett-Long [7] (see also [8]). As in Section 2.1, let $\mathcal{H}$ be a class of functions from a domain $X$ into $[0, 1]$. Let $\gamma \in (0, 1/2)$ be a parameter. We say that $\mathcal{H}$ $\gamma$-shatters a subset $V \subseteq X$ if there is a witness function $\sigma : V \rightarrow [0, 1]$ such that for every subset $E \subseteq V$, there is a function $H_E \in \mathcal{H}$ with

$$H_E(x) \geq \sigma(x) + \gamma, \quad \forall x \in E,$$

$$H_E(x) \leq \sigma(x) - \gamma, \quad \forall x \in V \setminus E.$$

An example is shown in Figure 4.

The *fat-shattering dimension* of $\mathcal{H}$, denoted by $\text{fat}_\gamma(\mathcal{H})$, is the size of the largest subset of $X$ that can be $\gamma$-shattered by $\mathcal{H}$. If subsets of unbounded finite size can be $\gamma$-shattered by $\mathcal{H}$, then we set $\text{fat}_\gamma(\mathcal{H}) = \infty$. Note that if $\mathcal{H}$ is a class of functions from $X$ into $[0, 1]$, then $\gamma$-fat-shattering dimension is the same as VC-dimension. An advantage of $\gamma$-fat-shattering dimension is that it is sensitive to the scale at which difference in the function values are considered important. Alon et al. [5] proved that if $\text{fat}_\gamma(\mathcal{H})$ is finite, where $c \in (0, 1)$ is a suitable constant, then $\mathcal{H}$ is $\epsilon$-learnable. The bound on the size of the training set was improved by Bartlett and Long [7]. In particular, their result implies that $\mathcal{H}$ is $\epsilon$-learnable with training set size

$$n_0(\epsilon, \delta) = \mathcal{O}\left(\frac{1}{\epsilon^2} \left(\text{fat}_\gamma\left(\frac{c}{\epsilon}\right) \log^2 \frac{1}{\epsilon} + \log \frac{1}{\delta}\right)\right).$$

Returning to the selectivity functions, let $\Sigma = (X, \mathcal{R})$ be a range space, let $\mathcal{D}$ be a family of probability distributions on $X$ and $\gamma \in (0, 1)$. Set $\mathcal{S} = \mathcal{S}(\mathcal{D})$ to be the selectivity functions defined by $\mathcal{D}$. Our main technical result is that if VC-dim($\Sigma$) $= \lambda$, for some constant $\lambda$, then $\text{fat}_{\lambda}(\mathcal{S}) = \tilde{O}\left(\frac{1}{\lambda^{\gamma}}\right)$. By plugging this result into the results of [5, 7], we prove the first part of Theorem 2.1.

Let $\mathcal{T} \subseteq \mathcal{R}$ be a subset $\gamma$-shattered by $\mathcal{S}$. To bound $\text{fat}_{\gamma}(\mathcal{S})$, it suffices to prove that $|\mathcal{T}| = \tilde{O}\left(\frac{1}{\lambda^{\gamma}}\right)$. First, we partition the ranges

---

**Figure 3:** (Left) Semi-algebraic range of $R = \{(x,y) \in \mathbb{R}^2 : (x^2 + y^2 \leq 4) \wedge (x^2 + y^2 \geq 1) \wedge (y - 2x^2 \leq 0)\}$. (Right) A disc-intersection query, discs intersected by the query disc (red) are shown in blue.

**Figure 4:** $x_1, x_2$ are $\gamma$-shattered by linear functions. We choose $H$ to be the linear function whose bit sequence $b_2b_1$ corresponds to $E$ (i.e., $b_1 = 1$ if $x_1 \in E$).
in $T$ based on the values of their respective witnesses $\sigma(R)$:

$$T_j = \{ R \in T : \sigma(R) \in [(j-1) \cdot \gamma, j \cdot \gamma), \text{for } j \in [1/\gamma]\}.$$  

Lemma 2.2. Suppose Equation (2) is realized for some subset $E \subseteq T_j$ by $s_D$ for some distribution $D \in \mathcal{D}$. Then, for any pair $R \in E, R' \in T_j \setminus E$, we have

$$s_D(R) - s_D(R') > \gamma. \quad (3)$$

Proof. By Equation (2), we have

$$s_D(R) \geq \sigma(R) + \gamma \quad \text{and} \quad -s_D(R') \geq -\sigma(R') + \gamma$$

Adding these, we get

$$s_D(R) - s_D(R') + (\sigma(R') - \sigma(R)) \geq 2\gamma. \quad (4)$$

Since $R, R' \in T_j$ for some $j \in [1/\gamma]$, we have $\sigma(R') - \sigma(R) < \gamma$. The lemma follows by using this inequality in Equation (4). \hspace{1cm} \Box

Now, consider any fixed ordering $\pi = (R_1, R_2, \ldots, R_k)$ of the ranges in $T_j$, where $k = |T_j|$. Let us also fix the subset:

$$E = \{ R_i | 1 \leq i \leq \lceil k/2 \rceil \} \quad (5)$$

to be the set of ranges with even index in $\pi$. We say that an object $x \in X$ crosses a pair of ranges $R, R'$ if $x \in R \cap R'$, where $\cap$ is the symmetric difference (see Figure 1 for $R_1 \cap R_2$). For $1 \leq i < k$ and for every $x \in X$, we define an indicator random variable as follows:

$$I_{i,x} = \begin{cases} 1 & \text{if } x \in R_i \cup R_{i+1}, \\ 0 & \text{otherwise}, \end{cases}$$

and let $I_x = \sum_{i=1}^{k-1} I_{i,x}$.

Since $T$ is $\gamma$-shattered by $\mathcal{S}$, there is a distribution $D_\pi \in \mathcal{D}$ that satisfies (2) for $E$. The next lemma is a direct consequence of Lemma 2.2, by summing up over the pairs of ranges $R_i, R_{i+1}$ for even $i \in T_j$:

Lemma 2.3. $\mathbb{E}_{x \sim D_\pi} [I_x] > \gamma (k - 1)$.

Proof. By Lemma 2.2, $\mathbb{E}_{x \sim D_\pi} [I_{i,x}] \geq \gamma$ for any index $i$ since exactly one of $R_i, R_{i+1}$ belongs to $E$. The lemma now follows by using linearity of expectation. \hspace{1cm} \Box

The lower bound on $\mathbb{E}_{x \sim D_\pi} [I_x]$ in Lemma 2.3 holds for any ordering $\pi$ of the ranges in $T_j$; the distribution $D_\pi$ obviously depends on $\pi$. We now complement this lower bound with an upper bound on $\mathbb{E}_{x \sim D_\pi} [I_x]$ for a specific ordering $\pi$ of $T_j$.

Lemma 2.4. There is an ordering $R_1, R_2, \ldots, R_k$ of the ranges in $T_j$ such that for any distribution $D$ defined on $X$, we have

$$\mathbb{E}_{x \in D} [I_x] = O\left(k^{1-1/\lambda} \log k\right),$$

where $\lambda = \text{VC-dim}(X, \mathcal{R}).$

Proof. Let $\mathcal{X} = (X, T_j)$ be the range space defined by the ranges in $T_j$. Note that $\text{VC-dim}(\mathcal{X}) \leq \text{VC-dim}(\Sigma) = \lambda$. Consider the dual range space $\mathcal{X}^*$ of $\mathcal{X}$, where $\mathcal{X}^* = (T_j, \{ R_x = \{ R \in T_j : x \in R \} \mid x \in X \})$, i.e., the objects of $\mathcal{X}^*$ are the ranges of $T_j$ and for each object $x \in X$, we have a dual range in $\mathcal{X}^*$ consisting of ranges of $\Sigma$ that contain $x$. Note that $\Sigma^* = \mathcal{X}^*$.

Note that although $\sigma(R) = 1$ is excluded by this definition if $1/\gamma$ is an integer, it is a well-defined partition since $\sigma(R)$ cannot be equal to 1 for any range $R \in T$. This follows from the observation that if $\sigma(R) = 1$, then Equation (2) cannot be satisfied for $R \in E$ since $s_D(R) \leq 1$ and $\gamma > 0$.

We compute the desired ordering of $T_j$, using the following results by Chazelle and Welzl [12]: Let $\Sigma = (V, \Gamma)$ be a finite range space with $|V| = m$. We say that a range $y \in \Gamma$ crosses a pair $v_i, v_j \in V$ if $y \cap \{v_i, v_j\} = \emptyset$. The result in [12] (Theorem 4.3) proves that there is an ordering $v_1, v_2, \ldots, v_m$ of objects in $V$ such that any range in $\Gamma \cap \Sigma^*$ crosses $O\left(\frac{m^{1-1/\lambda} \log m}{\gamma}\right)$ pairs $(v_i, v_{i+1})$ for $1 \leq i < m$, where $\lambda^*$ is the VC-dimension of the dual range space of $\Sigma$. Applying this result to $\Sigma^*$ and using the fact that $\Sigma^* = \mathcal{X}$, we obtain an ordering $R_1, R_2, \ldots, R_k$ of $T_j$ such that any range of $\Sigma^*$ crosses $O\left(\frac{k^{1-1/\lambda} \log k}{\gamma}\right)$ pairs $(R_i, R_{i+1})$. By the definition, a range $R_x$ crosses $R_i, R_{i+1}$ if $|R_x \cap \{R_i, R_{i+1}\}| = 1$, which is equivalent to saying that $x \in R_i \cap R_{i+1}$. Hence, for any $x \in X$, there are $O\left(\frac{k^{1-1/\lambda} \log k}{\gamma}\right)$ pairs $(R_i, R_{i+1})$ crossed by $x$. Since this bound holds for every $x \in X$, we conclude that

$$\mathbb{E}_{x \sim \mathcal{D}} [I_x] = O\left(\frac{k^{1-1/\lambda} \log k}{\gamma}\right).$$

We are now ready to bound the size of $T_j$.

Lemma 2.5. For any $j \in [1/\gamma]$, $|T_j| = O\left(\left(\frac{\gamma}{\log \gamma}\right)^{1/\lambda}\right)$.

Proof. Plugging Lemmas 2.4 and 2.3 together, we conclude there exists a constant $c$ such that

$$\gamma \cdot (k - 1) \leq c \cdot k^{1-1/\lambda} \log k,$$

which implies that

$$\frac{k^{1/\lambda}}{\log k} \leq 2c/\gamma, \text{ or } k = O\left(\frac{1}{2} \log \frac{1}{\gamma}\right).$$

Summing this bound over all $j \in [1/\gamma]$, we conclude that $|T| = O\left(\frac{1}{\gamma^{1/\lambda}}\right)$. Hence, the size of any set of query ranges in $\mathcal{R}$ that can be $\gamma$-shattered by $\mathcal{S}$ is $O\left(\frac{1}{\gamma^{1/\lambda}}\right)$, which implies the main technical result of this section.

Lemma 2.6. Let $\Sigma = (X, \mathcal{R})$ be a range space with $\text{VC-dim}(\Sigma) = \lambda$, let $\mathcal{D}$ be a family of probability distribution over $X$, and let $\mathcal{S} := \mathcal{S}_{\Sigma, \mathcal{D}}$ be the family of selectivity functions on $\Sigma$ by $\mathcal{D}$. For any $\gamma \in (0, 1)$, the $\gamma$-fat shattering dimension of $\mathcal{S}$ is $\tilde{O}\left(\frac{1}{\gamma^{1/\lambda}}\right)$.

Finally, plugging Lemma 2.6 into the results of Alon et al. [5] and Bartlett-Long [7], we obtain the first part of Theorem 2.1.

We next turn to the second part of Theorem 2.1. As in Section 2.1, let $\mathcal{H}$ be a class of functions from a domain $X$ into $[0, 1]$. Let $\gamma \in [0, 1]$ be a parameter. Alon et al. [5] proved that if $\text{fat}_\mathcal{H}(\epsilon) = \infty$, then $\mathcal{H}$ is not $(\epsilon^2/8 - \tau)$-learnable for any $\tau > 0$. Returning to the selectivity functions $\mathcal{S} := \mathcal{S}_{\Sigma, \mathcal{D}}$ defined on the range space $\Sigma = (X, \mathcal{R})$ and a family of probability distribution $X \times \mathcal{D}$. Our second technical result is that if $\text{VC-dim}(\Sigma) = \infty$, then $\text{fat}_\mathcal{S}(\gamma) = \infty$ for any $\gamma \in (0, 1/2)$.

Lemma 2.7. Let $\Sigma = (X, \mathcal{R})$ be a range space, let $\mathcal{D}$ be a family of probability distribution over $X$, and let $\mathcal{S} := \mathcal{S}_{\Sigma, \mathcal{D}}$ be the family of selectivity functions on $\Sigma$ by $\mathcal{D}$. If $\text{VC-dim}(\Sigma) = \infty$, the $\gamma$-fat shattering dimension of $\mathcal{S}$ is also $\infty$, for any $\gamma \in (0, 1/2)$.

\footnote{For $\lambda^* = 1$, the original paper [12] proves a slightly weaker bound of $O(\log^2 m)$ on the number of pairs crossed by a range. Using an improved bound on $\epsilon$-nets for range spaces of VC-dimension 1 (see e.g. [55], Chapter 15), the bound can be improved to $O(\log m)$.}
Proof. Consider the dual range space $\Sigma^*$ of $\Sigma$, where $\Sigma^* = (\mathcal{R}, \Gamma)$ where $\Gamma = \{ \mathcal{R}_x = \{ R \in \mathcal{R} : x \in R \} | x \in X \}$ as defined in the proof of Lemma 2.4. As shown in [12], since $\text{VC-dim}(\Sigma^*) = \infty$, we have $\text{VC-dim}(\Sigma') = \infty$. In other words, for any integer $k > 0$, there exists a subset $\mathcal{R}_k \subseteq \mathcal{R}$ of $k$ ranges shattered by $\Gamma$, i.e., for every subset $E \subseteq \mathcal{R}_k$, there is a point $x \in X$ such that $x \in E$ if $x \in E \cap \mathcal{R}_k \setminus \mathcal{R}_i$ for all $R_i \in E$ and $x \notin E$ for all $R_i \in \mathcal{R}_k \setminus E$.

Next, we show that $\mathcal{R}_k$ is $\gamma$-shattered by $\Sigma$. Set $\sigma(R_i) = 1/2$ for all $R_i \in \mathcal{R}_k$. Consider any subset $E \subseteq \mathcal{R}_k$. We choose $D \in \mathcal{D}$ as a delta function, which is 1 at $x_0$ and 0 everywhere else. The corresponding selectivity function $s_D(S)$ has $s_D(R_i) = 1$ if $x \in R_i$ and 0 otherwise, which realizes Equation (2) for any $\gamma \leq 1/2$. Hence for any $k > 0$, there always exists a subset $\mathcal{R}_k \subseteq \mathcal{R}$ of size $k$ that can be $\gamma$-shattered by $\Sigma$ for any $\gamma \in (0, 1/2)$, i.e., the $\gamma$-fat shattering dimension of $\Sigma$ is $\infty$. An example is illustrated in Figure 5. □

The above lemma proves second part of Theorem 2.1, thereby completing the proof of Theorem 2.1.

3 LEARNING ALGORITHM
Recall that Theorem 2.1 gives an upper bound on the size of training samples, but the definition of $\epsilon$-learnability assumes the existence of a learning procedure that for a given a finite training sample $z^n = (z_1, z_2, \cdots, z_n)$ where $z_i = (R_i, s_i) \in \mathcal{R} \times [0, 1]$, and a family $\mathcal{D}$ of data distributions, computes a distribution $D \in \mathcal{D}$ such that $s_D$ minimizes the expected loss function, i.e., returns $\min_{D \in \mathcal{D}} \frac{1}{n} \sum_{i=1}^{n} (s_D(R_i) - s_i)^2$. In this section, we describe algorithms for computing such a distribution. For simplicity, we focus on selectivity queries discussed in the introduction, namely orthogonal range, linear inequality, and distance-based queries, though our algorithm works for a much larger class of queries such as semi-algebraic range queries. The aim of this section is to describe simple, generic approaches, and we do not attempt to optimize the learning procedure for specific selectivity queries.

3.1 A Generic Procedure
We focus on two families of distributions, histograms and discrete distributions. In the former, a distribution is a piecewise-constant function, i.e., $D = \{(B_1, w_1), \cdots, (B_m, w_m)\}$, where $\sum_{i=1}^{m} w_i = 1$. $D$ has uniform density $\frac{\text{Vol}(B_i)}{\text{Vol}(B_i)}$ over each bucket $B_i$, where $\text{Vol}(B_i)$ is the volume of $B_i$, and each $B_i \subseteq \mathbb{R}^d$ is a simple region of constant complexity homomorphic to a ball (e.g., boxes, simplices, etc), also called Tarski cells [48]. $B_i$’s are pairwise disjoint and partition $\mathbb{R}^d$.

For a query range $R$, $s_D(R)$ is defined as

$$s_D(R) = \sum_{j=1}^{m} \frac{\text{Vol}(B_i \cap R)}{\text{Vol}(B_i)} \cdot w_j \quad (6)$$

Intuitively, $\frac{\text{Vol}(B_i \cap R)}{\text{Vol}(B_i)}$ computes the fraction of the bucket $B_i$ that intersects with the query region $R$. Note that we do not make any assumption on the ranges, which can be bounded or unbounded. Multiplying this fraction by $w_i$, $s_D(R)$ in essence makes the simple assumption that the data points within each cell are distributed uniformly. We note that when the range $R$ can be represented with a simple function, such as an orthogonal range, a halfspace or a ball, the volume of $R$ and its intersection with a bucket (as hyper-rectangle) can be easily computed exactly. In general, the volume of a complex range can be estimated via MCMC sampling [15].

A discrete distribution has a similar form $D = \{(B_1, w_1), \cdots, (B_m, w_m)\}$, but $B_i$’s are a set of $m$ points, which we also call buckets, in $\mathbb{R}^d$. As before $\sum_{i=1}^{m} w_i = 1$. For a query range $R$, $s_D(R)$ is defined as

$$s_D(R) = \sum_{i=1}^{k} w_i \cdot 1 \{ B_i \in R \} \quad (7)$$

In both cases, the algorithm computes $D$ in two phases. The first phase, called bucket-selection, constructs the set $\mathcal{B} = \{ B_1, B_2, \cdots, B_m \}$ of buckets. The second phase, called weight-estimation, computes the weight $w_i$ for each bucket $B_i$.

Bucket design. Let $\{ R_1, R_2, \cdots, R_n \}$ be the set of ranges in the training set $z^n$; here we treat each range as a geometric region defined by the query predicate (e.g. rectangles for orthogonal range queries, halfspaces for linear-inequality queries, balls for distance-based queries) rather than a subset of input objects. The arrangement of $\{ R_1, R_2, \cdots, R_n \}$ is the partition of $\mathbb{R}^d$ into maximal connected regions so that each region lies in the same subset of ranges of $\{ R_1, R_2, \cdots, R_n \}$. We further refine each region into small regions, called cells, so that each cell has constant complexity (i.e., constant number of vertices, edges, and faces that only depends on $d$) and its boundary is connected. It is known that such a decomposition of size $O(n^d)$ can be computed in $O(n^d \log n)$ time [4]. We choose $\mathcal{B}$, the set of buckets, to be the resulting set of cells. If we wish to construct a discrete distribution, we simply choose a random point in each cell, and these points form the bucket set $\mathcal{B}$.

Weight estimation. Let $\mathcal{B}$ be the set of buckets constructed in the previous phase. To estimate the weights, we set them as variables and solve the following convex quadratic programming:

$$\text{minimize } \sum_{i=1}^{n} (s_D(R_i) - s_i)^2$$

subject to $\sum_{j=1}^{m} w_j = 1$, $0 \leq w_j \leq 1$, $j \in \{1, 2, \ldots, k\}$.

where $s_D(R_i)$ is the function specified in Equation (6) and (7) for histograms and discrete distributions respectively. We solve this problem using open-sourced non-negative least squares solver [1].

The proof of the following lemma is given in the full version [3].

Lemma 3.1. The above algorithm constructs a histogram (resp. discrete distribution) that minimizes the loss function over all histograms (resp. discrete distributions).
We construct a histogram \( \tau_z \). The intractability of a number of related problems [33] suggests the problem at hand is also NP-Hard, and we leave it as an interesting direction of future research. In the next two subsections, we describe simple, efficient algorithms for constructing a histogram and a discrete distribution. The weight-estimation phase remains the same, so we focus on the bucket-design phase.

3.2 Histogram

We construct a histogram \textsc{QuadHist}, intended for low-dimensional data and queries. For simplicity, we assume finite lower and upper bounds on the range of values for each dimension. Regardless of the query class—orthogonal range, linear inequality, or distance-based queries—\textsc{QuadHist}’s buckets are a disjoint set of orthogonal ranges coming from the partitioning of \( D \) by a quadtree. The construction of the quadtree is guided by both the geometry of training queries and their selectivities, such that the resulting partitioning of the data space is finer in parts where queries and data are denser.

![Figure 6: Bucket refinement for QuadHist. The blue rectangle is a training range \( R \) with selectivity 0.2 and the underlying grid is the quadtree leaves. Splitting is recursively applied to each node until the density of its intersection with \( R \) is estimated no greater than \( \tau = 0.026 \). The left is the quadtree before processing \( R \), and the right is the quadtree after.](image)

Let \( z^n = (z_1, z_2, \cdots, z_n) \) be the training set with \( z_i = (R_i, s_i) \). We construct a quadtree on the ranges \( R_1, R_2, \cdots, R_n \) in the training set \( z^n \) as follows. We start with a single-node quadtree corresponding to a single bucket spanning the whole data space. We process each \( z_i = (R_i, s_i) \) to refine (if needed) the buckets as follows. For each leaf node \( B \) of the quadtree (interpreting \( B \) as a range), we compute \( \text{Vol}(B \cap R_i) \cdot s_i \). In the same spirit as \( s_B(R) \), this quantity estimates the fraction (out of all data points) of the data points in \( R \) that are also in \( B \). We compare this estimate with a predetermined threshold \( \tau \in (0, 1) \). If the estimate is higher than \( \tau \) (informally, \( B \) carries “too much” density), we split the quadtree leaf \( B \) into \( 2d \) children and recursively apply the procedure on them. See Figure 6 for an illustration. After going through all training queries, we take all leaves of the final quadtree to be our QuadHist buckets (and proceed to the weight-assignment phase). We can control the model size \( k \) by varying the parameter \( \tau \) or adding a hard termination condition on the number of leaves in the splitting procedure.

\textbf{Remarks}. Several points are worth noting here (details are in the full version [3]). Considering the selectivities in bucket design protects us from devoting more buckets than necessary to regions where data is sparse (although the weight estimation step utilizes selectivities, the buckets would have been chosen already).

Second, the simplicity of quadtree-guided bucket design procedure gives rise to an interesting and desirable property of stability: given a training workload, the resulting collection of buckets is always the same regardless of the ordering in which we process the workload. This property is unfortunately missing for many complex selectivity estimation schemes with more bells and whistles. Combining the stability of bucket design with the determinism of weight estimation, we know that \textsc{QuadHist} trained on the same query workload would always behave consistently.

Third, the quadtree doubles up as a convenient data structure for speeding up the bucket design step of the training process. For example, the \( \tau \)-based splitting procedure can piggyback on the efficient and generic quadtree procedure for answering \( R \) as range query, regardless of \( R \)’s shape.

3.3 Discrete Distribution

We present a discrete distribution \textsc{PtsHist} as an alternative instantiation of our generic method for high dimensions. \textsc{QuadHist} is not expected to perform well in high dimensions because of the well-known challenges: 1) rectangles are poor representations of high-dimensional data distributions, and 2) computing volumes of intersections between orthogonal ranges and other types of query ranges (e.g., balls) is difficult. Hence, \textsc{PtsHist} turns to using a collection of \textit{points} in the data space (as opposed to ranges) as buckets.

Given a target model size \( k \), we take the following two steps to generate the points representing buckets. 1) We draw 0.9\( k \) points from the \textit{interior} of all training query ranges. More specifically, for each \( z_i = (R_i, s_i) \in z^n \), we draw \( s_i / \sum_{j=1}^{k} s_j \cdot (0.9k) \) points uniformly at random from the range defined by \( R \). In other words, each \( R \) receives a “share” of points proportional to its selectivity. 2) We then draw the remaining 0.1\( k \) uniformly at random from the whole space. This step essentially makes it possible to allocate some density to regions not covered by the training queries.

Although sampling from the interior of geometric objects in high dimensions has its own challenges, it is a well-studied problem for specific shapes such as hyperrectangles, halfspaces, and balls. Our sampling implementation in Section 4 in fact uses straightforward rejection sampling from the smallest bounding box [42] of \( R \) (see our full version [3] for details), and we have found the generic approach to offer adequate performance in practice.

\textbf{Remarks}. The sampling procedure used by \textsc{PtsHist} does not guarantee an unbiased sample from any data distribution \( D \)—but that is not our goal of this procedure in the first place. Instead, we only aim to generate a number of points whose positions serve as buckets; the subsequent (generic) weight estimation step ensures the consistency between the \textsc{PtsHist} model and the training workload.

Figure 7 illustrates the real data distribution, the histogram built by \textsc{QuadHist}, and the discrete distribution built by \textsc{PtsHist} over the Power dataset (see more details in Section 4).
4 EXPERIMENTS

In this section, we implement QuadHist and PtsHist in Section 3, empirically evaluate their performance on real-world datasets and, when applicable, compare them against state-of-the-art solutions (for orthogonal range queries). As mentioned in Section 3, they are not intended to “beat” state-of-the-art solutions; rather, they are simple, generic implementations so that our experiments can focus on illustrating the power of our theoretical results instead of the artifacts of additional features. We implemented all our algorithms in Python and ran all our experiments on a server with 8 Intel Core i7-9700 CPUs (3.00GHz). All codes are public at [2].

Datasets [13]. We use real-world datasets adopted by a recent benchmark paper [46] for evaluation:

- Power contains electric power measurement gathered from a house over 47 months, with 2.1M tuples over 7 attributes.
- Forest contains forest cover type data, with 581k tuples over 10 numerical attributes. It is named as CoverType in [13].
- Census contains the basic population characteristics, with 49K tuples over 13 attributes (8 categorical and 5 numerical).
- DMV contains the vehicle registration records of NYC, with 11M tuples over 11 attributes (10 categorical and 1 numerical).

As datasets have multiple attributes, we usually choose a subset of attributes randomly and project the tuples on the chosen attributes. For simplicity, we normalize the domain of each attribute into [0, 1].

Workloads. We consider orthogonal range, halfspace, and ball queries. For orthogonal range queries, we generate three different synthetic workloads. Each orthogonal range query \( R \) can be represented by a center point and \( d \) side lengths (one per dimension). After fixing the center point, we sample each side length independently and uniformly from [0, 1]. Depending on the distribution of centers points, we distinguish three workloads (see Figure 8):

- Data-driven: uniformly sampling from the underlying dataset.
- Random: uniformly sampling from the \( d \)-dimensional unit cube.
- Gaussian: uniformly sampling from a \( d \)-dimensional Gaussian distribution (the mean and variance for each dimension of the Gaussian distribution is set as 0.5 and 0.167).

The Data-driven workload is arguably more realistic as queries typically “follow” the underlying data distribution, but we also want to evaluate on Random and Gaussian, which are independent from the underlying data. We will only generate equality predicates for categorical attributes; hence the width is zero in this case.

Workloads for other query types are generated in an analogous fashion. For ball queries, once we pick the center point, we then sample the ball radius uniformly from [0, 1]. For halfspace queries, once we pick the center point (lying on the boundary plane of the halfspace), we then randomly pick a \( d \)-dimensional unit vector (normal to that plane) that defines the orientation of the halfspace.

Unless explicitly noted otherwise, the set of training and test queries for each experiment are sampled uniformly and independently from the same query workload. Note that there could be very few queries in the overlapping, as well as the subset of predicates.

Methods Compared. For fair comparison, we restrict ourselves to methods that only have access to query workload, but not the underlying data. Since this paper is concerned with learned selectivity estimation models that can provide provable guarantees, we also do not include methods based on deep learning that may return models that do not correspond to any valid hypothesis, and consequently, have been observed to produce selectivity estimates that are not monotone or consistent [46]. For orthogonal range queries, based on the recent empirical study on cardinality estimation in [46] (see Section 5 for more details), Isomer [39] produces the best accuracy and QuickSel [36] achieves the best tradeoff between accuracy and efficiency, so we include both in our comparison with QuadHist and PtsHist. For halfspace and ball queries, there are no obvious candidates for comparison with our methods, as traditional histogram-based methods have not focused on these queries.

Error Measures. We adopt two common error measures for evaluating selectivity estimators. For a test query \( R \), let \( \hat{s}(R) \) and \( s(R) \) be the estimated and true selectivities of \( R \), respectively, and let \( n \) be the number of test queries.

- Root Mean Square (RMS) Error \( = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (\hat{s}(R_i) - s(R_i))^2} \).

- Q-Error(p) \( = p \)-th quantile of \( \left[ \frac{\max\{\hat{s}(R_i), s(R_i)\}}{\min\{\hat{s}(R_i), s(R_i)\}} : i \in [n] \right] \).

Q-error is a good complement of RMS error because Q-error is better at capturing errors that are small in absolute terms but large in relative terms, which occur frequently since many database queries tend to be selective. \( L_{\infty} \) error is used for investigating different objective functions in model training.
We start with selectivity estimation for orthogonal range queries, QuickSel we adopt with 200 vary PtsHist 4 QuadHist requires per bucket than for Data-driven query workload over Power data. As we have seen, 10, 11, and 12. Again, we consider 2D orthogonal range queries it is intended for higher dimensions). The results are shown in Fig- son. For completeness we also compare with our state-of-the-art competitors, practically acceptable accuracy with RMS error smaller than consistent with Theorem 2.1's prediction. The good news is that toward the origin), although the rate of decrease also diminishes, of training queries, error decreases (the series of line plots push buckets for merely 2000 Power data. We vary the number of training queries from 50 to 2000; each line plot corresponds to a specifically sized training set. Since the accuracy of QuadHist also depends on its model com- plexity (number of buckets), given a specific training set, we further vary r in the bucket design step (Section 3.2) to adjust the model complexity. As we can see from Figure 9, error generally decreases with model complexity, although it eventually flattens out, and in one case even trends up because of overfitting (when using 10000 buckets for merely 50 training queries). As we increase the number of training queries, error decreases (the series of line plots push toward the origin), although the rate of decrease also diminishes, consistent with Theorem 2.1’s prediction. The good news is that with 200 training queries and 500 buckets, QuadHist already offers practically acceptable accuracy with RMS error smaller than 0.02.

To help put our results in a boarder context, we next bring other state-of-the-art competitors, Isomer and QuickSel, into compari- son. For completeness we also compare with our PtsHist (although it is intended for higher dimensions). The results are shown in Figures 10, 11, and 12. Again, we consider 2D orthogonal range queries for Data-driven query workload over Power data. As we have seen, model complexity can affect prediction accuracy, so for comparison, we adopt QuickSel’s convention of using number of buckets 4× the number of training queries, for both QuadHist and PtsHist. For Isomer, it is difficult to enforce this convention, so we let it choose the number of buckets by itself; in our results, Isomer ends up using number of buckets 48-160× the number of training queries.

As we vary the number of the training queries, Figure 10 shows the actual number of buckets used by different models; Figure 11 shows their prediction accuracy; and Figure 12 shows their training time. All models become more accurate when more queries are used for training. While Isomer is the most accurate, it uses more buck- ets and is much slower than others: it could not finish training in 30 minutes with 500 training queries, so the figures do not show ISO- mer for larger training workloads. On the other hand, QuadHist, PtsHist, and QuickSel are comparable on all fronts and work very well: their training scales to large training sets, and they are able to deliver RMS error lower than 0.01 with 1000 training queries. An encouraging observation is that despite the simple, generic nature of QuadHist and PtsHist, they are able to match (if not beat) the performance of the state-of-the-art methods. Such competitiveness can only be attributed to the fundamental learnability of the task.

Finally, another important practical consideration is prediction time. Note that all these models work similarly in terms of predic- tion. QuickSel, QuadHist, and Isomer have the same estimation procedure that involves computing intersections between orthogonal ranges, while PtsHist involves checking whether an orthogonal range contains a point. Therefore, their prediction time is dictated by their model complexity and already reflected in Figure 10; we do not separately report the prediction time for these different models.

4.1 Learnability with Enough Training Samples

We start with selectivity estimation for orthogonal range queries, which has been studied extensively, and validate the learnability of selectivity functions. Recall that Theorem 2.1 shows a clear de- pendency between error ε and training size m: we now explore this dependency empirically. We first take a closer look at the 2D case, where QuadHist is our generic implementation. Figure 9 shows the QuadHist results for Data-driven query workload over Power data. We vary the number of training queries from 50 to 2000; each line plot corresponds to a specifically sized training set. Since the accuracy of QuadHist also depends on its model com- plexity (number of buckets), given a specific training set, we further vary r in the bucket design step (Section 3.2) to adjust the model complexity. As we can see from Figure 9, error generally decreases with model complexity, although it eventually flattens out, and in one case even trends up because of overfitting (when using 10000 buckets for merely 50 training queries). As we increase the number of training queries, error decreases (the series of line plots push toward the origin), although the rate of decrease also diminishes, consistent with Theorem 2.1’s prediction. The good news is that with 200 training queries and 500 buckets, QuadHist already offers practically acceptable accuracy with RMS error smaller than 0.02.

To help put our results in a boarder context, we next bring other state-of-the-art competitors, Isomer and QuickSel, into compari- son. For completeness we also compare with our PtsHist (although it is intended for higher dimensions). The results are shown in Figures 10, 11, and 12. Again, we consider 2D orthogonal range queries for Data-driven query workload over Power data. As we have seen, model complexity can affect prediction accuracy, so for comparison, we adopt QuickSel’s convention of using number of buckets 4× the number of training queries, for both QuadHist and PtsHist. For Isomer, it is difficult to enforce this convention, so we let it choose

Outline. We next investigate the following questions on the learned model to verify our learning theory developed in Section 2:

• Section 4.1: How does the training size affect the learned model?
• Section 4.2: What is the performance of learned model if the query workload is not correlated with the underlying data distribution?
• Section 4.3: What is the performance of learned model if training and testing query distributions do not match?
• Section 4.4: How does the dimensionality of data space affect the learned model?
• Section 4.5: How does the query type affect the learned model?

4.1 Learnability with Enough Training Samples

We next investigate the following questions on the learned model to verify our learning theory developed in Section 2:

• Section 4.1: How does the training size affect the learned model?
• Section 4.2: What is the performance of learned model if the query workload is not correlated with the underlying data distribution?
• Section 4.3: What is the performance of learned model if training and testing query distributions do not match?
• Section 4.4: How does the dimensionality of data space affect the learned model?
• Section 4.5: How does the query type affect the learned model?

4.2 Data vs. Query Distributions

So far, we have only presented results on Power data. We have also repeated these experiments in the last section on other datasets and made similar observations. Due to space constraints, we present our complete results in the full version [3].

A second question is how the query distribution—in connection with the underlying data distribution—may affect accuracy. We note that one strength of our theory in Section 2 is that they hold for arbitrary data distribution and arbitrary query distribution—even if they are highly skewed. Indeed, most real-world datasets exhibit skewness, including for example Power, and we have seen that our approach indeed works well on them. However, the query workload used so far was Data-driven. A valid question arises: what if the query workload is not correlated with the underlying data distribution? According to our theory, we should still do fine, but let us validate it empirically with other query distributions Random and Gaussian, which are independent from the underlying data distribution. Figures 13 and 15 show the prediction errors of differ- ent models, using the same exact setup on Power data as Figure 11 earlier, except we replace the Data-driven query workload with Random and Gaussian, respectively. We also report the prediction errors of non-empty queries in Random workload in Figure 14, as we have observed up to 97% Random queries with selectivity near 0. The result in Figure 14 is similar to Figure 13, except with slight increase in the RMS error of Isomer. Overall, we can make similar observations as before—consistent with our theory, selectivity is still learnable even if query distribution is independent of data.

Note that this setup unfairly disadvantages PtsHist, because it uses much less space per bucket than QuickSel and QuadHist. For example, assuming 2D, each bucket (point) in PtsHist requires 2 numbers, while each bucket (2D range) in QuickSel and QuadHist requires 4, twice that of PtsHist.

To be fair, we note that Isomer was not originally designed for batch training; instead, it builds its histogram by partitioning existing buckets upon observing the selectivity of each new query, which contributes to its slow training time in this setting.
Figure 9: RMS Error vs. model complexity on Data-driven workload of Power.

Figure 10: Model complexity vs. training size on Data-driven workload of Power.

Figure 11: RMS Error vs. training size on Data-driven workload of Power.

Figure 12: Training time vs. training size on Data-driven workload of Power.

Figure 13: RMS error vs. training size on random workload of Power.

Figure 14: RMS error vs. training size on non-empty queries in random workload of Power.

Figure 15: RMS Error vs. training size on Gaussian workload of Power.

Figure 16: RMS Error vs. Difference between training and test query workload.

Figure 17: RMS error vs. training size on orthogonal ranges.

Figure 18: RMS error vs. dimensions on Data-driven workload of Forest.

Figure 19: Training time vs. dimensions on Data-driven workload of Forest.

Figure 20: RMS error vs. training size on halfspace queries.
Besides RMS error, we also report Q-error results in Table 1 for all query workloads. The additional insight provided by Q-error is useful because a data-independent query workload (such as Random) on skewed data (such as Power) may result in many low-selectivity queries, where Q-error would be more informative. For example, we have observed up to 97% Random queries with selectivity near 0. From Table 1, we see that QUADHist and PtsHist, despite (and perhaps thanks to) their simplicity, are able to provide robust accuracy in terms of Q-error, often beating the more sophisticated QuickSel. QuickSel sometimes see Q-error larger than 50 even when the training size is up to 2000, while QUADHist and PtsHist have lower Q-errors even with just 50 training queries. The result on Q-error for non-empty queries are reported separately, for which PtsHist performs the best over all training sizes.

It is also instructive to dig deeper to see how QUADHist and PtsHist are able to work well on query workloads that do not correspond to the underlying data distribution. Figure 7 shows the set of buckets in QUADHist and PtsHist that are learned from the Random query workload over Power. Unknown to the learner, the real data is concentrated in the lower half. However, the Random query workload contains enough number of large queries that span both dense and sparse regions of the data space; the learner unfortunately only gets the overall selectivity of each such query, so some of the density “bleeds” into the sparse upper region, and we can see that the buckets are actually not ideal. Luckily, the subsequent weight assignment step (Section 3) mitigates this issue by assigning low weights to the upper region, making the resulting distribution more consistent with the underlying data distribution.

4.3 Training vs. Testing Query Distributions

Our next question is: what if training and testing query distributions do not match? The learning theory will not provide us with any guarantee on the performance over a different query workload, but in practice, if the test query workload is not completely disjoint from the training query workload, we should still expect to gain something from a learned model. In this set of experiments, we explore different combinations of training and testing query workloads. We use 2D Gaussian query workloads, but shift the Gaussian distribution (from which the center points of queries are drawn) such that its mean is located at (0.2, 0.2), (0.3, 0.3), (0.4, 0.4), (0.5, 0.5), (0.6, 0.6), or (0.7, 0.7) (while the covariance remains at 0.033). Figure 16 shows, as a heat map, the prediction error of QUADHist under each training/testing combination. First, we can see that when the training and test query workloads are the same (along the diagonal), the error is the smallest in most cases. If we fix the training query workload, say the column indexed by (0.6, 0.6), we observe that error gradually increases with the test query workload shifting away from (0.6, 0.6) to (0.2, 0.2) or (0.7, 0.7). Symmetrically, if we fix the test query workload, say the row indexed by (0.7, 0.7), we observe that the error gradually decreases with training query workload shifting from (0.3, 0.3) to (0.7, 0.7). In this case, even when the shift between two query workloads is large, there is still considerable overlap between their coverage of the underlying data space; hence the error remains manageable.

4.4 Effect of Dimensionality

Recall that Theorem 2.1 shows a training size $m = \tilde{O}_d((\frac{1}{\epsilon})^{f(d)})$, where $f(d) = 2d + 3$ for orthogonal range queries, $f(d) = d + 4$ for linear inequality queries, and $f(d) = d + 5$ for distance-based queries. For all these range types, we see an exponential dependency of sampling complexity $m$ on the dimensionality $d$. We now investigate this dependency experimentally. Recall that PtsHist is our method of choice in higher dimensions. For each setting of dimensionality $d$, we use a $d$-dimensional subspace of Forest and a $d$-dimensional Data-driven orthogonal range query workload; Figure 17 shows the error of PtsHist under different training sizes for the given $d$ as a line plot. The model complexity of PtsHist is always set to $4 \times$ the number of training queries, consistent with earlier experiments. For each line plot, we see that the error gradually decreases with more training queries, and eventually flattens out. As we increase $d$, we see the series of line plots pushing away from origin. Moreover, if we set the desired accuracy by drawing a horizontal line in Figure 17, its intersections with various line plots will show that as the dimensionality goes up, the number of training queries required to achieve this accuracy also goes up, as our theory predicates. (Figures on model complexity and training time are in the full version [3].) Since PtsHist’s training time primarily depends on the model complexity, which is pegged to the training size here, we do not see significant differences among different $d$. However, combining this observation with the observation from Figure 17 that a higher $d$ demands more training queries, we still conclude higher dimensions require longer training time.
Next, we compare \texttt{PtsHist} with other methods. \textsc{Isomer} is difficult to scale to higher dimensions due to the exponential dependency of its model complexity on \(d\), so we compare with \texttt{QuickSel} and our own \texttt{QuadHist} here. As before, the number of buckets used by \texttt{QuadHist} and \texttt{PtsHist} is set to be no larger than that of \texttt{QuickSel}. We vary the dimensionality of Forest from 2 to 10, using 1000 Data-driven training queries in each case. Figures 18 and 19 show the error and training time of the three methods. Overall, the three methods have competitive prediction accuracy, and all see larger error in higher dimensions. Since the model complexity is fixed (across \(d\)), differences in training time primarily come from solver speed (which can be highly situational and hard to interpret) and per-bucket computational cost. Thanks to \texttt{PtsHist}'s simpler buckets and lower per-bucket computational cost, we see that \texttt{PtsHist} holds significant advantage in terms of training time in high dimensions, which is the case that it is intended for.

### 4.5 Other Query Types

Beyond orthogonal range queries, we are also interested in other classes of range queries, such as halfspace and ball, which have many applications in databases but have seen much less work on selectivity estimation than orthogonal range queries, perhaps since the problem is perceived to be more difficult. From our theory in Section 2.2, however, selectivity functions for these queries are also learnable. This last set of experiments is designed to verify this claim. We focus on \texttt{QuadHist} and \texttt{PtsHist} show results across different dimensions for Forest data: Figures 20, 21 for halfspace queries and Figures 22, 23 for ball queries. All query workloads are Data-driven, and the model complexity is always no more than \(4 \times \) the number of training queries. We only show the results on \texttt{QuadHist} for \(d = 2\), since in higher dimensions, its prediction for halfspace and ball queries involve complicated intersection operations that make it too slow compared with \texttt{PtsHist}. In Figures 20 and 22, we see that error generally decreases as we increase the training size, and higher dimensionality generally requires a bigger training size to achieve the same level of accuracy. \texttt{QuadHist} is more accurate than \texttt{PtsHist} in 2D, although it is not applicable in higher dimensions. In terms of training time shown in Figures 21 and 23, we observe that \texttt{QuadHist} is slow than \texttt{PtsHist} in 2D, and as dimensionality increases, \texttt{PtsHist}'s training procedure remains very scalable, because its complexity is primarily dependent on its model complexity instead of \(d\). Overall, we observe that as simple as it is, \texttt{PtsHist} provides a reasonable solution for learning the selectivity of halfspace and ball queries, thanks to the theoretical guarantees on the learnability of these query classes.

**Summary.** Our main findings can be summarised as follow:

- The empirical results have verified our learning theory by showing a clear dependency between error \(\epsilon\), training size \(m\) and dimension \(d\). When fixing \(d\), more training samples lead to more accurate modeling of data distribution, thus smaller prediction error. On the other hand, when fixing \(m\), higher dimension leads to more coarse-grained modeling of data distribution, thus larger prediction error. (Section 4.1 and 4.4)

- The empirical results have verified that our learning theory holds for arbitrary data distribution and query distribution, even if they are highly skewed. (Section 4.2)

- Our learning theory does not provide us with any guarantee on the performance of learned model when the training and test query distributions do not match. The empirical results imply that we will get the most accurate model when training and test query distribution matches exactly, but we can still gain something from a learned model when there is overlap between their coverage of the underlying data space. (Section 4.3)

- Two simple algorithms proposed in Section 3 work well in practice, matching or even outperforming the state-of-art methods. \texttt{QuadHist} stands out for efficiency and accuracy in lower dimensional space while \texttt{PtsHist} scales better in higher dimensional space. (Section 4.1, 4.2 and 4.4)

### 5 RELATED WORK

**Orthogonal range queries.** We summarize current works on learning selectivity estimation for orthogonal range queries in Table 2 (see [46] for a comprehensive review) under two metrics: methodology and input. There are two main approaches: one is to build a mapping between queries and their selectivities via feature vectors, and the other is to learn data distribution. Depending on the input to models, we divide them into three cases: data only, query workload only, and both data and query (hybrid). Combining these two metrics, we review the following four categories:

- **Regression model with hybrid input.** \texttt{MSCN} [23] represents a query as a feature vector which contains three modules (i.e., table, join, and predicate) and uses a multi-set convolutional network for training. \texttt{MSCN} enriches the training queries with materialized samples from underlying data. \texttt{LW} [14] a lightweight selectivity estimation method, uses both queries and heuristic selectivity estimators as features. It adopts both neural network and gradient boost tree model separately for training.

- **Regression model only with query as input.** \texttt{DQM} [16] proposes one-hot encoding to encode categorical attributes (and treats numerical attributes as categorical attributes by automatic discretization), and uses a neural network for training.

- **Data distribution model only with data as input.** This line of work takes samples from the underlying data distribution. Both \texttt{Naru} [49] and \texttt{DQM-D} [28] decompose the data distribution into conditional data distributions using the product rule: \texttt{Naru} uses progressive sampling to sample values attribute by attribute according to the internal output of conditional probability distribution, and \texttt{DQM-D} selects samples in proportional to the contribution they make to the query cardinality according to the result from the previous stage. \texttt{DeepDB} [19] builds sum-product networks on random data samples to capture the data distribution.

- **Data distribution model only with query as input.** As far as we are aware, query-driven histograms [11, 20, 28, 29, 36, 39] are the methods that learn data distribution only from previous queries, for example \texttt{STHoles} [11] exploits results of queries in the workload and gathers associated statistics to progressively build and refine a histogram; \texttt{Isomer} [39] applies \texttt{STHoles} for histogram bucket creations, and computes the density for buckets by maximizing entropy distribution; \texttt{QuickSel} [36] uses a mixture model of uniform distributions to represent the underlying data distribution, which can be viewed as overlapping histograms.
Table 1: Q-error over Power. The bold numbers are those ranking the smallest in 99th Q-error.

<table>
<thead>
<tr>
<th>Models</th>
<th>Data</th>
<th>Hybrid</th>
<th>Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regression</td>
<td>-</td>
<td>MSCN [23]</td>
<td>DMQ-Q [16]</td>
</tr>
<tr>
<td></td>
<td>DMQ-D [16]</td>
<td>QuickSel [36]</td>
<td>QuickSel [36]</td>
</tr>
</tbody>
</table>

**Distance-based queries.** Deep learning has been studied to estimate the selectivity for distance-based queries. Wang et al. [47] learned the cardinality by resorting to the VAE (Variational Autoencoders) and embeddings for different thresholds separately for enhancing accuracy and guaranteeing monotonicity. Sun et al. [41] utilized deep neural network to learn cardinality and adopt two strategies to improve the accuracy and reduce the size of training data, i.e., query segmentation and data segmentation. There are some other methods depending on the underlying data, for example, clustering-based methods [10], kernel-based methods [31].

**Learning theory.** It is beyond the scope of this paper to review the relevant work on learning theory and we refer reader to a few classical books on this topic [6, 22, 44]. In the literature of ML, Valiant’s pioneering work [43] first established the notion of learnability and the framework of **probably approximately correct** (PAC) learning. Intuitively, the learner in this framework receives random samples from underlying training data, and aims to select a hypothesis from a set of possible hypotheses which has low generalization error ($\epsilon$) with high probability $1 - \delta$ on the unobserved samples from the same distribution. Surprisingly, relationships between the PAC-learnability of a hypothesis class and its inherent properties have been proved. For example, a Boolean hypothesis class is learnable if and only if it has finite VC-dimension [6, 44, 45], and a real-valued function class is PAC-learnable if and only if it has finite fat-shattering dimension [6, 8, 21]. This framework of PAC learning is exactly the theoretical foundation behind our investigation of learnability of selectivity functions.

**6 CONCLUSIONS AND FUTURE WORK**

In this paper, we presented an ML-based technique for estimating the selectivity of selection queries in DB systems. Central to our approach were generalization bounds that we proved for this problem (Theorem 2.1), thereby establishing a formal framework for applying classical ML theory (PAC learning) to DB problems. In contrast, the predominant approach in previous work has been to use deep learning techniques, which have consistently outperformed traditional optimization methods for a range of important problems in DB research. However, in spite of much empirical success, obtaining generalization bounds for deep learning remains one of the outstanding open challenges of the modern era. We expect the trend of using ML for DB will accelerate even further in the future, and hope that our work in initiating a formal study of the learnability of DB problems will complement the existing efforts at leveraging deep learning for improving the performance of DB systems in practice. There are several interesting directions for future work. As mentioned earlier, understanding the complexity of finding an optimal distribution with a given model complexity is an open problem. Although our framework does not assume query ranges to be bounded and thus works even if we consider data distributions with unbounded support, e.g., Gaussian mixtures, developing an algorithm that computes a Gaussian mixture (or another model) with a small loss given a training sample is also an open problem. Developing theory and algorithms for learning selectivity of range queries of practical interest with unbounded VC dimensions and extending the learning framework to unsupervised approaches are other intriguing directions for future work.
REFERENCES